

Design IoT Projects with Raspberry Pi, Arduino and ESP32

Programming with Node-RED

The Node-RED interface shows a complex flowchart with the following components and connections:

- Inputs:** Two ESP32 MAC addresses (etsp05/enocean/00 8B FA E2 and 00 8B 76 EB) connect to 'Switch' and 'Rocker' nodes respectively.
- Control:** 'Switch' and 'Rocker' nodes connect to 'Lab Lamp', 'Lab Fan', and 'Red LED Appear' nodes.
- Data Processing:** A 'tweets' input connects to 'Split out English tweets', which then connects to 'sentiment' and 'Filter Positive Tweets' nodes. A 'debug' node is also connected to the 'sentiment' node.
- Outputs:** 'Filter Positive Tweets' connects to 'Red LED Scroll' and another 'debug' node.
- Hardware Control:** Three buttons labeled 'Red', 'Green', and 'Blue' connect to a 'Big Clock (top)' node.
- UI Generation:** A 'lab/fixed/iris/id' input connects to a 'create UIF message' node, which then connects to 'To Realtime' and 'ETSlab' nodes.
- Debug:** A 'debug' node is connected to the bottom of the flowchart.

Below the Node-RED interface, a diagram illustrates the hardware setup:

- A Raspberry Pi is connected to a Wi-Fi Router.
- A NodeMCU (Arduino) is connected to a Button and a LED.
- The NodeMCU is also connected to the Wi-Fi Router.
- The Raspberry Pi is connected to the Wi-Fi Router and the LED.

On the right side, three gauge charts display real-time data:

- Ambient Temperature:** 29.07 C
- Ambient Humidity:** 43.66 %
- Ambient Pressure:** 978.1 hPa

Each gauge chart includes a line graph showing the data over time.

Programmation avec Node-RED

Concevoir des projets IoT avec Raspberry Pi, Arduino et ESP32



Dogan Ibrahim



elektor



● Il s'agit d'une publication Elektor. Elektor est la marque de médias de

Elektor International Media B.V.

78 York Street, London W1H 1DP, UK

Téléphone : (+44) (0)20 7692 8344

● Tous droits réservés. Aucune partie de ce livre ne peut être reproduite sous quelque forme que ce soit, y compris

photocopier ou stocker sur tout support par des moyens électroniques et, même de façon transitoire ou accessoire, dans un autre contexte de la présente publication, sans l'autorisation écrite du titulaire des droits d'auteur, sauf conformément aux dispositions de la Copyright Designs and Patents Act 1988 ou aux termes d'une licence délivrée par l'Agence de licences de droit d'auteur Ltd., 90 Tottenham Court Road, London, England W1P 9HE. Les demandes d'autorisation de reproduction de toute partie de la publication doivent être adressées aux éditeurs.

● Déclaration

L'auteur et l'éditeur ont fait de leur mieux pour s'assurer que les informations contenues dans ce livre sont correctes. Ils n'assument aucune responsabilité envers une partie pour toute perte ou dommage causé par des erreurs ou omissions dans ce livre, que ces erreurs ou omissions résultent d'une négligence, d'un accident ou de toute autre cause.

● British Library Catalogage in Publication Data

Un catalogue de ce livre est disponible à la British Library

● **ISBN 978-1-907920-88-2**

© Copyright 2020 : Elektor International Media b.v.

Production de prépresse : D-Vision, Julian van den

Berg Première publication au Royaume-Uni en 2020



Elektor fait partie d'EIM, la principale source mondiale d'informations techniques essentielles et de produits électroniques pour les ingénieurs professionnels, les concepteurs en électronique et les entreprises qui cherchent à les engager. Chaque jour, notre équipe internationale développe et diffuse des contenus de qualité - via une variété de canaux médias (y compris les magazines, la vidéo, les médias numériques et les médias sociaux) dans plusieurs langues - liés à la conception électronique et l'électronique de bricolage. www.elektormagazine.com

.....
Chapitre 1 • Raspberry Pi 4	15
1.1 Aperçu	15
1.2 Pièces de la Raspberry Pi 4	15
1.3.1 Option de configuration 1	18
1.3.2 Option de configuration 2	19
Résumé	19
Chapitre 2 • Installation du système d'exploitation Raspberry Pi	20
2.1 Aperçu	20
2.2 Étapes d'installation de Raspbian Buster sur Raspberry Pi 4	20
2.3 Accès à distance	22
2.4 Utilisation de la pâte	23
2.4.1 Configuration de Putty	24
2.5 Accès à distance du bureau	25
2.6 Résumé	26
Chapitre 3 • Utilisation de la ligne de commande	27
3.1 Aperçu	27
3.2 La structure de répertoire du Raspberry Pi	27
3.3 Autorisations de fichier	28
3.4 Aide	32
3.5 Date, heure et calendrier	33
3.6 Opérations de fichier	33
3.7 Informations sur le système et l'utilisateur	35
3.8 Surveillance des ressources sur Raspberry Pi	37
3.9 Fermeture	39
3.10 Résumé	40
Chapitre 4 • Installation du logiciel Node-RED sur le Raspberry Pi	41
4.1 Aperçu	41
4.2 Installation du noeud RED de Raspberry Pi	41
4.3 Interface Node-RED vers le monde externe	43
4.4 Disposition de l'écran Node-RED	44
4.5 Projet 1 – Bonjour tout le monde!	44

4.6	Projet 2 – Date et heure.	47
4.7	Projet 3 – Conversion de la température.	52
4.8	Importing and exporting flow programs	54
4.8.1	Flux d'exportation.	54
4.8.2	Importation de flux	55
4.9	Copier des nœuds dans le même espace de travail	55
4.10	Nœuds centraux	55
4.10.1	Nœuds d'entrée	55
4.10.2	Nœuds de sortie	56
4.10.3	Nœuds de fonction	57
4.10.4	Nœuds sociaux.	58
4.10.5	Nœuds de stockage	58
4.10.6	Nœuds d'analyse	58
4.10.7	Nœuds avancés	58
4.10.8	Noeuds de Raspberry Pi	58
4.11	Projet 4 – Numéro des dés	59
4.12	Projet 5 – Double dé	60
4.13	Projet 6 – Conversions d'unités - Entrées multiples pour une fonction	62
4.14	Entrées multiples et sorties multiples d'une fonction	65
4.15	Projet 7 – Moyenne des nombres - Utilisation du nœud lisse.	67
4.16	Projet 8 – Carrés de nombres.	68
4.17	Projet 9 – Obtenir les bulletins météorologiques – Afficher le bulletin météorologique local	70
4.18	Projet 10 – Afficher la température actuelle	73
4.19	Projet 11 – Envoi de la température actuelle à un courriel	74
4.20	Projet 12 – Envoi de la température et de l'atmosphère actuelles pression sur votre compte Twitter	76
4.21	Configuration de Node-RED	78
4.22	Résumé	79
Chapitre 5 • Projets Raspberry Pi basés sur le noeud RED utilisant GPIO		80
5.1	Aperçu	80
5.2	GPIO – Interface parallèle.	80
5.3	Projet 13 – Commande DEL.	82
5.4	Projet 14 – LED clignotante.	85

5.4.1 Utilisation d'une variable de contexte pour clignoter la LED.	87
5.5 Projet 15 – LED clignotantes à la fois	88
5.5.1 Utilisation de variables de contexte pour faire clignoter les LED alternativement.	89
5.6 Projet 16 – Alarme de température avec buzzer.	90
5.7 Projet 17 – Contrôle à distance d'un port GPIO par courriel	93
5.8 Projet 18 – Confirmation de l'état du buzzer	95
5.9 Projet 19 – Contrôle à distance de plusieurs ports GPIO par courrier électronique.	98
5.10 Projet 20 – Simulateur de feux de circulation	102
5.11 Projet 21 – Entrée du commutateur à bouton-poussoir	106
5.12 Projet 22 – Modification de la luminosité des DEL – Sortie PWM	110
5.13 Résumé	112
6.1 Aperçu	113
6.2 Raspberry Pi I ² Ports C	114
6.3 I ² C LCD	115
6.4 Installation du I ² Logiciel LCD C sur Node-RED	115
6.5 Projet 23 – Affichage de l'heure actuelle sur l'écran LCD	118
6.6 Projet 24 – Affichage de la température et de l'humidité locales sur l'écran LCD.	120
6.7 Projet 25 – Affichage des nombres de dés.	120
6.8 Projet 26 – Comptoir d'événements avec écran LCD	122
6.9 Projet 27 – Capteur de température et d'humidité DHT11 avec écran LCD.	124
6.10 Projet 28 – Capteur de distance à ultrasons avec écran LCD	129
6.11 Projet 29 – Alarme de distance à ultrasons avec buzzer	134
6.12 Projet 30 – Système de stationnement à ultrasons avec buzzer	135
6.13 Utilisation d'un écran LCD parallèle.	137
6.14 Projet 31 – Affichage du message sur un écran LCD parallèle	139
6.15 Résumé	141
Chapitre 7 • Utilisation de l'ADC dans les projets Raspberry Pi Node-RED	142
7.1 Aperçu	142
7.2 Le MCP3002 ADC.	142
7.3 Projet 32 – Voltmètre avec sortie LCD	144
7.4 Projet 33 – Mesure de la température à l'aide d'un capteur analogique	146
7.5 Projet 34 – Contrôle de la température ON/OFF.	148

7.6 Résumé	152
Chapitre 8 • La palette du tableau de bord	153
8.1 Aperçu	153
8.2 Installation du tableau de bord	153
8.3 Projet 35 – Utilisation d'une jauge pour afficher la température	154
8.4 Utilisation d'un graphique linéaire pour afficher la température	156
8.5 Utilisation d'un graphique à barres pour afficher la température	157
8.6 Projet 36 – Utilisation de jauges pour afficher la température et l'humidité	158
8.7 Projet 37 – Utilisation de plusieurs jauges	160
8.8 Projet 38 – Utiliser un curseur pour modifier la luminosité de la DEL	161
8.9 Projet 39 – Utilisation de nœuds à bouton pour commander une DEL	163
8.10 Projet 40 – Utilisation de commutateurs et de nœuds texte pour commander une DEL	165
8.11 Projet 41 – Prévisions météorologiques	166
8.12 Configuration du tableau de bord	169
8.13 Résumé	169
Chapitre 9 • Projets basés sur le réseau UDP/TCP Wi-Fi	170
9.1 Aperçu	170
9.2 Projet 42 – Contrôle d'une DEL à partir d'un téléphone mobile – Communication basée sur le PPU	
9.3 Projet 43 – Contrôle de plusieurs LED à partir d'un téléphone mobile – basé sur le protocole UDP communication	173
9.5 Projet 45 – Contrôle d'une DEL à partir d'un téléphone mobile – Communication basée sur TCP	178
9.6 Projet 46 – Contrôle de plusieurs LED à partir d'un téléphone mobile – Communication basée sur TCP	181
9.7 Projet 47 – Envoi de la température et de l'humidité au téléphone mobile – Communication basée sur TCP	182
9.8 Projet 48 – Programme de clavardage – Clavardage du téléphone mobile au Raspberry Pi	184
9.9 Projet 49 – Utilisation du ping	187
9.10 Résumé	189
Chapitre 10 • Nœuds de stockage	190
10.1 Aperçu	190
10.2 Projet 50 – Stockage de la température et des données horodatées données d'humidité dans un fichier	190

10.3	Projet 51 – Lecture du contenu d'un fichier	192
10.4	Projet 52 – Lecture d'un fichier ligne par ligne	195
10.5	Résumé	197
Chapitre 11 • Communication en série		198
11.1	Aperçu	198
11.2	Projet 53 – Communication avec Arduino sur ligne série	199
11.3	Projet 54 – Réception de données en série – Réception de données GPS	204
11.4	Projet 55 – Réception des données GPS – Extraction de la latitude et de la longitude.	208
11.5	Projet 56 – Afficher notre emplacement sur une carte.	212
11.6	Projet 57 – Envoi de données série à arduino	216
11.7	Projet 58 – Connexion de Raspberry Pi et d'Arduino Uno à l'aide de ports USB.	220
11.8	Projet 59 – Envoi de données série du Raspberry Pi à l'Arduino par radio RF	222
11.9	Résumé	224
Chapitre 12 • Utilisation de la HAT sensée		225
12.1	Aperçu	225
12.2	La carte HAT de Sense	225
12.3	Node-RED Sense HAT nodes	226
12.4	Projet 60 - Affichage de la température, de l'humidité, et la pression (événements environnementaux)	227.
12.5	Projet 61 - Affichage du cap de la boussole (événements de mouvement).	229
12.6	Projet 62 - Affichage de l'accélération (événements de mouvement)	230
12.7	Projet 63 - Affichage de l'orientation (événements de mouvement)	231
12.8	Utilisation du joystick	233
12.9	Utilisation de la matrice LED	234
12.10	Projet 64 – Feux DEL clignotant aléatoirement de couleurs aléatoires	238
12.11	Projet 65 – Affichage de la température par comptage DEL	239
12.12	Affichage et défilement des données sur la matrice LED	242
12.13	Projet 66 – Défilement des valeurs de pression sur la matrice DEL	242
12.14	Résumé	244
Chapitre 13 • Node-RED avec Arduino Uno		245
13.1	Aperçu	245
13.2	Installation de Node-RED pour Arduino Uno	245

13.3	Projet 67 – LED clignotante	247
13.4	Projet 68 – Affichage de la température ambiante dans la fenêtre de débogage	248
13.5	Projet 69 – Affichage de la température ambiante dans le tableau de bord	250
13.6	Projet 70 – Affichage de la température ambiante sous forme de jauge et de graphique	251
13.7	Utilisation du port série Arduino Uno	253
13.7.1	Projet 71 – Utilisation du DHT11 avec l'Arduino	253
13.8	Résumé	256
Chapitre 14 • Utilisation du DevkitC ESP32 avec Node-RED		257
14.1	Aperçu	257
14.2	ESP32 DevKitC et Node-RED	259
14.3	Projet 72 – Contrôle d'une DEL connectée à ESP32 DevKitC	259
14.4	Résumé	262
Chapitre 15 • Utilisation d'Amazon Alexa dans les projets Node-RED		263
15.1	Aperçu	263
15.2	Projet 73 – Contrôle d'une DEL à l'aide d'Alexa	263
15.3	Projet 74 – Contrôle d'une DEL et d'un buzzer à l'aide d'Alexa	265
15.4	Résumé	268
Chapitre 16 • Accès au Raspberry Pi Node-RED depuis n'importe où		269
16.1	Aperçu	269
16.2	Le ngrok	269
16.3	Démarrage automatique de Node-RED au redémarrage	271
16.4	Résumé	271
Chapitre 17 • Utilisation de la technologie Bluetooth avec Node-RED		272
17.1	Aperçu	272
17.2	Projet 75 – Commande d'une LED et d'un buzzer via Bluetooth	272
17.3	Résumé	276
Chapitre 18 • Node-RED et MQTT		278
18.1	Aperçu	278
18.2	Comment fonctionne le MQTT	278
18.3	Le courtier de Mosquitto	280
18.4	Utilisation de MQTT dans la domotique et les projets IoT	281
18.5	Projet 76 – Contrôle d'une DEL au moyen de la technologie MQTT	282

18.6 Le processeur ESP8266	283
18.7 Projet 77 – Clignotement d'une LED à l'aide de l'unité NodeMCU ESP8266	285
18.8 Utilisation de l'ESP8266 NodeMCU avec MQTT	287
18.9 Projet 78 – Contrôle d'une DEL à l'aide de l'unité NodeMCU ESP8266 avec MQTT – LED connectée à Raspberry Pi	288
18.10 Projet 79 – Contrôle d'une DEL à l'aide de la norme NodeMCU ESP8266 avec MQTT – DEL connecté à ESP8266 NodeMCU	293
18.11 Résumé	299
Chapitre 19 • Utilisation de HTTP dans les projets Node-RED	300
19.1 Aperçu	300
19.2 Utilisation de HTTP GET.	300
19.3 Serveur Web	301
19.4 Projet 80 – Contrôle de 4 relais à l'aide du serveur Web	302
19.5 Résumé	308
Annexe A • Le nœud de fonction	309
A.1 Aperçu	309
A.2 Variables	309
A.3 Sorties multiples	310
A.4 Manipulation de chaînes	310
A.5 Fonctions mathématiques	313
A.6 Conversion des nombres et vérification des nombres	314
A.7 Date	315
A.8 Tableaux	315
A.9 Énoncés conditionnels	316
A.10 Répétition (boucles)	317
A.12 Exemples	319
Annexe B • Programmes de flux utilisés dans le livre	321
A.1 Aperçu	321
A.2 Utilisation des programmes de flux	321
Annexe C • Composants utilisés dans le livre	322

Préface

Il devient important pour les utilisateurs de microcontrôleurs de s'adapter rapidement aux nouvelles technologies et d'apprendre l'architecture et l'utilisation des microcontrôleurs 32 bits haute performance. Plusieurs fabricants proposent des microcontrôleurs 32 bits en tant que processeurs à usage général dans les applications intégrées. L'architecture ARM est actuellement l'une des architectures de microcontrôleurs 32 bits les plus utilisées dans les appareils mobiles, tels que les téléphones portables, les iPads, les consoles de jeux et de nombreux autres appareils portables.

Raspberry Pi est basé sur l'architecture ARM et il est actuellement l'un des ordinateurs à carte unique les plus utilisés par les étudiants, les ingénieurs et les amateurs. Il y a des idées de projets basés sur Raspberry Pi disponibles sur internet

Arduino est également une carte de développement de microcontrôleurs très populaire. Bien qu'il soit basé sur une architecture 8 bits, il est largement utilisé car il est supporté par un grand nombre de bibliothèques logicielles, ce qui facilite le développement de projets en relativement peu de temps.

Un autre microcontrôleur populaire est ESP32. Il est largement vendu comme carte de développement connue sous le nom de ESP32 DevKitC. La raison pour laquelle l'ESP32 est très populaire est qu'il a une capacité Wi-Fi et Bluetooth intégrée, de nombreux ports d'entrée numériques et analogiques et des minuteries intégrées. Par ailleurs, sa consommation électrique est très faible et il dispose d'un processeur qui peut être mis en mode veille, ce qui consomme un courant extrêmement faible.

L'Internet des objets (IdO) devient un domaine d'application majeur des systèmes embarqués. Par conséquent, de plus en plus de gens s'intéressent à la conception et à la programmation intégrées. De plus, nous pouvons voir que davantage d'écoles techniques et d'universités s'éloignent des microcontrôleurs 8 bits et 16 bits existants et introduisent l'intégration 32 bits. Certaines applications de l'IdO exigent une précision, une puissance de traitement élevée et une consommation d'énergie très faible.

Node-RED est un éditeur visuel open source pour le câblage de l'Internet des objets produit par IBM. Node-RED est livré avec un grand nombre de nœuds pour gérer une variété de tâches. Les nœuds requis sont sélectionnés et assemblés pour effectuer une tâche particulière. Node-RED est basé sur la programmation de type flux où les nœuds sont configurés et joints ensemble pour former un programme d'application. Il y a des nœuds pour faire des tâches très complexes, y compris l'accès web, Twitter, E-mail, HTTP, Bluetooth, MQTT, le contrôle des ports GPIO, etc. La bonne chose à propos de Node-RED est que le programmeur n'a pas besoin d'apprendre comment écrire des programmes complexes. Par exemple, un courriel peut être envoyé en joignant quelques nœuds et en écrivant quelques lignes de code.

Le but de ce livre est d'enseigner comment Node-RED peut être utilisé dans des projets. Les principales procédures- Le modèle utilisé dans la majorité des projets de ce livre est le Raspberry Pi 4. Les chapitres sont inclus pour montrer comment Node-RED peut être utilisé avec Arduino Uno, ESP32 DevKitC et les cartes de développement de microcontrôleurs NodeMCU ESP8266.

De nombreux exemples de projets sont donnés dans le livre. Tous les projets ont été entièrement testés et étaient en cours d'exécution au moment où ce livre a été écrit. Les utilisateurs peuvent sélectionner des programmes de flux des projets à partir du site web du livre et les configurer pour répondre à leurs propres applications. Le fonctionnement de chaque programme de flux est entièrement décrit dans le livre.

J'espère que vous apprécierez la lecture de ce livre et que vous pourrez utiliser Node-RED avec plaisir dans vos futurs projets.

Prof Dr. Dogan Ibrahim
London, 2020

Chapitre 1 • Raspberry Pi 4

1.1 Aperçu

Le Raspberry Pi est devenu récemment l'un des ordinateurs monocartes les plus populaires et les plus puissants utilisés par les étudiants, les amateurs et les ingénieurs professionnels. Raspberry Pi 4 est la version la plus récente et la plus puissante de Raspberry Pi. Dans ce chapitre, nous allons examiner les spécifications et les exigences de base de l'ordinateur Raspberry Pi 4. Ce qui est inclus dans ce chapitre peut facilement être appliqué à d'autres modèles de la famille Raspberry Pi.

1.2 Pièces de la Raspberry Pi 4

Tout comme ses versions antérieures, le Raspberry Pi 4 est un ordinateur à carte unique qui présente les caractéristiques de base suivantes :

- CPU quad-core 64 bits 1,5 GHz
- 1 Go, 2 Go ou 4 Go de RAM
- Wi-Fi IEEE 802.11ac 2,4 GHz et 5,0 GHz
- Bluetooth 5.0 BLE
- Gigabit Ethernet
- 2 x ports USB 3.0, 2 x ports USB 2.0 et 1 x ports USB-C
- 2 x ports micro-HDMI pour double affichage, prenant en charge jusqu'à la résolution 4K
- Ports d'affichage DSI et de caméra CSI
- Slot pour carte micro SD pour le système d'exploitation et le stockage des données
- Port audio stéréo et vidéo composite 4 pôles
- Connecteur GPIO 40 broches
- Alimentation par Ethernet (PoE) activée avec le HAT PoE
- Graphiques OpenGL ES 3.0

La figure 1 montre la carte Raspberry Pi 4 avec ses principaux composants identifiés.

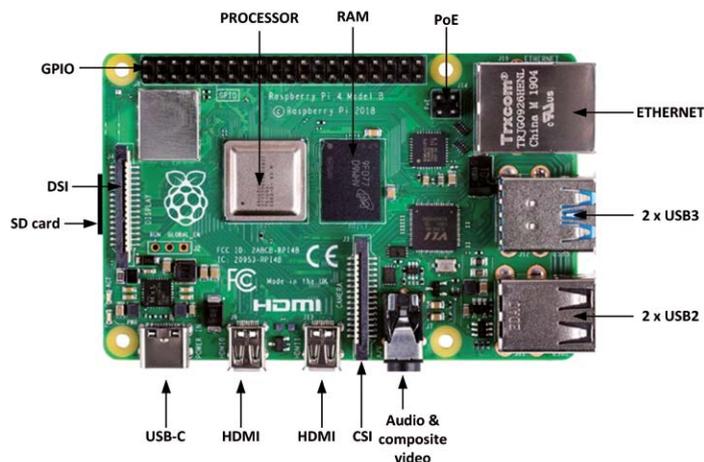


Illustration 1.1 Carte Raspberry Pi 4

Une brève description des différents composants de la carte est donnée ci-dessous :

Processeur : le processeur est enfermé dans un bouchon en métal et il est basé sur Broadcom BCM2711B0, qui se compose d'un noyau Cortex A-72, fonctionnant à 1.5GHz.

RAM : Il existe 3 versions de Raspberry Pi 4 en fonction de la quantité de RAM DDR4 re- Requis : 1 Go, 2 Go et 4 Go.

Ports USB : Raspberry Pi 4 comprend 2 x USB 3.0, 2 x USB 2.0 et 1 x ports USB-C. Le débit de transfert des données USB 3.0 est de 4800 Mbps (mégabits par seconde), tandis que l'USB 2.0 peut transférer jusqu'à 480 Mbps, soit 10 fois plus lent que l'USB 2.0. Le port USB-C permet de connecter la carte à une source d'alimentation appropriée.

Ethernet : Le port Ethernet permet de connecter la carte directement à un port Ethernet sur un routeur. Le port prend en charge les connexions Gigabit (125Mbps).

HDMI : Deux ports micro HDMI sont fournis qui prennent en charge des résolutions d'écran jusqu'à 4K. Les adaptateurs HDMI peuvent être utilisés pour connecter la carte à des périphériques HDMI de taille standard.

GPIO : Un connecteur à 40 broches est fourni en tant que GPIO (General Purpose Input Output). Ceci est compatible avec les ports GPIO antérieurs.

Port audio et vidéo : Une prise jack de 3,5 mm est fournie pour l'audio stéréo et com- Interface vidéo appropriée. Des écouteurs peuvent être connectés à ce port. Amplificateur externe de- Les services seront tenus de connecter des haut-parleurs à ce port. Ce port prend également en charge la vidéo composite, permettant aux téléviseurs, projecteurs et autres dispositifs d'affichage compatibles avec la vidéo composite d'être connectés au port.

Port CSI : Il s'agit du port de caméra (Camera Serial Interface), permettant à une caméra compatible d'être connectée au Raspberry Pi.

Port DSI : Il s'agit du port d'affichage (interface série d'affichage), permettant de connecter un écran compatible (par ex. écran Raspberry Pi 7 pouces) au Raspberry Pi.

Port PoE : Ceci est un connecteur 4 broches, permettant au Raspberry Pi de recevoir l'alimentation à partir d'un réseau- connexion de travail.

Carte micro SD : Cette carte est montée sur le support de carte placé au bas du tableau et elle contient le logiciel du système d'exploitation ainsi que les données du système d'exploitation et de l'utilisateur. Exigences du Raspberry Pi 4

Comme indiqué ci-dessous, un certain nombre de périphériques externes sont nécessaires avant que le Raspberry Pi puisse être utilisé :

- Alimentation
- Carte Micro SD

- Logiciel du système d'exploitation
- Clavier et souris USB
- Un câble micro HDMI pour recevoir des signaux audio et vidéo
- Écran ou téléviseur compatible HDMI (vous aurez peut-être besoin d'adaptateurs micro HDMI vers DVI-D ou VGA. Un câble et une prise de type 3,5 mm TRRS seront nécessaires si vous utilisez un ancien téléviseur avec vidéo composite)

Alimentation : Une alimentation 5V 3A avec un connecteur de type USB-C est nécessaire. Vous pouvez soit acheter l'alimentation officielle Raspberry Pi 4 (Figure 1.2) ou utiliser un adaptateur USB-C- Fournir de l'énergie à partir d'une source externe.



Figure 1.2 Alimentation électrique officielle du Raspberry Pi 4

Carte micro SD : Il est recommandé d'utiliser une carte micro SD avec une capacité d'au moins 8 Go, bien que la plus grande capacité (par ex. 16 Go ou 32 Go) soit préférable car il y aura de la place pour grandir à l'avenir. Une carte de classe 10 (ou plus rapide) est recommandée.

Système d'exploitation : Vous pouvez acheter le système d'exploitation préchargé sur une carte micro SD, connue sous le nom de NOOBS (New Out Of Box Software) qui nécessite une configuration minimale avant qu'il soit entièrement fonctionnel. L'alternative est d'acheter une carte micro SD vierge et de télécharger le système d'exploitation sur cette carte. Les étapes pour préparer une nouvelle carte micro SD avec l'op- Le système de classification est présenté dans le chapitre suivant.

Clavier et souris USB : Vous pouvez utiliser une paire de souris et de clavier sans fil ou filaire. Si vous utilisez une paire câblée, vous devez connecter le clavier à l'un des ports USB et la souris à un autre port USB. Si vous utilisez un clavier et une souris sans fil, vous devez connecter le dongle sans fil à l'un des ports USB.

Affichage : Un moniteur d'affichage compatible HDMI standard avec un adaptateur micro HDMI vers HDMI standard peut être utilisé. Vous pouvez également utiliser un moniteur d'affichage de type VGA avec un adaptateur micro HDMI vers VGA ou un adaptateur DVI-D. Si vous avez un vieux téléviseur avec une vidéo composite interface, vous pouvez alors le connecter au port Raspberry Pi 3.5mm avec un connecteur de type TRRS. Vous pouvez également envisager d'acheter des pièces supplémentaires, telles qu'un boîtier, un ventilateur de processeur, etc. Le boîtier est très utile car il protège votre électronique Raspberry Pi. La température de travail du processeur peut atteindre 80 degrés centigrades. Utiliser un ventilateur (voir la figure 1.3) rend le processeur plus efficace car il peut abaisser sa température d'environ 50%.



Illustration 1.3 Ventilateur du processeur Raspberry Pi 4 (www.seeedstudio.com)

1 3 1 Option de configuration 1

Comme le montre la figure 1.4, dans cette option, divers appareils sont connectés directement au Raspberry Pi 4. Selon le type de moniteur d'affichage que nous avons, nous pouvons utiliser un écran HDMI, un moniteur VGA, un moniteur DVI-D ou un téléviseur. Notez que selon les périphériques USB externes utilisés, vous pouvez utiliser soit le port USB 2.0 ou le port USB 3.0.

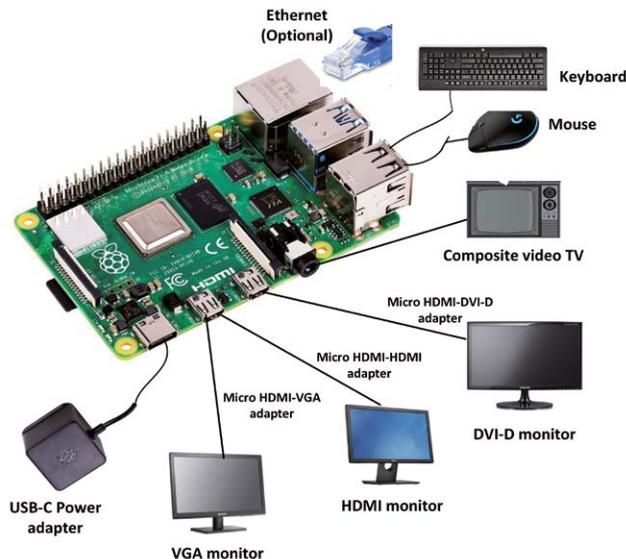


Illustration 1.4 Configuration du Raspberry Pi 4 - option 1

1 32 Option de configuration 2

Dans cette option, illustrée à la figure 1.5, un concentrateur d'alimentation est utilisé pour connecter les périphériques USB.

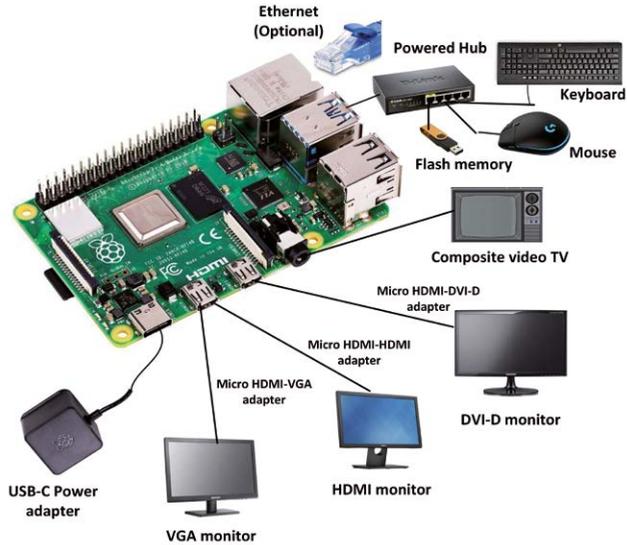


Illustration 1.5 Configuration de Raspberry Pi 4 – option 2

Résumé

Dans ce chapitre, nous avons appris sur les principales parties et leurs fonctions de la carte Raspberry Pi 4. En outre, nous avons appris comment configurer le Raspberry Pi 4.

Dans le prochain chapitre, nous allons apprendre à installer le dernier système d'exploitation Raspberry Pi, Raspbian Buster, sur une nouvelle carte micro SD vierge.

Chapitre 2 • Installation du système d'exploitation Raspberry Pi

2.1 Aperçu

Dans le dernier chapitre, nous avons examiné certaines des fonctionnalités matérielles de Raspberry Pi 4 et appris comment configurer le matériel à l'aide de divers périphériques externes. Dans ce chapitre, nous allons apprendre à installer le dernier système d'exploitation Raspberry Pi, Raspbian Buster, sur une carte SD.

2.2 Étapes d'installation de Raspbian Buster sur Raspberry Pi 4

Raspbian Buster est le dernier système d'exploitation de Raspberry Pi 4. Cette section présente les étapes à suivre pour installer ce système d'exploitation sur une nouvelle carte SD vierge, prête à être utilisée avec votre Raspberry Pi 4. Vous aurez besoin d'une carte micro SD d'au moins 8 Go (16 Go sont préférables) avant d'installer le nouveau système d'exploitation.

Les étapes d'installation de Raspbian Buster sont les suivantes :

- Télécharger l'image de Buster dans un dossier sur votre PC (p. ex., C : RPIBuster) à partir du lien suivant en cliquant sur le fichier ZIP à la section Télécharger **Raspbian Buster avec logiciel de bureau et recommandé** (voir la figure 2.1). Au moment de l'écriture de ce livre, le fichier s'appelait : **2019-07-10-raspbian-buster-full.img**. Vous devrez peut-être utiliser le logiciel Windows 7Zip pour décompresser le téléchargement car certaines des fonctionnalités ne sont pas prises en charge par les anciens logiciels de décompression.

<https://www.raspberrypi.org/downloads/raspbian/>

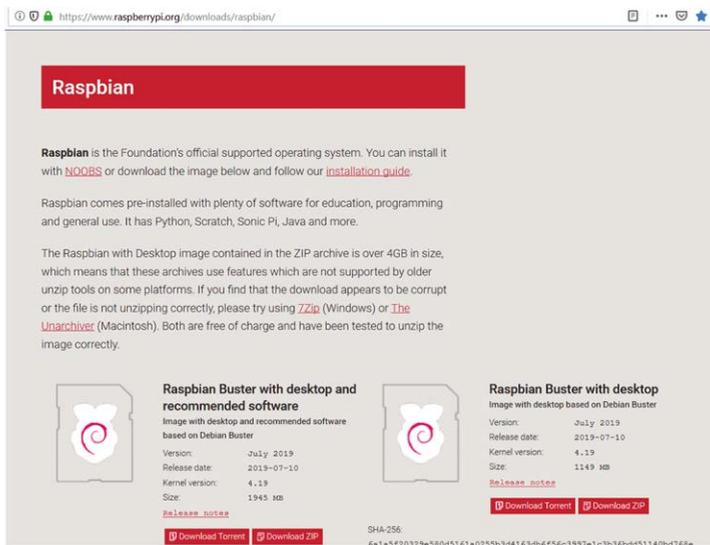


Figure 2.1 Page de téléchargement de Raspbian Buster

- Insérez votre carte micro SD vierge dans la fente de votre ordinateur. Vous devrez peut-être utiliser un adaptateur pour le faire.

- Télécharger le programme Etcher sur votre PC pour faire clignoter l'image du disque. Le lien est (voir la figure 2.2) :

<https://www.balena.io/etcher/>

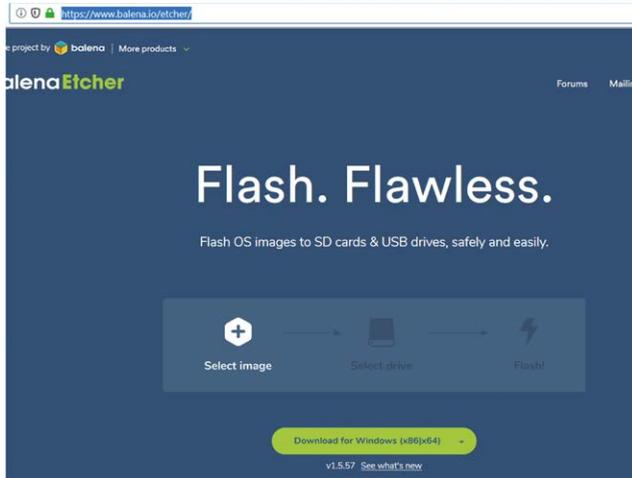


Figure 2.2 Télécharger l'etcheur

- Double-cliquez pour ouvrir Etcher, puis cliquez sur **Sélectionner l'image**. Sélectionnez le fichier Raspbian Buster que vous venez de télécharger.
- Cliquez **Sélectionner la cible** et sélectionnez la carte micro SD.
- Cliquez **Flash** (voir la figure 2.3). Cela peut prendre plusieurs minutes, attendez que le- La carte micro SD est alors démontée. Vous pouvez retirer votre carte micro SD une fois qu'elle est démontée.



Figure 2.3 Cliquez sur Flash pour faire clignoter l'image disque

- Vous êtes maintenant prêt à utiliser votre carte micro SD sur votre Raspberry Pi 4.

- Connectez votre Raspberry Pi 4 à un moniteur HDMI (vous devrez peut-être utiliser un câble adaptateur pour la conversion mini-HDMI en HDMI standard), branchez un clavier USB et mettez le Raspberry Pi sous tension.
- Le menu de démarrage s'affiche sur l'écran. Cliquez sur Suivant pour commencer.
- Sélectionnez le réseau Wi-Fi et entrez le mot de passe de votre routeur Wi-Fi.
- Cliquez sur l'icône Wi-Fi en haut à droite de l'écran et notez le fil- moins l'adresse IP de votre Raspberry Pi (notez que cette adresse IP n'est pas statique et qu'elle peut changer la prochaine fois que vous allumerez votre Raspberry Pi).
- Vous devriez maintenant être prêt à utiliser votre Raspberry Pi 4 (voir Bureau dans la figure 2.4).

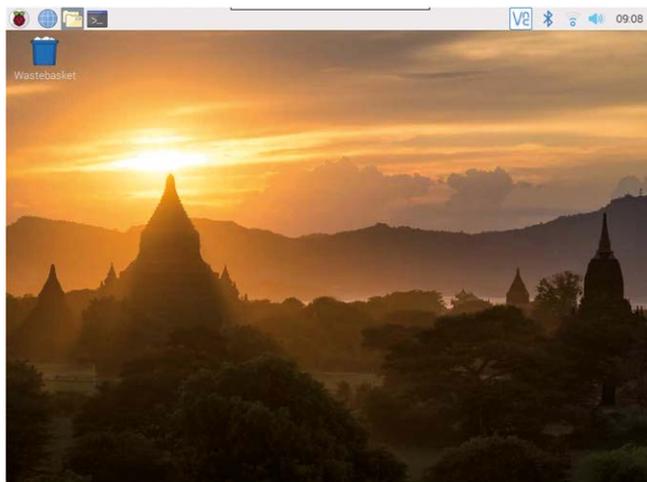


Illustration 2.4 Bureau Raspberry Pi 4

Notez que l'adresse IP de votre Raspberry Pi peut également être vue dans votre routeur. Vous pouvez également obtenir l'adresse IP de votre Raspberry Pi à partir de votre téléphone mobile. Il existe plusieurs programmes gratuits qui peuvent être installés sur votre téléphone mobile et qui vous montreront les adresses IP de tous les appareils connectés à votre routeur. Dans cette section, l'utilisation de l'Android **applications** appelé **Who's On My Wi-Fi – Network Scanner** par Magdalm est utilisé pour montrer comment l'adresse IP de votre Raspberry Pi peut être affichée. En exécutant ce programme, l'adresse IP sans fil de Raspberry Pi s'affichera sous le titre **Raspberry Pi Trading Ltd**. En outre- Ce programme affiche tous les paramètres tels que l'adresse MAC, l'adresse de la passerelle, le masque IP, etc.

2.3 accès à distance

Il est beaucoup plus facile d'accéder au Raspberry Pi à distance via Internet, par exemple en utilisant un PC plutôt que de connecter un clavier, une souris et un écran. Avant de pouvoir agir- Pour connecter votre Raspberry Pi à distance, nous devons activer le SSH et le VNC en saisissant la commande suivante dans une session de terminal :

```
pi@raspberrypi :~$ sudo raspi-config
```

Allez dans le menu de configuration et sélectionnez **Options d'interface**. Descendez à **P2 SSH** (voir la figure 2.5) et activez SSH. Cliquez sur **<Finish>** pour quitter le menu.

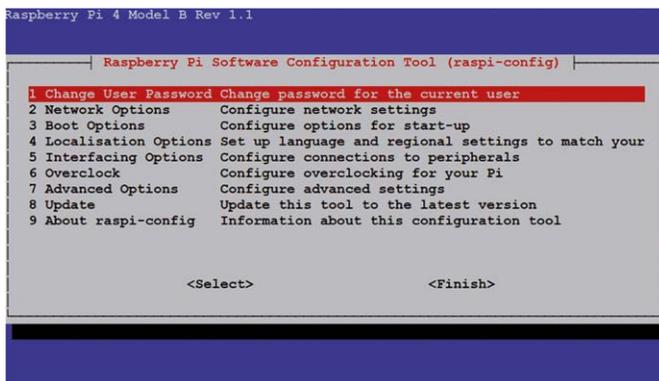


Figure 2.5 Activation de SSH

Vous devez également activer VNC afin que votre Raspberry Pi puisse être accessible graphiquement sur Internet. Cela peut être fait en entrant la commande suivante dans une session de terminal :

```
pi@raspberrypi :~$ sudo raspi-config
```

Allez dans le menu de configuration et sélectionnez **Options d'interface**. Descendez à **P3 VNC** et enable VNC. Cliquez sur **<Finish>** pour quitter le menu. À ce stade, vous pouvez arrêter votre Raspberry Pi en cliquant sur **Menu des applications** sur le bureau et en sélectionnant **Arrêt** option.

2.4 Utilisation de la pête

Putty est un programme de communication qui permet de créer une connexion entre votre PC et le Raspberry Pi. Cette connexion utilise un protocole sécurisé appelé SSH (Secure Shell). Putty n'a pas besoin d'être installé car il peut simplement être stocké dans un dossier de votre choix et exécuté à partir de là.

Putty peut être téléchargé à partir du site web suivant :

<https://www.putty.org/>

Double-cliquez simplement pour l'exécuter et l'écran de démarrage de Putty sera affiché. Cliquez **SSH** et entrez l'adresse IP du Raspberry Pi, puis cliquez sur **Ouvrir** (voir la figure 2.6). Le message de la figure 2.7 s'affiche lorsque vous accédez pour la première fois à votre Raspberry Pi. Cliquez sur **Oui** d'accepter cette alerte de sécurité.

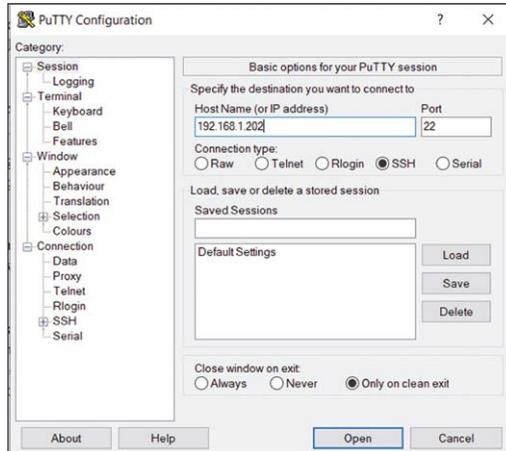


Figure 2.6 Écran de démarrage de Putty



Figure 2.7 Cliquez sur Oui pour accepter

Vous serez invité à saisir le nom d'utilisateur et le mot de passe. Notez que l'utilisateur par défaut- Le nom et le mot de passe sont :

nom d'utilisateur : **pi**
 mot de passe : **framboise**

Vous avez maintenant une connexion au terminal avec le Raspberry Pi et vous pouvez taper des commandes, y compris les commandes sudo. Vous pouvez utiliser les touches de curseur pour faire défiler vers le haut ou vers le bas les commandes que vous avez précédemment entrées dans la même session. Vous pouvez également exécuter des programmes- Pas de programmes graphiques.

2 . 4. 1 Configuration de Putty

Par défaut, l'arrière-plan de l'écran Putty est noir avec des caractères blancs en premier plan. Dans ce livre, nous avons utilisé un fond blanc avec des caractères noirs en premier plan, la taille du caractère étant définie sur 12 points gras. Les étapes pour configurer Putty avec ces paramètres sont données ci-dessous. Notez que dans cet exemple les paramètres sont enregistrés avec le nom **RPI4** afin de pouvoir les rappeler à chaque fois que Putty est redémarré :

- Redémarrer le mastic
- Sélectionner **SSH** et entrez l'adresse IP de la Raspberry Pi
- Cliquez **Couleurs** sous **Fenêtre**
- Définir le **Premier plan par défaut** et **Premier plan gras par défaut** couleurs au noir (Rouge:0, Vert:0, Bleu:0)
- Définir le **Arrière-plan par défaut** et **Fond gras par défaut** vers le blanc (Rouge:255, Vert:255, Bleu:255)
- Définir le **Texte du curseur** et **Couleur du curseur** au noir (Rouge:0, Vert:0, Bleu:0)
- Sélectionner **Apparence** sous **Fenêtre** et cliquez **Changement** dans **Paramètres de police**. Définir la police **Bold 12**.
- Sélectionner **Session** et donnez un nom à la session (p. ex., RPI4) et cliquez sur **Sauver**.
- Cliquez **Ouvrir** pour ouvrir la session Putty avec la configuration enregistrée
- La prochaine fois que vous redémarrez Putty, sélectionnez la session enregistrée et cliquez sur **Charger** suivie par **Ouvrir** pour démarrer une session avec la configuration enregistrée

2 . 5 Accès à distance du bureau

Vous pouvez contrôler votre Raspberry Pi via Putty, et exécuter des programmes sur celui-ci à partir de votre PC Windows. Ceci, cependant, ne fonctionnera pas avec les programmes graphiques parce que Windows ne sait pas comment représenter l'affichage. En conséquence de cela, par exemple, nous ne pouvons pas exécuter de pro graphique- grammes en mode Bureau. Nous pouvons contourner ce problème en utilisant un logiciel supplémentaire. Deux programmes populaires utilisés à cette fin sont VNC (Virtual Network Connection) et Xming. Ici, nous allons apprendre comment utiliser VNC.

Installation et utilisation de VNC

VNC se compose de deux parties : VNC Server et VNC Viewer. VNC Server fonctionne sur le Raspberry Pi, et VNC Viewer fonctionne sur le PC. Le serveur VNC est déjà installé sur votre Raspberry Pi. Vous pouvez démarrer le serveur en saisissant la commande suivante dans le mode de commande :

```
pi$raspberrypi :~ $ vncserver :1
```

Les étapes suivantes sont indiquées pour installer et utiliser VNC Viewer sur votre ordinateur :

- Il existe de nombreux visionneuses VNC, mais la plus recommandée est TightVNC qui peut être téléchargée à partir du site web suivant :

<https://www.tightvnc.com/download.php>

- Télécharger et installer **TightVNC** logiciel pour votre PC. Vous devrez choisir un mot de passe lors de l'installation.
- Démarrer **Visionneuse TightVNC** sur votre PC et saisissez l'adresse IP du Raspberry Pi (voir la figure 2.8) suivie de :1. Cliquez **Connecter** pour vous connecter à votre Raspberry Pi.

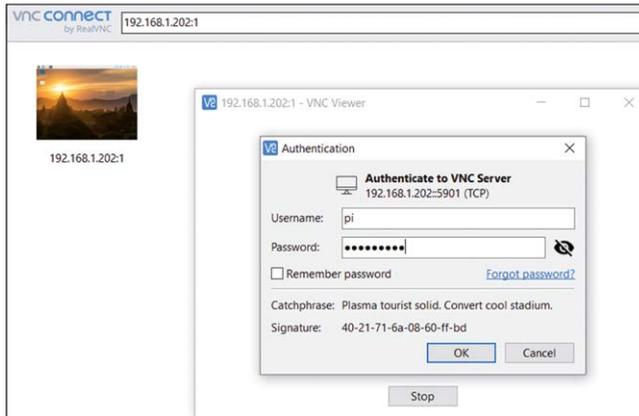


Figure 2.8 Démarrez le TightVNC et entrez l'adresse IP

La figure 2.9 montre le bureau du Raspberry Pi affiché sur l'écran du PC.

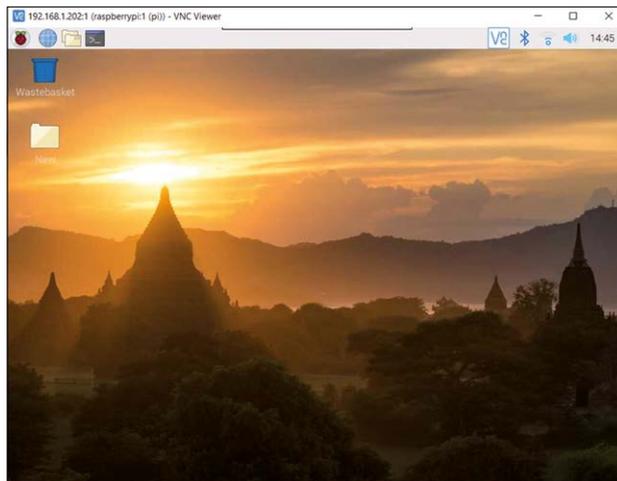


Figure 2.9 Bureau Raspberry Pi sur l'écran du PC]

2 . 6 Résumé

Dans ce chapitre, nous avons appris comment installer le dernier système d'exploitation Raspberry Pi sur une carte SD. Maintenant que le logiciel est installé et que notre Raspberry Pi fonctionne, dans le prochain chapitre, nous allons examiner certaines des commandes importantes du système d'exploitation Raspberry Pi.

Chapitre 3 • Utilisation de la ligne de commande

3.1 Aperçu

Le système d'exploitation Raspberry Pi est basé sur une version du système d'exploitation Linux. Linux est l'un des systèmes d'exploitation les plus populaires utilisés aujourd'hui. Linux est très similaire aux autres systèmes d'exploitation, tels que Windows et Unix. Linux est un système d'exploitation ouvert basé sur Unix et développé de manière collaborative par de nombreuses entreprises et universités depuis 1991. En général, Linux est plus difficile à gérer que d'autres systèmes d'exploitation comme Windows. Le système offre plus de flexibilité et des options de configuration plus larges. Il existe plusieurs versions populaires du système d'exploitation Linux telles que Debian, Ubuntu, Red Hat, Fedora et ainsi de suite. Dans ce chapitre, nous allons examiner quelques-unes des commandes couramment utilisées par le Raspberry Pi que nous pouvons entrer à partir de la ligne de commande. Les commandes saisies par l'utilisateur sont affichées en gras pour plus de clarté.

3.2 La structure du répertoire Raspberry Pi

La structure de répertoire du Raspberry Pi est constituée d'un seul répertoire racine, avec des répertoires et sous-répertoires sous la racine. Les différents programmes, applications et données utilisateur du système d'exploitation sont stockés dans des répertoires et sous-répertoires différents.

Le répertoire racine est identifié par la "/" symbole. Sous la racine, nous avons des répertoires nommés tels que **bin**, **démarrer**, **dev**, **etc.**, **à la maison**, **lib**, **lost+found**, **médias**, **mnt**, **opt**, **proc**, et bien d'autres. Le répertoire important pour les utilisateurs est le **à la maison** répertoire qui contient des sous-répertoires pour chaque utilisateur du système. Le chemin complet vers le répertoire home est /home/pi. **Nous pouvons passer à notre répertoire d'accueil depuis n'importe quel autre répertoire en entrant la commande** `cd ~`

Quelques commandes de répertoire utiles sont données ci-dessous. Commande **pwd** affiche le répertoire d'accueil de l'utilisateur :

```
pi@raspberrypi:~$ pwd
/home/pi pi@raspberrypi:~$
```

Pour afficher la structure du répertoire, entrez la commande **ls /** comme indiqué à la figure 3.1.

```
pi@raspberrypi:~$ ls /
bin  dev  home  lost+found  mnt  proc  run  srv  tmp  var
boot  etc  lib  media      opt  root  sbin  sys  usr
```

Figure 3.1 Liste de la structure des répertoires

Pour afficher les sous-répertoires et les fichiers de notre répertoire personnel, saisissez **ls** comme indiqué à la figure 3.2.

```
pi@raspberrypi:~$ ls
Desktop  Downloads  Music      Public      Videos
Documents  MagPi     Pictures  Templates
```

Figure 3.2 Liste des fichiers dans notre répertoire personnel

La commande `ls` peut prendre un certain nombre d'arguments. Quelques exemples sont donnés ci-dessous : Pour afficher les sous-répertoires et les fichiers dans une seule ligne :

```
pi@framberrypi :~ $ ls -l
```

Pour afficher le type de fichier, saisissez la commande suivante. Notez que les répertoires ont un « / » après leur nom, les fichiers exécutables ont un caractère « * » après leur nom :

```
pi@framberrypi :~ $ ls -F
```

Pour afficher les résultats séparés par des virgules :

```
pi@framberrypi :~ $ ls -m
```

Nous pouvons mélanger les arguments comme dans l'exemple suivant :

```
pi@framberrypi :~ $ ls -m -F
```

Les sous-répertoires sont créés à l'aide de la commande `mkdir` suivi du nom de la sous-direction. Dans l'exemple suivant, le sous-répertoire **myfiles** est créé dans notre répertoire de travail (voir Figure 3.3).

```
pi@raspberrypi:~ $ mkdir myfiles
pi@raspberrypi:~ $ ls
Desktop  Downloads  Music     Pictures  Templates
Documents  MagPi     myfiles  Public    Videos
pi@raspberrypi:~ $
```

Figure 3.3 Création d'un sous-répertoire

Utiliser la commande `rmdir` suivi du nom du sous-répertoire pour supprimer un sous-répertoire.

3.3 Permissions du fichier

Un des arguments importants utilisés avec le `ls` commande est `-l` (lettre minuscule l) qui affiche les permissions des fichiers, la taille des fichiers et quand ils ont été modifiés pour la dernière fois. Dans l'exemple de la figure 3.4, chaque ligne se rapporte à un répertoire ou fichier. En lisant de droite à gauche, le nom du répertoire ou du fichier est sur la droite. La date de création du répertoire ou du fichier se trouve à gauche de son nom. Vient ensuite la taille en octets. Les caractères au début de chaque ligne concernent les autorisations. c.-à-d. qui est autorisé à utiliser ou modifier le fichier ou le sous-répertoire.

```

pi@raspberrypi:~ $ ls -l
total 36
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Desktop
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Documents
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Downloads
drwxr-xr-x 2 pi pi 4096 Jun 20 17:55 MagPi
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Music
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Pictures
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Public
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Templates
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Videos
pi@raspberrypi:~ $

```

Figure 3.4 Exemple d'autorisations de fichier

Les autorisations sont divisées en 3 catégories :

- Ce que l'utilisateur (ou le propriétaire, ou le créateur) peut faire – appelé UTILISATEUR
- Ce que le propriétaire du groupe (les personnes du même groupe) peut faire – appelé GROUPE
- Ce que tout le monde peut faire – appelé MONDE

La première **pi** dans l'exemple montre qui est l'utilisateur du fichier (ou sous-répertoire) et le second supplémentaire **pi** indique le nom du groupe qui possède le fichier (ou sous-répertoire). Dans cet exemple, les noms de l'utilisateur et du groupe sont **pi**.

Les permissions peuvent être analysées en décomposant les caractères en quatre parties pour le type de fichier, l'utilisateur, le groupe et le monde. Le premier caractère d'un fichier est "-", et pour un répertoire, il est "d". Vient ensuite les permissions pour l'utilisateur, le groupe et le monde. Les autorisations sont les suivantes :

- **Permission de lecture (r)** : l'autorisation d'ouvrir et de lire un fichier ou de lister un répertoire
- **Permission d'écriture (w)** : l'autorisation de modifier un fichier, ou de supprimer ou créer un fichier dans un répertoire
- **Exécuter l'autorisation (x)** : l'autorisation d'exécuter le fichier (s'applique à l'exécutables-fichiers possibles), ou pour entrer dans un répertoire

Les trois lettres **rwX** sont utilisés comme groupe et si aucune autorisation n'est attribuée, un caractère "-" est utilisé.

A titre d'exemple, considérant la **Musique** sous-répertoire, nous avons les codes d'autorisation suivants :

`drwxr-xr-x`, qui se traduit par :

- `d` : c'est un répertoire
- `rwX` : l'utilisateur (propriétaire) peut lire, écrire et exécuter
- `r-x` : groupe peut lire et exécuter, mais ne peut pas écrire (p. ex., créer ou supprimer)
- `r-x` : monde (tout le monde) peut lire et exécuter, mais ne peut pas écrire

Le **chmod** est utilisée pour modifier les permissions des fichiers. Avant d'entrer dans les détails de la façon de modifier les autorisations, regardons et voyons quels arguments sont disponibles dans **chmod** pour modifier les autorisations des fichiers.

Les arguments disponibles pour changer les permissions des fichiers sont donnés ci-dessous. Nous pouvons utiliser ces arguments pour ajouter/supprimer des permissions ou pour définir explicitement des permissions. Il est important de réaliser que si nous définissons explicitement les autorisations, alors toutes les autorisations non spécifiées dans le com- La demande sera révoquée :

```

u :      groupe d'utilisateurs
g :      (ou de propriétaires)
o :      autres (monde)
a :      tous

+ :      ajouter
- :      retirer
= :      set

r :      lire
w :      écrire
x :      exécuter
    
```

Pour modifier les permissions d'un fichier, on tape le **chmod** commande, suivie de l'une des lettres **u**, **g**, **o**, ou **a** sélectionner les personnes, puis les **+** ou **=** pour sélectionner le type de modification, et enfin suivi du nom du fichier. Un exemple est donné ci-dessous. Dans cet exemple, sous-répertoire Musique l'utilisateur a les autorisations de lecture et d'écriture. Nous allons modifier le- missions de sorte que l'utilisateur n'ait pas de permission de lecture sur ce fichier :

```

pi@framberrypi ~$ chmod -u -r Musique
pi@framberrypi ~$ ls -l
    
```

Le résultat est illustré à la figure 3.5.

```

pi@raspberrypi:~ $ chmod -u -r Music
pi@raspberrypi:~ $ ls -l
total 36
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Desktop
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Documents
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Downloads
drwxr-xr-x 2 pi pi 4096 Jun 20 17:55 MagPi
d----- 2 pi pi 4096 Jun 20 18:20 Music
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Pictures
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Public
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Templates
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Videos
pi@raspberrypi:~ $
    
```

Figure 3.5 Autorisations de fichier du sous-répertoire Music

Dans l'exemple suivant, **rwX** les autorisations utilisateur sont données au sous-répertoire **Musique** :

```
pi@framberrypi ~$ chmod u+rwX Musique
```

La figure 3.6 montre les nouvelles autorisations du sous-répertoire Music.

```
pi@raspberrypi:~ $ chmod u+rwX Music
pi@raspberrypi:~ $ ls -l
total 36
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Desktop
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Documents
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Downloads
drwxr-xr-x 2 pi pi 4096 Jun 20 17:55 MagPi
drwx----- 2 pi pi 4096 Jun 20 18:20 Music
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Pictures
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Public
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Templates
drwxr-xr-x 2 pi pi 4096 Jun 20 18:20 Videos
pi@raspberrypi:~ $
```

Figure 3.6 Nouvelles autorisations du sous-répertoire Music

Pour changer notre répertoire de travail la commande **cd** est utilisé. Dans l'exemple suivant, nous changeons notre répertoire de travail en **Musique** :

```
pi@framberrypi ~$ cd /home/pi/Musique
pi@raspberrypi ~/Music $
```

Monter d'un niveau de répertoire, c.-à-d. dans notre répertoire de travail par défaut :

```
pi@raspberrypi ~/Music $ cd.
```

```
pi@framberrypi ~$
```

modifier notre répertoire de travail pour **Musique**, on peut aussi entrer la commande :

```
pi@framberrypi ~$ cd ~/Music
pi@raspberrypi ~/myfiles $
```

pour retourner dans le répertoire de travail par défaut, on peut entrer :

```
pi@raspberrypi ~/Music $ cd ~
pi@framberrypi ~$
```

pour obtenir plus d'informations sur un fichier, on peut utiliser la commande **file**. Par exemple :

```
pi@framberrypi ~$ fichier Musique
Musique : répertoire pi@raspberrypi
~$
```

la **-R** argument de commande **ls** liste tous les fichiers de tous les sous-répertoires du répertoire de travail actuel. Un exemple est montré dans la Figure 3.7.

```
pi@raspberrypi:~ $ ls -R MagPi
MagPi:
MagPi82.pdf
pi@raspberrypi:~ $
```

Figure 3.7 Utilisation de la commande **-R** avec **ls**

3.4 Aide

Homme

Pour afficher des informations sur l'utilisation d'une commande, nous pouvons utiliser la commande **man**. Par exemple, pour obtenir de l'aide sur l'utilisation de la **mkdir** commande :

```
pi@framberrypi ~$ man mkdir
MKDIR(1)

NOM
  Mkdir – créer des répertoires

Synopsis
  Mkir [OPTION]... RÉPERTOIRE...

DESCRIPTION
  Créer le(s) RÉPERTOIRE, s'ils n'existent pas déjà.

  Les arguments obligatoires pour les options longues sont obligatoires pour les options courtes

  -m, --mode=MODE
    Définir le mode fichier (comme dans chmod), pas a=rwx – umask
-----
-----
```

Entrer **Ctrl+Z** pour quitter l'affichage de l'homme.

Aide

Le **homme** donne généralement plusieurs pages d'informations sur la façon d'utiliser une commande. Nous pouvons taper **q** pour sortir du **homme** et retourner à l'invite du système d'exploitation.

La commande **less** permet d'afficher une longue liste page par page. À l'aide des touches de direction vers le haut et vers le bas, nous pouvons passer d'une page à l'autre. Un exemple est donné ci-dessous. Type **q** pour sortir :

```
pi@framberrypi ~$ hommes | moins
<display of help on using the ls command>
```

```
pi@framberrypi ~$
```

3 ° 5 Date, heure et calendrier

Pour afficher la date et l'heure actuelles, le **date** est utilisée. De même, la commande **cal** com- Affiche le calendrier actuel. La figure 3.8 en est un exemple.

```
pi@raspberrypi:~ $ date
Wed 10 Jul 2019 11:53:55 AM BST
pi@raspberrypi:~ $ cal
      July 2019
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
  7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

pi@raspberrypi:~ $
```

Figure 3.8 Commandes : **date** et **cal**

3 . 6 Opérations de fichier

Copier un fichier

Pour faire une copie d'un fichier, utilisez la commande **cp**. Dans l'exemple suivant, une copie du fichier **mytestfile . txt** est créé et le nouveau fichier est nommé **test . txt** :

```
pi@framberrypi ~$ cp mytestfile.txt test.txt
```

Wildcards

Nous pouvons utiliser des caractères génériques pour sélectionner plusieurs fichiers ayant des caractéristiques similaires. p. ex., les fichiers ayant le même nom d'extension de fichier. La "*" est utilisé pour correspondre à un nombre quelconque de caractères. De même, le "?" est utilisé pour correspondre à un seul caractère. Dans l'examen- ple ci-dessous tous les fichiers avec extensions " . txt" sont énumérés :

```
pi@framberrypi ~$ ls *.txt
```

Les caractères génériques [a-z] peuvent être utilisés pour correspondre à n'importe quel caractère dans la plage de caractères spécifiée. Un exemple est donné ci-dessous qui correspond à tous les fichiers qui commencent par des lettres **o**, **p**, **q**, **r**, **s**, et **t**, et avec le " . txt" extension :

```
pi@framberrypi ~$ ls [o-t]*.txt
```

Renommer un fichier

Vous pouvez renommer un fichier en utilisant la **mv** . Dans l'exemple ci-dessous, le nom du fichier **test . txt** est remplacé par **test2 . txt** :

```
pi@framberrypi ~$ mv test.txt test2.txt
```

Suppression d'un fichier

La commande **rm** peut être utilisé pour supprimer (supprimer) un fichier. Dans l'exemple ci-dessous **test2.txt** est supprimée :

```
pi@framberrypi ~$ rm test2.txt
```

L'argument **-v** peut être utilisé pour afficher un message lorsqu'un fichier est supprimé. De plus, le **-i** demande confirmation avant qu'un fichier soit supprimé. En général, les deux arguments sont utilisés ensemble comme **-vi**. **Un exemple est donné ci-dessous :**

```
pi@framberrypi ~$ rm -vi test2.txt
rm : supprimer le fichier régulier 'test2.txt'? y
supprimé 'test2.txt'
pi@framberrypi ~$
```

Suppression d'un répertoire

Un répertoire peut être supprimé en utilisant le **rmdir** commande :

```
pi@framberrypi ~$ rmdir Music
```

Réorientation de la sortie

Le plus grand signe **>** peut être utilisé pour rediriger la sortie d'une commande vers un fichier. Par exemple, nous pouvons rediriger la sortie de la **ls** commande vers un fichier appelé **lstest.txt** :

```
pi@framberrypi ~$ ls > lstest.txt
```

Le **cat** permet d'afficher le contenu d'un fichier :

```
pi@framberrypi ~$ cat mytestfile.txt
Il s'agit d'un
fichier
Ceci est la ligne 2
pi@framberrypi ~$
```

Utiliser deux signes majeurs **>>** ajoute à la fin d'un fichier.

Écriture sur l'écran ou dans un fichier

Le **echo** peut être utilisé pour écrire sur l'écran. Il peut être utilisé pour effectuer des opérations mathématiques simples si les nombres et l'opération sont entourés de deux crochets, précédés d'un caractère **\$** :

```
pi@framberrypi ~$ echo $((5*6))
30
pi@framberrypi ~$
```

La commande `echo` peut également être utilisée pour écrire une ligne de texte dans un fichier. Voici un exemple :

```
pi@framberrypi ~$ echo une ligne de texte > lin.dat
pi@framberrypi ~$ cat lin.dat
une ligne de texte
pi@framberrypi ~$
```

Correspondance d'une chaîne

Le **grep** peut être utilisée pour faire correspondre une chaîne dans un fichier. Un exemple est donné ci-dessous en supposant que le fichier **lin.dat** contient une piqûre **une ligne de texte**. Notez que le mot correspondant est affiché en gras :

```
pi@framberrypi ~$ greffe line lin.dat a ligne de
texte
pi@framberrypi ~$
```

Commandes de tête et de queue

La commande `head` peut être utilisée pour afficher les 10 premières lignes d'un fichier. Le format de cette commande est le suivant :

```
pi@framberrypi ~$ head mytestfile.txt
.....
.....
pi@framberrypi ~$
```

De même, la commande `tail` est utilisée pour afficher les 10 dernières lignes d'un fichier. Le format de cette commande est le suivant :

```
pi@framberrypi ~$ tail mytestfile.txt
.....
.....
pi@framberrypi ~$
```

3 . 7 Informations système et utilisateur

Ces commandes sont utiles car elles nous donnent des informations sur le système. Commande **cat /proc/cpuinfo** affiche des informations sur le processeur (commande **chat** affiche le con- d'un fichier. Dans cet exemple, le contenu du fichier **/proc/cpuinfo** est affiché). Comme il y a 4 cœurs dans le Raspberry Pi 4, l'affichage est en 4 sections. La figure 3.9 montre un exemple d'affichage, où seule une partie de l'affichage est ici représentée.

```

pi@raspberrypi:~ $ cat /proc/cpuinfo
processor       : 0
model name     : ARMv7 Processor rev 3 (v7l)
BogoMIPS      : 108.00
Features       : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xd08
CPU revision   : 3

processor       : 1
model name     : ARMv7 Processor rev 3 (v7l)
BogoMIPS      : 108.00
Features       : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xd08
CPU revision   : 3

```

Figure 3.9 Commande : `cat /proc/cpuinfo` (seule une partie de la sortie est affichée)

Commandement **uname -s** affiche le nom du noyau du système d'exploitation, qui est Linux. Commande **uname -a** affiche des informations détaillées complètes sur le noyau et l'opérat- système de gestion.

Commandement **cat /proc/meminfo** affiche des informations sur la mémoire de votre Pi. Infor- Les informations telles que la mémoire totale et la mémoire libre au moment de l'émission de la commande sont affichées.

Commandement **whoami** affiche le nom de l'utilisateur actuel. Dans ce cas, il s'agit **pi**.

Un nouvel utilisateur peut être ajouté à votre Raspberry Pi en utilisant la commande **ajouter un utilisateur**. Dans l'ex- Dans la figure 3.10, un utilisateur nommé Johnson est ajouté. Un mot de passe pour le nouvel utilisateur peut être ajouté en utilisant **passwd** suivi du nom d'utilisateur. Dans la figure 3.10, le passage- mot pour l'utilisateur Johnson est réglé à **mon mot de passe** (non affiché pour des raisons de sécurité). Notez que les deux **ajouter un utilisateur** et **passwd** sont des commandes privilégiées et le mot-clé **sudo** doit être saisi avant ces commandes. Notez que la **-m** option crée un répertoire personnel pour le nouvel utilisateur.

```

pi@raspberrypi:~ $ sudo useradd -m Johnson
pi@raspberrypi:~ $ sudo passwd Johnson
New password:
Retype new password:
passwd: password updated successfully
pi@raspberrypi:~ $

```

Figure 3.10 Commandes : `useradd` et `passwd`

Nous pouvons nous connecter au nouveau compte utilisateur en spécifiant le nom d'utilisateur et le mot de passe.

Quel logiciel est installé sur mon Raspberry Pi

Pour savoir quel logiciel est installé sur votre Raspberry Pi, entrez la commande suivante. Vous devriez obtenir plusieurs pages d'affichage :

```
pi@framberrypi ~$ dpkg -l
.....
.....
pi@framberrypi ~$
```

Nous pouvons également savoir si un certain logiciel est déjà installé sur notre ordinateur. Un exemple est donné ci-dessous qui vérifie si oui ou non le logiciel appelé **xpdf** (Lecteur PDF) est installé. Dans cet exemple **xpdf** est installé et les détails de ce logiciel s'affichent :

```
pi@framberrypi ~$ dpkg --s xpdf Forfait :
xpdf
Statut : install ok installed Priorité :
facultatif
Section : texte
Taille installée : 395
.....
.....
pi@framberrypi ~$
```

Si le logiciel n'est pas installé, nous recevons un message similaire à celui ci-dessous (en supposant que nous vérifions si un paquet logiciel appelé **bbgd** est installé) :

```
pi@framberrypi ~$ dpkg --s bbgd
dpkg-query : le paquet 'bbgd' n'est pas installé et aucune information n'est disponible
.....
.....
pi@framberrypi ~$
```

Commandes de l'utilisateur principal

Certaines commandes sont privilégiées et seules les personnes autorisées peuvent les utiliser. Insertion du mot **sudo** au début d'une commande nous donne l'autorité d'utiliser la commande sans avoir à se connecter en tant qu'utilisateur autorisé.

3 . 8 Surveillance des ressources sur Raspberry Pi

La surveillance du système est un sujet important pour gérer l'utilisation de votre Raspberry Pi. L'une des commandes de surveillance du système les plus utiles est la commande **top**, qui affiche l'utilisation actuelle des ressources du système et indique quels processus sont en cours d'exécution et combien de mémoire et de temps CPU ils consomment.

La figure 3.11 montre un affichage typique des ressources système obtenu en saisissant la commande suivante (Entrée **Ctrl+Z** pour sortir) :

```
pi@framberrypi ~$ top
pi@framberrypi ~$
```

```

pi@raspberrypi: ~
top - 13:04:21 up 5:17, 3 users, load average: 0.00, 0.00, 0.00
Tasks: 149 total, 1 running, 146 sleeping, 0 stopped, 2 zombie
%Cpu(s): 0.0 us, 5.2 sy, 0.0 ni, 94.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1939.5 total, 1353.7 free, 151.3 used, 434.5 buff/cache
MiB Swap: 100.0 total, 100.0 free, 0.0 used, 1669.8 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM     TIME+  COMMAND
 3419 pi          20   0 10320 2876 2512 R  11.8   0.1   0:00.06 top
    1 root        20   0 33724 7904 6288 S   0.0   0.4   0:03.78 systemd
    2 root        20   0    0    0    0 S   0.0   0.0   0:00.01 kthreadd
    3 root        0 -20    0    0    0 I   0.0   0.0   0:00.00 rcu_gp
    4 root        0 -20    0    0    0 I   0.0   0.0   0:00.00 rcu_par_gp
    8 root        0 -20    0    0    0 I   0.0   0.0   0:00.00 mm_percpu+
    9 root        20   0    0    0    0 S   0.0   0.0   0:00.06 ksoftirqd+
   10 root        20   0    0    0    0 I   0.0   0.0   0:00.93 rcu_sched
   11 root        20   0    0    0    0 I   0.0   0.0   0:00.00 rcu_bh
   12 root        xt   0    0    0    0 S   0.0   0.0   0:00.02 migration+
   13 root        20   0    0    0    0 S   0.0   0.0   0:00.00 cpuhp/0
   14 root        20   0    0    0    0 S   0.0   0.0   0:00.00 cpuhp/1
   15 root        xt   0    0    0    0 S   0.0   0.0   0:00.01 migration+
   16 root        20   0    0    0    0 S   0.0   0.0   0:00.04 ksoftirqd+
   19 root        20   0    0    0    0 S   0.0   0.0   0:00.00 cpuhp/2
   20 root        xt   0    0    0    0 S   0.0   0.0   0:00.01 migration+
   21 root        20   0    0    0    0 S   0.0   0.0   0:00.10 ksoftirqd+

```

Figure 3.11 Affichage typique des ressources du système

Certains des points importants de la figure 3.11 sont résumés ci-dessous (pour les lignes 1 à 5 de l’affichage) :

- Il y a un total de 149 processus dans le système
- Actuellement, un seul processus est en cours d’exécution, 146 processus sont en veille et 2 processus sont à l’état zombie
- Le pourcentage d’utilisation du processeur est de 0,0us pour les applications utilisateur (us)
- Le pourcentage d’utilisation du processeur pour les applications système est de 5,2 (sy)
- Il n’y a pas de processus exigeant plus ou moins de priorité (ni)
- 94,8% du temps où le CPU est inactif (id)
- Il n’y a pas de processus en attente d’achèvement des E/S (wa)
- Il n’y a pas de processus qui attendent les interruptions matérielles (salut)
- Il n’y a pas de processus qui attendent les interruptions logicielles (si)
- Il n’y a pas de temps réservé pour un hyperviseur (st)
- La mémoire utilisable totale est de 1939.5MB, dont 151.3MB d’octets sont utilisés, 1353.7MB sont libres et 434.5MB sont utilisés par les buffers/cache
- La ligne 5 affiche l’utilisation de l’espace d’échange

Le tableau des processus donne les informations suivantes pour tous les processus chargés dans le système :

- PID : le numéro d’identification du processus
- UTILISATEUR : le propriétaire du processus
- PR : priorité du processus
- NI : la valeur du processus
- VIRT : la quantité de mémoire virtuelle utilisée par le processus
- RES : la taille de la mémoire résidente
- SHR : mémoire partagée utilisée par le processus
- S : état du processus (sommeil, course, zombie)
- %CPU : le pourcentage de CPU consommé
- %MEM : pourcentage de la mémoire vive utilisée

- TIME+ : temps total du processeur de la tâche utilisée
- COMMANDE : Le nom réel de la commande

Le **ps** peut être utilisée pour lister tous les processus utilisés par l'utilisateur actuel. Un exemple est présenté à la figure 3.12.

```
pi@raspberrypi:~ $ ps
  PID TTY          TIME CMD
 1290 pts/0    00:00:00 bash
 2070 pts/0    00:00:01 Xtightvnc
 2074 pts/0    00:00:00 xstartup
 2077 pts/0    00:00:00 lxsession
 2120 pts/0    00:00:00 openbox
 2124 pts/0    00:00:08 lxpanel
 2126 pts/0    00:00:00 pcmanfm
 2171 pts/0    00:00:00 sh <defunct>
 3419 pts/0    00:00:03 top
 3457 pts/0    00:00:00 ps
pi@raspberrypi:~ $
```

Figure 3.12 Commande : **ps**

la commande **ps -ef** donne beaucoup plus d'informations sur les processus en cours dans le système- tem.

Tuer un processus

Il y a plusieurs options pour tuer (ou arrêter) un processus. Un processus peut être tué par spec- En saisissant son PID et en utilisant la commande suivante :

```
pi@framberrypi ~$ tuer -9 <PID>
```

Utilisation du disque

La commande libre de disque **df** peut être utilisé pour afficher les statistiques d'utilisation du disque. Un exemple est montré dans la figure 3.13. option **-h** Affiche sous une forme lisible par l'homme.

```
pi@raspberrypi:~ $ df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/root        28683772 4397836  22805824 17% /
devtmpfs         860920      0      860920  0% /dev
tmpfs            993016      0      993016  0% /dev/shm
tmpfs            993016      8684   984332  1% /run
tmpfs             5120        4        5116  1% /run/lock
tmpfs            993016      0      993016  0% /sys/fs/cgroup
/dev/mmcblk0p6  258094      39984   218111 16% /boot
tmpfs            198600      0      198600  0% /run/user/1000
```

Figure 3.13 Commande : **df**

3.9 Fermeture

Bien que vous puissiez débrancher l'alimentation de votre Raspberry Pi lorsque vous avez terminé de travailler avec elle, il n'est pas recommandé car il y a beaucoup de processus en cours d'exécution sur le système- Cela peut corrompre le système de fichiers. C'est beaucoup mieux d'arrêter le système de manière ordonnée.

La commande suivante arrête tous les processus et sécurise le système de fichiers, puis éteint le système en toute sécurité :

```
pi@framberrypi ~$ sudo halt
```

La commande suivante arrête puis redémarre le système :

```
pi@framberrypi ~$ redémarrage sudo
```

Le système peut également être arrêté puis redémarré après un certain temps en saisissant la commande suivante. Un message d'arrêt peut être affiché en option si vous le souhaitez :

```
pi@framberrypi ~$ arrêt -r <time> <message>
```

3 . 10 Résumé

Ce chapitre a décrit l'utilisation d'un certain nombre de commandes importantes sous Linux. Vous devriez être en mesure d'obtenir plus d'informations sur chaque commande et autres commandes Linux à partir d'internet et d'autres livres sur Raspberry Pi et Linux.

Dans le prochain chapitre, nous verrons comment installer le logiciel Node-RED sur notre ordinateur Raspberry Pi.

Chapitre 4 • Installation du logiciel Node-RED sur le Raspberry Pi

4 . 1 Aperçu

Dans le dernier chapitre, nous avons vu comment utiliser certaines des commandes de ligne de commande courantes de Raspberry Pi à partir d'un terminal. Dans ce chapitre, nous allons installer le logiciel Node-RED sur notre carte SD Raspberry Pi.

4 . 2 Raspberry Pi Node-RED installation

Node-RED est déjà installé sur le système d'exploitation Raspbian Buster de Raspberry Pi (vous verrez qu'il peut aussi être installé si vous utilisez une version antérieure du système d'exploitation, par ex. Jessie-Version 2017-04-010, version 0.15.3 ou plus récente, et il peut être démarré immédiatement).

Si Node-RED n'est pas installé sur votre Raspberry Pi, vous pouvez facilement l'installer en saisissant la commande suivante :

```
pi@framberrypi :~ $ sudo apt-get install nodered
```

Au moment de la rédaction de ce livre, la version par défaut de Node-RED pré-installée avec le système d'exploitation Raspbian Buster était v0.20.6 et ce n'était pas la dernière version. Si le Node-RED est déjà installé sur votre Raspberry Pi, vous pouvez mettre à niveau vers la dernière version (recommandée) en saisissant la commande suivante (voir lien : <https://nodered.org/docs/getting-started/raspberrypi>) :

```
pi@framberrypi :~ $ frapper <(curl -sL https://raw.githubusercontent.com/
node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

Après la mise à niveau de Node-RED, la dernière version au moment de la rédaction de ce livre était v1.0.3 et c'est la version utilisée dans tous les exemples de ce livre.

Vous pouvez démarrer Node-RED en saisissant la commande suivante dans le mode de commande (char- Les entrées de l'utilisateur sont en gras pour plus de clarté) :

```
pi@framberrypi :~ $ noeud-rouge-début
```

Lorsque vous démarrez, vous devriez voir des messages défiler dans votre écran, où une partie des messages est montrée dans la figure 4.1.

```
pi@raspberrypi:~ $ node-red-start

Start Node-RED

Once Node-RED has started, point a browser at http://192.168.1.202:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use  node-red-stop           to stop Node-RED
Use  node-red-start         to start Node-RED again
Use  node-red-log           to view the recent log output
Use  sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use  sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
```

Figure 4.1 Partie des messages affichés au démarrage de Node-RED

Voici une liste des commandes importantes :

noeud-rouge-début démarrer Node-RED
: **node-red-stop** : arrêter le noeud-ROUGE
node-red-log : voir la sortie du journal récent
sudo systemctl enable nodered . service : autostart Node-RED après chaque démarrage
sudo systemctl disable nodered . service : désactiver le démarrage automatique après chaque démarrage

Notez qu'il s'agit d'une application de type service et que vous pouvez y accéder à partir de n'importe quel- où sur le réseau et d'autres ordinateurs aussi. Pour ce faire, nous devons connaître l'adresse IP de notre Raspberry Pi. On peut le trouver en saisissant la commande suivante à l'invite de commandes :

```
pi@framberrypi:~ $ ifconfig
```

L'adresse IP de notre Raspberry Pi dans cet exemple est 192.168.1.202 comme indiqué sur la figure 4.2 (seule une partie de la sortie est montrée ici).

```
--
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.202  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fd83:29d0:5e0f:1:2fe0:b5bf:1692:5d3a  prefixlen 64  scopeid 0x0<gl
obal>
        inet6 fe80::7abd:1a67:fe80:e3ba  prefixlen 64  scopeid 0x20<link>
        ether dc:a6:32:00:e4:29  txqueuelen 1000  (Ethernet)
        RX packets 1135  bytes 114157 (111.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2207  bytes 2964048 (2.8 MiB)
        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

Figure 4.2 Trouver l'adresse IP de notre Raspberry Pi

Nous pouvons maintenant saisir la commande suivante dans notre navigateur web pour afficher l'écran de démarrage du Node-RED :

<http://192.168.1.202:1880>

Lorsque vous démarrez, vous devriez voir l'écran de démarrage de Node-RED comme dans la figure 4.3. Lorsque Node-RED est démarré en tant que service, appuyer sur Cntrl+C n'arrête pas le service car il continue à s'exécuter en arrière-plan.

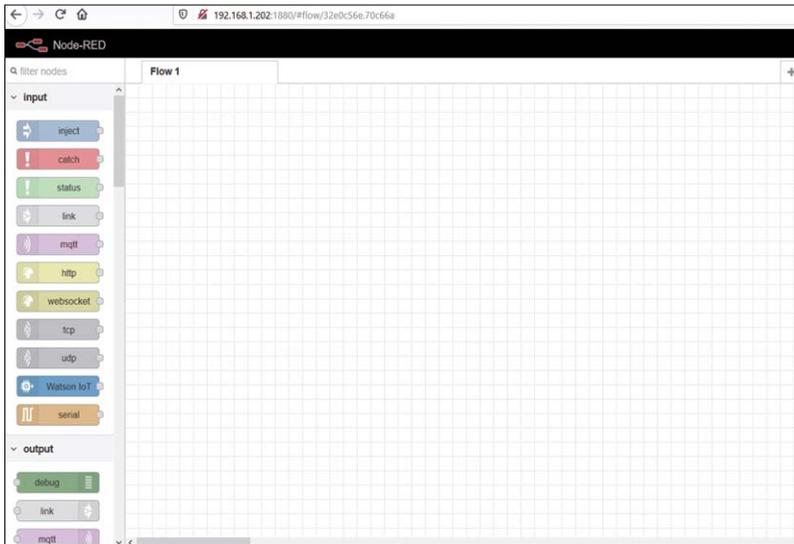


Figure 4.3 Écran de démarrage Node-RED

Notez que si vous utilisez un navigateur sur le même ordinateur que celui où est exécuté Node-RED, vous pouvez accéder à Node-RED en utilisant l'URL suivante :

<http://localhost:1880>

4.3 Interface noeud-RED vers le monde extérieur

Node-RED est normalement accessible à partir d'un navigateur web. La figure 4.4 montre le Node-RED dans- Accès au monde extérieur. Raspberry Pi accède au logiciel Node-RED via le routeur Wi-Fi local (ou Internet). Les capteurs, les actionneurs et tout autre matériel sont directement connectés aux ports GPIO du Raspberry Pi via le connecteur 40 broches monté sur le Raspberry Pi. Raspberry Pi communique avec le matériel externe en recevant des commandes de Node-RED via le routeur Wi-Fi local.

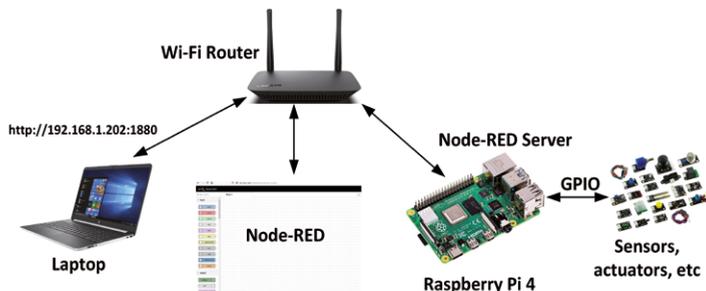


Figure 4.4 Interface noeud-RED vers le monde extérieur

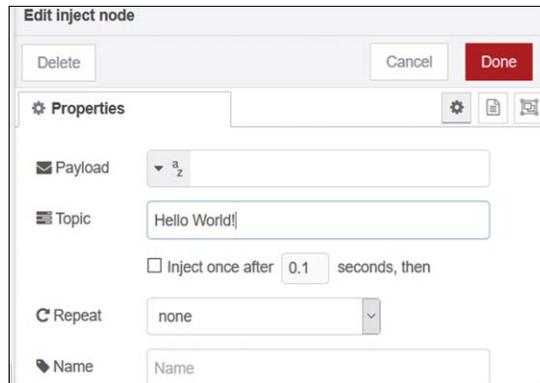


Figure 4.6 Entrez dans Hello World!

- Cliquez **Terminé**
- Ajouter le débogage de nœud en cliquant sur **déboguer** node et le faire glisser vers la droite du nœud inject déjà dans l'espace de travail
- Raccordez les deux nœuds. Placez le curseur de la souris sur **injecter** nœuds- Mettre le port (un petit carré gris sur la droite du nœud), puis cliquez avec le bouton gauche de la souris et faites glisser un fil vers le port d'entrée du nœud de débogage. Un fil doit maintenant relier les deux nœuds comme indiqué dans la figure 4.7



Figure 4.7 Raccordement des deux nœuds

- Cliquez sur le **Déployer** bouton en haut à droite de l'écran
- Cliquez sur le petit carré gris à droite du nœud **déboguer** pour activer le nœud. Vous devriez voir le message **Activé avec succès : debug** affiché par la fenêtre de la console
- Sélectionnez l'onglet de débogage en haut à droite.
- Cliquez sur le **injecter** bouton de nœuds, qui est le petit carré sortant du côté gauche du nœud. Cliquer sur le bouton injectera le message dans le flux et dans le nœud de débogage. La sortie doit être affichée sur la fenêtre de débogage comme indiqué dans la figure 4.8

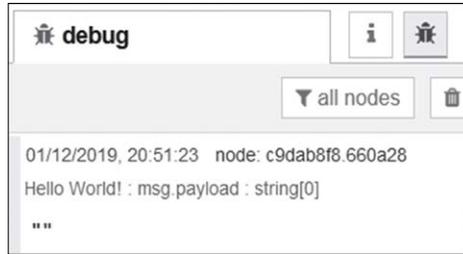


Figure 4.8 Affichage de la sortie à la fenêtre de débogage

Notez que si vous placez un mauvais nœud sur l'espace de travail, vous pouvez cliquer sur le nœud et utiliser le bouton supprimer du clavier pour le supprimer.

Dans cet exemple, nous avons d'abord créé un nœud inject. Le nœud inject peut être utilisé pour déclencher manuellement un flux en cliquant sur le bouton nodes dans l'espace de travail. Il peut également être utilisé pour- déclencher automatiquement des flux à intervalles réguliers. Nous avons ensuite **Bonjour tout le monde!** Dans ce nœud. Ensuite, nous avons créé un nœud de débogage et connecté les deux nœuds ensemble et activé le nœud de débogage. En déclenchant le nœud inject, la chaîne est envoyée au nœud de débogage et s'affiche dans la fenêtre de débogage. Le noeud Debug peut être utilisé pour afficher les messages dans la barre latérale Debug de l'éditeur. En plus des messages affichés, la barre latérale de débogage inclut des informations sur le moment où le message a été reçu et aussi l'ID du noeud de débogage qui a envoyé le message. Cliquer sur l'ID révèle le nœud qui a envoyé le message.

Programme modifié

Nous pouvons modifier le programme flow, par exemple pour afficher la chaîne toutes les 5 secondes. Les étapes à suivre sont indiquées ci-dessous :

- Double-cliquez sur **injecter** noeud
- Définir l'intervalle de répétition sur toutes les 5 secondes (figure 4.9) et cliquer **Terminé**

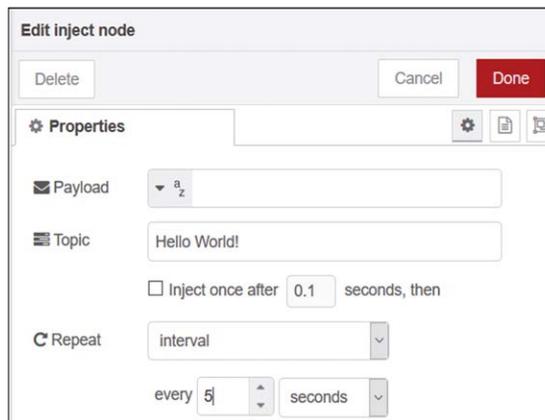


Figure 4.9 Réglez l'intervalle de répétition à 5 secondes

- Cliquez **Déployer**
- Activer la **déboguer** noeud.
- Cliquez sur le **injecter** nodes. Vous devriez voir le message affiché toutes les 5 secondes comme dans la figure 4.10

```

""
01/12/2019, 21:26:01 node: c9dab8f8.660a28
Hello World! : msg.payload : string[0]
""
01/12/2019, 21:26:06 node: c9dab8f8.660a28
Hello World! : msg.payload : string[0]
""
01/12/2019, 21:26:11 node: c9dab8f8.660a28
Hello World! : msg.payload : string[0]
""
01/12/2019, 21:26:16 node: c9dab8f8.660a28
Hello World! : msg.payload : string[0]
""
01/12/2019, 21:26:21 node: c9dab8f8.660a28
Hello World! : msg.payload : string[0]

```

Figure 4.10 Affichage du message toutes les 5 secondes

4 . 6 Projet 2 – Date et heure

Dans ce programme, nous utiliserons les **fonction** node pour retourner la date et l'heure actuelles à la fenêtre de débogage. Les étapes sont les suivantes :

- Dans la palette de noeuds, cliquez sur **injecter** Le nœud sur la gauche et faites-le glisser vers l'espace de travail.
- Double-cliquez sur ce nœud, cliquez sur **Charge utile**, sélectionner **Chaîne** et entrez la chaîne suivante **LA DATE ET L'HEURE SONT** :. Sélectionnez l'intervalle de répétition à 1 seconde comme indiqué sur la figure 4.11.

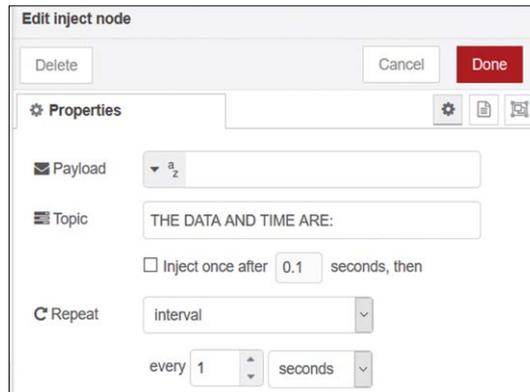


Figure 4.11 Le nœud injecteur

- Cliquez **Terminé**
- Ajouter une fonction de nœud en cliquant sur **fonction** node et le faire glisser vers la droite du nœud inject déjà dans l'espace de travail
- Raccordez les deux nœuds ensemble. Cliquez sur le nœud de fonction et saisissez ce qui suit comme indiqué dans la figure 4.12. Cliquez **Terminé** :

```
msg.payload=new Date();
renvoie msg;
```

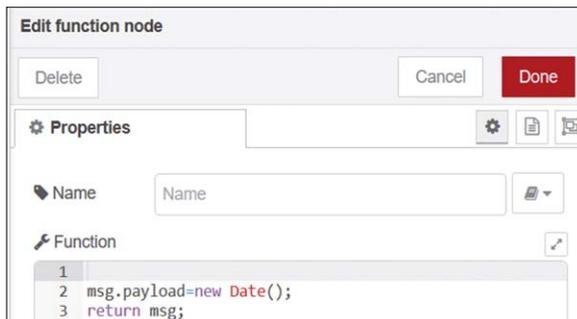


Figure 4.12 Fonction pour obtenir les données et l'heure

- Ajouter le débogage de nœud en cliquant sur **déboguer** et le faire glisser vers la droite du nœud injecté déjà dans l'espace de travail.
- Raccorder les trois nœuds ensemble comme indiqué à la figure 4.13.



Figure 4.13 Joindre les trois nœuds

- Cliquez sur le petit carré gris à droite du nœud **débugger** pour activer le nœud. Vous devriez voir le message **Activé avec succès : debug** affiché par la fenêtre de la console
- Cliquez sur le **injecter** bouton nodes. La date et l'heure devraient s'afficher toutes les secondes dans le format suivant (voir la figure 4.14) :

LA DATE ET L'HEURE SONT :

```
"Mon Dec 02 2019 15:11:03 GMT+0000
(Greenwich Mean Time)"
```



Figure 4.14 Affichage du message toutes les 5 secondes

Nous avons appris dans cet exemple que le nœud de fonction permet au code JavaScript d'être exécuté contre les messages qui sont passés à travers lui. A **fonction** nœud reçoit des données de sa gauche, traite les données et en sort de sa droite.

Un nœud de fonction peut avoir plusieurs sorties. Pour créer plus d'une sortie, double-cliquez sur le nœud de fonction et **Sorties** dans le coin inférieur gauche à la valeur requise. De plus, un nœud de fonction peut créer une bibliothèque en enregistrant vos fonctions. Cela se fait en double-cliquant sur le nœud de fonction, puis en cliquant sur **livre** icône à côté **Nom**. Ici, vous aurez des options pour **Ouvrir** une bibliothèque existante ou **Sauver** la fonction dans une bibliothèque.

Noms de flux

Nous pouvons ajouter de nouveaux flux en cliquant sur le bouton **+** signe en haut à droite de l'écran. Le nom d'un flux peut être modifié comme suit :

- Cliquez sur le menu situé en haut à droite de l'écran
- Sélectionner **Flux** -> **Renommer** comme indiqué à la figure 4.15

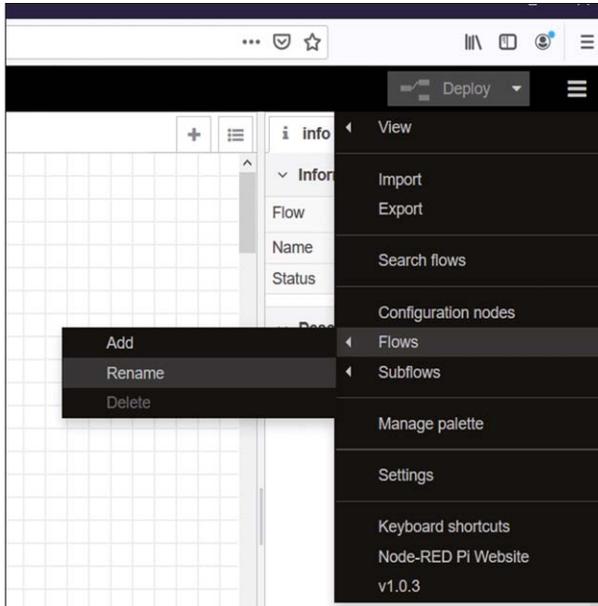


Figure 4.15 Renommer un flux

- Modifier le nom du flux existant au besoin, puis cliquer sur **Terminé**.

Programme modifié

Nous pouvons modifier la fonction dans le programme de flux ci-dessus, par exemple juste pour afficher l'heure locale dans le format suivant :

Heure = hh:mm:ss

Les étapes sont indiquées ci-dessous :

- Insérer un **injecter** node à répéter chaque seconde avec la chaîne THE CURRENT TIME
- Insérer une **fonction** node et entrez les instructions suivantes dans ce nœud de fonction et cliquez sur **Terminé** :

```
var LocalTime = new Date();  
var Tim = LocalTime.toLocaleTimeString(); msg.payload =  
msg.payload + "Time = " + Tim; return msg;
```

- Insérer une **déboguer** noeud.
- Rejoindre les trois nœuds. Cliquez sur Déployer et activez le **déboguer** noeud

- Cliquez sur le **injecter** nodes. Vous devriez voir le temps de mise à jour toutes les secondes dans la fenêtre Debug comme dans le format montré dans la figure 4.16.

```
02/12/2019, 18:55:18 node: c9dab8f8.660a28
THE CURRENT TIME : msg.payload : string[15]
"Time = 18:55:17"
```

Figure 4.16 Affichage de l'heure locale dans la fenêtre de débogage

Il est important de réaliser que la programmation en Node-RED est basée sur JavaScript qui est basé sur des objets. Les variables sont représentées par le mot-clé **var**. Par exemple, une personne peut être représentée comme suit :

```
var person = {prénom : "Dogan", nom de famille : "Ibrahim", âge : 50 ans, taille : 160};
```

les éléments de la personne peuvent être consultés, par exemple, comme suit :

```
personne.prénom ou person.nom, etc.
```

L'objet message Node-RED est appelé **msg** et il a les propriétés suivantes : payload, topic, and _msgid.

charge utile : cette propriété par défaut stocke la charge utile du message. La charge utile peut prendre les valeurs suivantes (voir lien : <https://nodered.org/docs/user-guide/messages>) :

- Booléen - vrai, faux
- Nombre - par exemple 0, 123.4
- String - "hello world"
- Tableau - [1,2,3,4,5]
- Objet - { "a" : 1, "b" : 2 }
- Nul

Par défaut, le noeud de débogage affiche la **msg . charge utile** propriété d'un message mais peut être configurée pour afficher n'importe quelle propriété ou le message entier.

Lorsque nous utilisons un noeud de fonction, nous pouvons stocker une nouvelle chaîne de texte dans la variable msg comme suit :

```
var msg = {payload : "Ceci est une chaîne de caractères"};
```

ou, nous pouvons utiliser l'énoncé suivant :

```
msg.payload = "Ceci est une chaîne de caractères";
```

sujet stocke la rubrique parent d'un message

_msgid : il s'agit d'un ID qui peut être utilisé pour identifier et suivre la progression du message à travers un flux.

4 . 7 Projet 3 – Conversion de la température

Dans ce programme, nous présenterons les **changement** nœud. Il s'agit d'un programme de conversion de température. Le programme convertit les degrés Celsius en degrés Fahrenheit. Par exemple, 10oC est converti en oF et s'affiche dans la fenêtre de débogage.

Les étapes sont les suivantes :

- Créer un **injecter** nœud avec te chaîne **Conversion de la température**
- Créer un **changement** et définissez le msg.payload à 10 (ceci est les degrés Celsi- us) comme indiqué dans la Figure 4.17. Cliquez **Terminé**

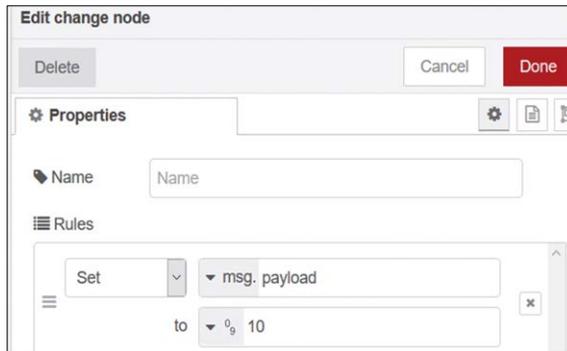


Figure 4.17 Créer un nœud de changement

- Créer un **fonction** nœud avec les instructions suivantes pour convertir oC en oF :

```
var F = msg.payload;
var C = F*1,8+32;
msg.payload = F + "C = "+ C.toString() + "F"; renvoie msg;
```

- Créer un **déboguer** et joindre les quatre nœuds ensemble comme indiqué dans la figure 4.18.



Figure 4.18 Joindre les nœuds ensemble

- Cliquez **Déployer**
- Activer la **déboguer** nœud, puis cliquez sur **injecter** bouton de noeuds. Vous devriez voir 10oC est converti en 50oF dans le noeud de débogage comme montré dans la figure 4.19

```
02/12/2019, 21:14:10 node: c9dab8f8.660a28
Temperature Conversion : msg.payload : string[9]
"10C = 50F"
```

Figure 4.19 Conversion affichée par la fenêtre de débogage

Dans cet exemple, nous avons utilisé un **changement** nœud. Le nœud de changement offre les fonctionnalités suivantes :

- Définir une propriété à une valeur
- Modifier une propriété de chaîne (en effectuant la recherche et le remplacement)
- Supprimer une propriété
- Déplacer une propriété

Avec l'opération set, nous pouvons définir une valeur à msg.payload. Cette valeur peut être une chaîne ou un nombre comme dans l'exemple ci-dessus, ou elle peut provenir d'un autre message.

Fonction d'enregistrement dans une bibliothèque

La figure 4.20 montre comment les instructions de la fonction de conversion oC en oF peuvent être enregistrées dans une bibliothèque appelée **TempConv**. Pour ce faire, cliquez sur l'icône du livre dans le nœud de fonction, puis cliquez sur **Enregistrer dans la bibliothèque**, et cliquez sur **Sauver**

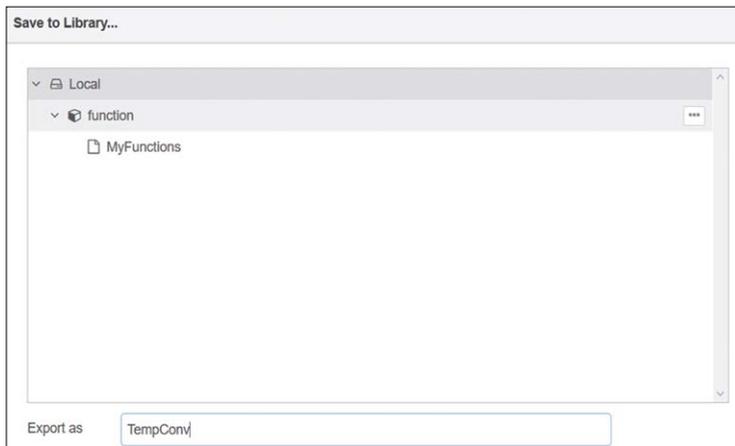


Figure 4.20 Fonction d'enregistrement dans une bibliothèque

Fonction de chargement à partir d'une bibliothèque

La figure 4.21 montre comment un élément de bibliothèque peut être chargé à partir de la bibliothèque. Dans cet exemple, l'article de brary **TempConv** qui a été enregistré plus tôt est chargé dans la fonction. Ouvrez la bibliothèque en cliquant sur le bouton **livre** icône dans **fonction** nœud, sélectionner **TempConv** et cliquez **Charger**.

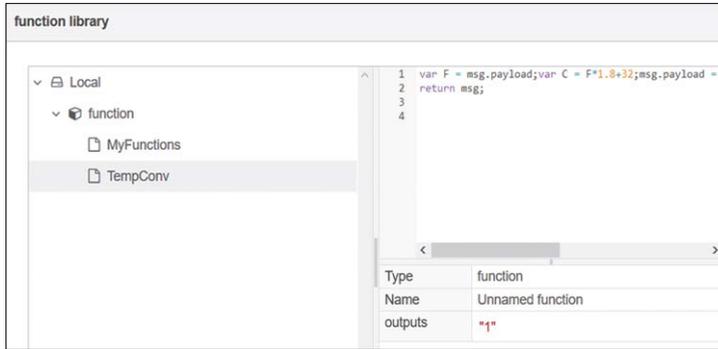


Figure 4.21 Chargement d'un élément de bibliothèque

Il y a beaucoup de noeuds disponibles dans le standard Node-RED, et il n'est pas possible d'en examiner tous les détails dans ce livre. Il existe également des noeuds tiers, créés et distribués par plusieurs entreprises. Nous allons seulement regarder les nœuds et nous familiariser avec les nœuds que nous utiliserons dans des projets avec notre Raspberry Pi. Nous pouvons résumer que les nœuds sont les parties fondamentales de node-RED et que certains de leurs domaines d'application sont :

- Accéder aux broches GPIO du Raspberry Pi. Cela nous permet de- Des capteurs et des actionneurs connectés à la Raspberry Pi
- Envoi et réception de paquets de données à partir d'Internet. Cela nous permet, pour ex- Pour effectuer des opérations de contrôle à distance, par exemple pour contrôler l'équipement sur le Wi-Fi
- Communication entre machines
- Détecter et recevoir des données depuis le Cloud. Cette application nous permet de- Utilisation des applications IoT avec l'aide de Node-RED et Raspberry Pi
- Apprendre et améliorer nos compétences en programmation

4 8 Importation et exportation de programmes d'écoulement

Nous pouvons importer et exporter des programmes de flux entre ordinateurs en utilisant les fonctionnalités d'exportation et d'importation de Node-RED. Les flux sont exportés sous forme de fichier JSON et également importés sous forme de fichier JSON.

4 . 8 . 1 Flux d'exportation

Presse **CTRL** et en même temps Cliquez sur chaque nœud du flux que vous voulez exporter. Les nœuds cliqués seront mis en surbrillance. Si vous souhaitez exporter tous les nœuds de l'espace de travail, cliquez sur **CTRL+A**. Puis, cliquez sur **Menu -> Exporter** . Une fenêtre comme dans la figure 4.22 s'affiche. Cliquez **Copier dans le presse-papiers** . Il y a trois onglets sur la fenêtre avec le **sélectionner- nœuds** onglet sélectionné. Vous pouvez le modifier pour exporter l'intégralité du flux ou tous les flux de l'espace de travail en sélectionnant **flux de courant** ou le **tous les flux** tabs respectivement. Vous devez maintenant ouvrir un fichier dans un éditeur de texte et coller le contenu des données du flux JASON dans ce fichier en utilisant **Coller**, ou **CTRL+V**. Enfin, donnez un nom au fichier et enregistrez-le dans un dossier de votre choix.

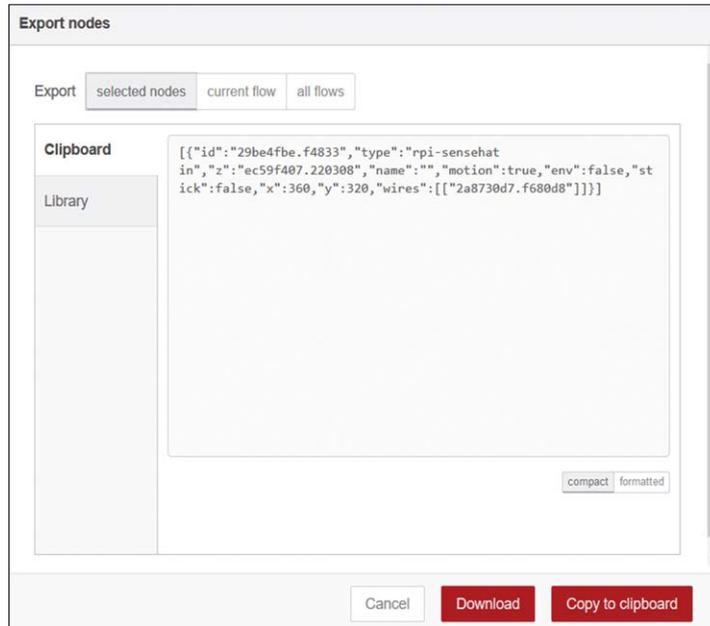


Figure 4.22 Exportation d'un flux

4.8.2 Importation de flux

Un fichier de flux au format JASIN doit être ouvert dans un éditeur de texte et les données de flux copiées dedans en utilisant **CTRL+C**. Alors vous devriez aller à **Menu->Import** et coller le fichier à partir du clip- dans la fenêtre. Le nouveau flux peut être importé dans un flux existant ou dans un nouveau flux. Enfin, cliquez sur **Importation** pour importer le flux. Si vous constatez que le bouton Importer ne fonctionne pas (p. ex., la couleur du bouton n'est pas modifiée), il est probable que le fichier que vous essayez d'importer n'est pas un fichier JSON valide.

4.9 Copier des nœuds dans le même espace de travail

Il peut y avoir des applications où vous pouvez vouloir copier un nœud dans Node-RED. Pour cela, il suffit de **CTRL** et cliquez sur les nœuds pour les copier, ou appuyez sur **CTRL+A** pour sélectionner tous les nœuds de l'espace de travail. Cliquez ensuite sur **CTRL+C** pour copier les nœuds. Déplacez le curseur à l'endroit où vous voulez que les nœuds soient copiés et appuyez sur **CTRL+V** pour coller les nœuds dans le nouvel emplacement de l'espace de travail.

4.10 nœuds de base

Lorsque Node-RED est installé sur Raspberry Pi, il commence avec des catégories de nœuds connues sous le nom de nœuds de base (voir la figure 4.5). Nous allons maintenant examiner brièvement les nœuds de chaque catégorie.

4.10.1 Nœuds d'entrée

injecter : ce nœud injecte un timestamp ou texte dans un message Le nœud peut être conu- Il est recommandé d'injecter manuellement ou à intervalles réguliers.

attraper : Si un nœud donne une erreur lors de la manipulation d'un message, il est probable que le flux s'arrête. Le nœud catch peut être utilisé pour récupérer les erreurs et renvoyer des messages à propos de l'erreur.

mqtt en : ce nœud se connecte à un courtier MQTT et s'abonne aux messages du sujet spécifié

http dans : ce nœud crée un point de terminaison HTTP pour la création de services web

websocket : ce nœud fournit un point de terminaison pour un navigateur afin qu'une connexion WebSocket puisse être établie avec Node-RED

tcp : ce nœud est utilisé pour établir des communications TCP sur un port spécifié. Il détecte les demandes TCP entrantes

udp : ce nœud est utilisé pour établir la communication en utilisant le protocole UDP. Il accepte les paquets UDP entrants

en série dans : ce nœud lit les données série du port série du périphérique local. Le nœud peut être configuré pour lire à des moments spécifiques.

4 10 2 Nœuds de sortie

déboguer : Ce nœud est utilisé pour afficher les messages sur la fenêtre de débogage. Le nœud peut être configuré pour afficher soit le message entier ou juste le msg.payload

En dehors : ce nœud se connecte à un courtier MQTT et publie des messages

réponse http : ce nœud renvoie des réponses aux requêtes HTTP reçues du nœud d'entrée HTTP.

requête http : ce nœud envoie des requêtes HTTP et renvoie la réponse

websocket : ce nœud envoie msg.payload au websocket configuré

tcp : ce nœud répond à un port TCP configuré.

udp : ce nœud envoie des messages UDP à l'hôte UDP configuré où l'annonce IP- Le port est spécifié. La diffusion est prise en charge

série : ce nœud envoie des données au port série défini du périphérique local. Le nœud peut être configuré pour envoyer certains caractères de contrôle tels que la nouvelle ligne après un message

4 103 Nœuds de fonction

fonction : c'est un nœud de traitement, utilisé pour écrire des fonctions compatibles avec JavaScript

modèle : ce nœud est utile pour construire des messages et des fichiers de configuration où les paires name:value sont insérées dans le modèle

retard : ce nœud retarde les messages par un temps aléatoire ou spécifique

déclencher : ce nœud peut être utilisé comme un minuteur de surveillance. Il peut également être utilisé pour créer deux messages de sortie avec des intervalles de temps lorsqu'un message d'entrée est reçu

commentaire : ce nœud est utilisé pour insérer des commentaires

requête http : ce nœud est utilisé pour construire et envoyer une requête HTTP à une URL donnée

demande de TCP : ce nœud envoie msg.payload à un serveur TCP et attend une réponse. Le nœud est configurable, par exemple, il peut être configuré pour attendre un caractère spécifique, ou pour revenir immédiatement

commutateur : ce nœud est utilisé pour acheminer les messages en fonction de leurs propriétés

changement : ce nœud peut être utilisé pour définir, modifier ou supprimer les propriétés des messages entrants

gamme : ce nœud met en correspondance les données numériques d'entrée avec les nouvelles données de sortie

csv : ce nœud analyse msg.payload et convertit au format CSV ou à partir de celui-ci. L'entrée peut être une chaîne, auquel cas un objet JavaScript est émis. Si, d'autre part, l'entrée est un objet JavaScript, alors une chaîne au format CSV est générée

html : ce nœud est utilisé pour extraire des données d'un document de type html dans msg.payload

json : ce nœud convertit vers ou à partir d'un objet JSON et d'un objet JavaScript

xml : ce nœud convertit vers ou à partir du format XML et de l'objet JavaScript

aléatoire : ce nœud génère un nombre aléatoire entre une valeur haute et une valeur basse

lisse : ce nœud est utilisé pour diverses fonctions telles que max, min, moyenne, haute et filtre passe-bas

rbe : c'est le nœud rbe (Report By Exception) qui génère le message si son entrée est différente de la précédente

4 10 4 Nœuds sociaux

envoyer par courriel : ce nœud est utilisé pour lire les nouveaux emails à mesure qu'ils arrivent sur l'hôte local. Il peut être configuré pour lire plusieurs fois

twitter dans : ce nœud est utilisé pour renvoyer des messages tweeter

envoyer par courriel : ce nœud est utilisé pour envoyer des messages électroniques

twitter out : ce nœud tweete le msg.payload sur le compte utilisateur configuré. Des messages texte et binaires (image) peuvent être envoyés

4 10 5 Nœuds de stockage

queue : ce nœud termine un fichier et injecte le contenu dans le flux

déposer : ce nœud lit le fichier spécifié et envoie son contenu sous la forme de msg.payload

fichier : ce nœud écrit le fichier msg.payload dans le fichier spécifié

4 10 6 Nœuds d'analyse

sentiment : ce nœud analyse le msg.payload et évalue les mots entrants en utilisant la liste de mots AFINN-165, et attache une propriété sentiment.score au msg

4 10 7 nœuds avancés

regarder : ce nœud surveille un dossier pour les changements et envoie des événements lorsque les fichiers sont ajoutés, modifiés, créés ou supprimés

feedparse : ce nœud surveille un flux RSS/Atom pour les nouvelles entrées et s'il y a de nouvelles entrées, il les livre comme messages

exécutif : ce nœud appelle une commande système et fournit stdout, stderr, et le code de retour

4 . 10 . 8 Raspberry Pi nodes

rpi gpio dans : il s'agit du nœud d'entrée Raspberry Pi. Selon l'état de la- Ce nœud génère une charge utile msg.payload avec un 0 ou 1. Nous utiliserons ce nœud dans les projets basés sur Raspberry Pi dans les chapitres suivants

rpi gpio out : c'est le nœud de sortie du Raspberry Pi. La broche hardware sélectionnée de Raspberry Pi est définie sur HIGH ou LOW selon que msg.payload est 0 ou 1. Nous utiliserons ce nœud dans les projets basés sur Raspberry Pi dans les chapitres suivants

rpi souris : c'est le nœud du bouton de la souris Raspberry Pi. Le nœud génère un 0 ou 1 lorsque le bouton sélectionné est cliqué et relâché. Une souris USB est requise

rpi clavier : ce nœud est utilisé pour capturer les frappes de clavier à partir d'un clavier USB

Sense HAT : c'est le nœud d'entrée (ou de sortie) Raspberry Pi Sense HAT. Il y a deux nœuds avec le même nom : un pour l'entrée, et l'autre pour la sortie

4 . 11 Projet 4 – Numéro des dés

Dans ce programme, nous présenterons les **aléatoire** node. Le programme affichera un nombre entre 1 et 6 chaque fois qu'on clique dessus pour démarrer. Les étapes sont les suivantes :

- Créer un **injection** nœud avec le titre **Votre chance**
- Créer un **aléatoire** node et nommez-le comme Dice. Double-cliquez pour l'éditer et définir pour générer des nombres entiers entiers entre 1 et 6 comme indiqué dans la figure 4.23. Cliquez **Terminé**

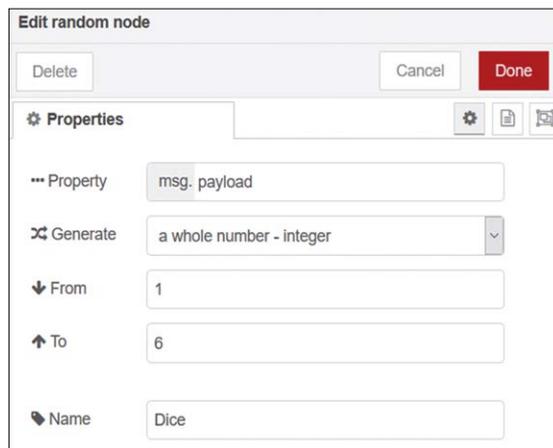


Figure 4.23 Générer des nombres aléatoires entre 1 et 6

- Créer un **déboguer** noeud
- Rejoindre les trois nœuds comme indiqué dans la figure 4.24 et cliquer **Déployer**



Figure 4.24 Joindre les trois nœuds

- Cliquez sur le **injecter** nodes. Vous devriez voir un nombre compris entre 1 et 6 disjouté dans la fenêtre de débogage. Chaque fois que vous cliquez sur le bouton d'injection de nœuds, un nombre aléatoire sera généré et affiché entre 1 et 6 comme indiqué dans la figure 4.25

```

03/12/2019, 21:05:05 node: da790285.bd274
Your Chance : msg.payload : number
6
-----
03/12/2019, 21:05:07 node: da790285.bd274
Your Chance : msg.payload : number
4
-----
03/12/2019, 21:05:08 node: da790285.bd274
Your Chance : msg.payload : number
1
    
```

Figure 4.25 La fenêtre de débogage

4 . 12 Projet 5 – Double dé

Dans de nombreux jeux de dés comme le backgammon, deux dés sont jetés à tout moment par chaque joueur avant de faire un mouvement. Le programme donné dans le projet précédent est modifié dans ce projet de sorte que deux nombres aléatoires de dés sont générés et affichés chaque fois que le **injecter** Le bouton nodes est cliqué. Ce programme utilise deux **aléatoire** les nœuds et une **rejoindre** Le nœud et les étapes de conception du flux sont donnés ci-dessous :

- Créer un **injection** nœud avec le titre **Votre chance**
- Créer deux **aléatoire** les nœuds avec leurs noms **Dice1** et **Dice2**. Double-cliquez pour les éditer et définir pour générer des nombres entiers entiers entre 1 et 6 pour les deux nœuds comme dans le projet précédent.
- Créer un **rejoindre** noeud avec le nom **joindre les charges utiles**. Double-cliquez sur ce nœud et modifiez comme indiqué dans la figure 4.26. Notez que les parties du message sont définies à 2 et séparées par un caractère d'espace. Cliquez **Terminé**

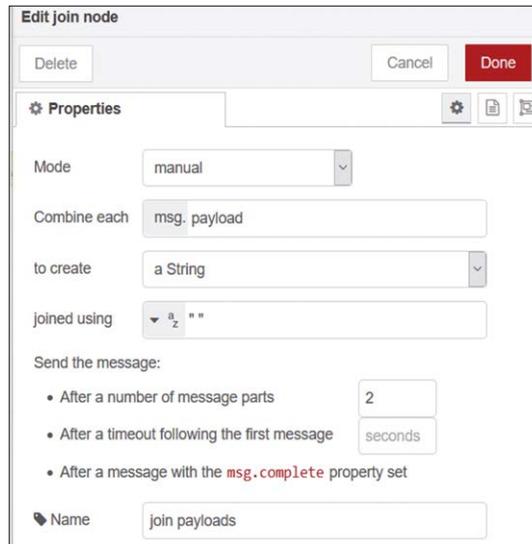


Figure 4.26 Modifier le nœud de jointure

- Créer un nœud de débogage comme avant et l'activer.
- Rejoindre les cinq nœuds comme indiqué dans la figure 4.27, et cliquer **Déployer**

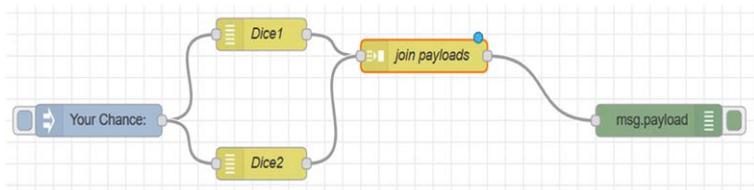


Figure 4.27 Joindre les cinq nœuds ensemble

- Cliquez sur le **injecter** nodes pour générer deux nombres aléatoires l'un à côté de l'autre dans la fenêtre Debug comme indiqué sur la figure 4.28.

```

03/12/2019, 21:59:27 node: 3d66beb2.0791ca
Your Chance : msg.payload : string[5]
"4" "6"

03/12/2019, 21:59:28 node: 3d66beb2.0791ca
Your Chance : msg.payload : string[5]
"5" "4"

03/12/2019, 21:59:28 node: 3d66beb2.0791ca
Your Chance : msg.payload : string[5]
"2" "2"

```

Figure 4.28 La fenêtre de débogage

Notez ici que, si le **Après un certain nombre de parties du message** dans la jointure de nœud était définie sur 1, alors les deux nombres de dés n'auraient pas été affichés l'un à côté de l'autre comme indiqué dans la Figure 4.29.

```

03/12/2019, 22:20:59 node: 3d66beb2.0791ca
Your Chance : msg.payload : string[1]
"3"

03/12/2019, 22:20:59 node: 3d66beb2.0791ca
Your Chance : msg.payload : string[1]
"1"

```

Figure 4.29 Affichage des deux chiffres séparément

4 . 13 Projet 6 – Conversions d'unités - Entrées multiples pour une fonction

Dans ce programme de flux, nous aurons 3 entrées pour une fonction. Le flux a 3 nœuds d'injection nommés `inchToCm`, `cmToinch` et `mToCm` où,

inchToCm : ce nœud injecte un nombre à la fonction pour convertir les pouces en cm. Dans cet exemple, la valeur (sujet) est définie sur 12 pouces à des fins de démonstration

cmToinch : ce nœud injecte un nombre à la fonction pour convertir cm en pouces. Dans cet exemple, la valeur (sujet) est définie sur 10 cm à des fins de démonstration

mToCm : ce nœud injecte un nombre à la fonction pour convertir les mètres en cm. Dans cet exemple, la valeur (topic) est définie sur 2 m à des fins de démonstration

Les étapes sont les suivantes :

- Créer 3 **injecter** Les nœuds ayant les caractéristiques suivantes :

Noeud	Charge utile	Sujet
1	inchToCm	12
2	cmToInch	10
3	mToCm	2

- Créer un **fonction** noeud nommé **Conv**, et entrez le code suivant dans la fonction (voir Figure 4.30) et cliquez **Terminé** :

```

var conv = msg.payload
var number = msg.topic

if(conv == "inchToCm") {

    msg.payload = number+"inch = "+number*2,54+"cm";
}
sinon si(conv == "cmToInch") {

    msg.payload = nombre+"cm = "+ nombre/2,54+"pouce";
}
sinon si(conv == "mToCm") {

    msg.payload = nombre+"m = "+nombre * 100+"cm";
}

retourner le message;

```

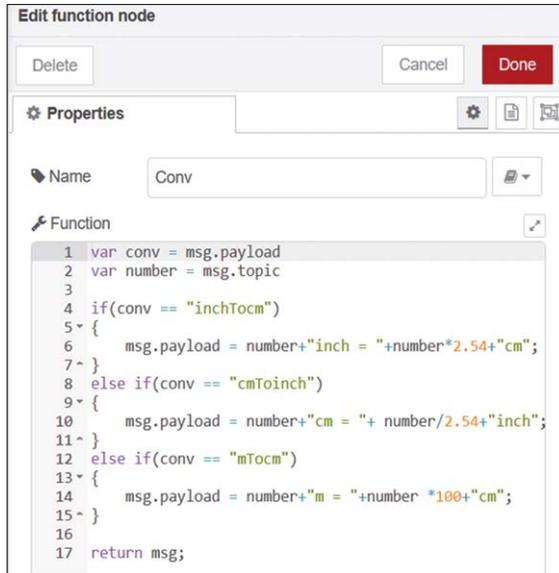


Figure 4.30 Création d'un nœud de fonction

- Créer un **débogueur** et puis connecter les 5 nœuds ensemble comme indiqué dans la Figure 4.31. Cliquez ensuite sur **Déployer**.

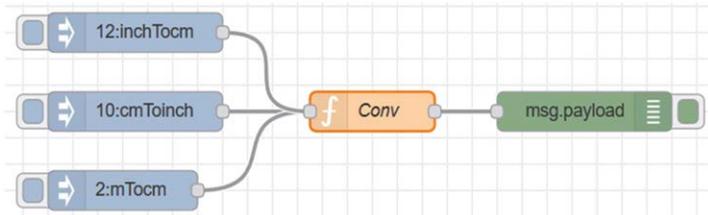


Figure 4.31 Joindre les 5 nœuds ensemble

- Cliquez sur l'un des **injecter** boutons de nœuds pour afficher la conversion dans la fenêtre Debug comme indiqué sur la figure 4.32. Dans cette figure, les nœuds inchToCm et mToCm sont cliqués.



Figure 4.32 La fenêtre de débogage

4.14 Entrées multiples et sorties multiples d'une fonction

Dans ce programme de flux, nous verrons comment plusieurs entrées et sorties peuvent être traitées dans une fonction. Dans cet exemple, nous avons 3 nœuds d'injection, un nœud de fonction d'entrée et de sortie de 3 nœuds, et un nœud de débogage. Le fonctionnement de cet exemple est le suivant :

Lorsque le bouton de **injecter** node Le premier clic, le message **Premier nœud cliqué** est dis- joué par le noeud de débogage Debug1

Lorsque le bouton de **injecter** node La seconde fois que l'on clique, le message **Deuxième nœud cliqué** est affiché par le noeud de débogage Debug2

Lorsque le bouton de **injecter** Troisième clic, le message **Troisième nœud cliqué** est dis- joué par le noeud de débogage Debug3

Les étapes sont les suivantes :

- Créer 3 **injecter** Les nœuds ayant les caractéristiques suivantes :

Noeud	Sujet
1	Première
2	Deuxième
3	Troisième

- Créer un **fonction** noeud nommé **Func**, définissez ses Sorties sur 3 et entrez- Code à l'intérieur de la fonction (voir figure 4.33). Notez que dans une instruction impliquée- plusieurs **si** Les déclarations, il pourrait être plus lisible d'utiliser un **commutateur** . Cliquez maintenant sur **Terminé** :

```
var conv = msg.topic

if(conv == "Premier") {

    var msg1 = {payload : "First node clicked"}; return
    [msg1,null,null];
}
sinon si(conv == "Second") {

    var msg2 = {payload : "Second node clicked"};
    return [null,msg2,null];
}
sinon si(conv == "Troisième") {

    var msg3 = {payload : "Third node clicked"}; return
    [null,null,msg3];
}
```

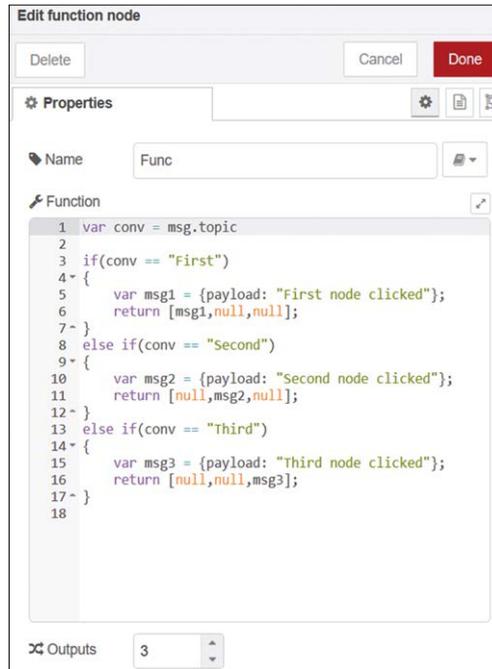


Figure 4.33 Création d'un nœud de fonction

- Créer 3 nœuds de débogage avec les caractéristiques suivantes :

Noeud	Production	Nom
1	msg.payload	Debug1
2	msg.payload	Debug2
3	msg.payload	Debug3

- Joindre les 7 nœuds comme indiqué dans la figure 4.34, cliquez sur Déployer

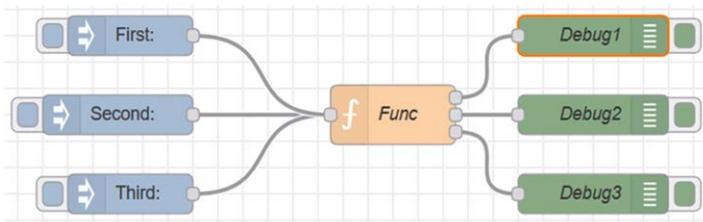


Figure 4.34 Joindre tous les nœuds

- Activer les 3 nœuds de débogage
- Cliquez sur l'un des **injecter** nodes pour afficher un message. La figure 4.35 montre les messages affichés dans la fenêtre de **injecter** Les nœuds Premier et Troisième ont été cliqués.

```

04/12/2019, 14:48:19 node: Debug1
msg.payload : string[18]
"First node clicked"

04/12/2019, 14:48:21 node: Debug3
msg.payload : string[18]
"Third node clicked"
    
```

Figure 4.35 Fenêtre de débogage

4 . 15 Projet 7 – Moyenne des nombres - Utilisation du nœud lisse

Dans ce projet, nous calculons la moyenne de 4 nombres et affichons ensuite le résultat dans la fenêtre Debug. Le programme flow se compose de 4 **injecter** nœuds, un **lisse** nœud et un **déboguer**. Le projet vise à montrer comment utiliser les technologies de l'information et des communications. Ce projet a pour but de **lisse** nœud pour calculer la moyenne d'un groupe de nombres. Dans ce projet, les nombres dont la moyenne est calculée sont 3, 7, 8 et 2.

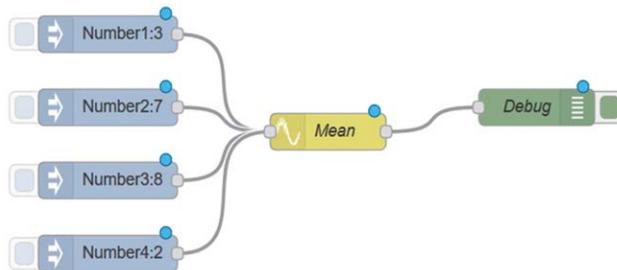


Figure 4.36 Le programme de flux du projet 7

Les étapes sont les suivantes :

- Créer 4 **injecter** Les nœuds ayant les caractéristiques suivantes :

Noeud	Charge utile	Sujet
1	3	Number1
2	7	Number2
3	8	Number3
4	2	Number4

- Créer un nœud lisse nommé **Signifier**, comme indiqué dans la figure 4.37, cliquez sur **Terminé**, et définissez les paramètres suivants :

Propriété : msg.payload
 Action : Renvoie la valeur moyenne
 Sur les 4 dernières valeurs
 Traitement : Tous msg comme un seul flux

Réduire : n'émettent qu'un seul message par valeur N la plus récente

L'action renvoie la moyenne (moyenne) de 4 nombres d'entrée. Treat est configuré de sorte que tous les 4 nombres d'entrée sont traités comme un flux. Le paramètre Réduire est défini pour afficher uniquement la valeur moyenne calculée et non les valeurs intermédiaires.

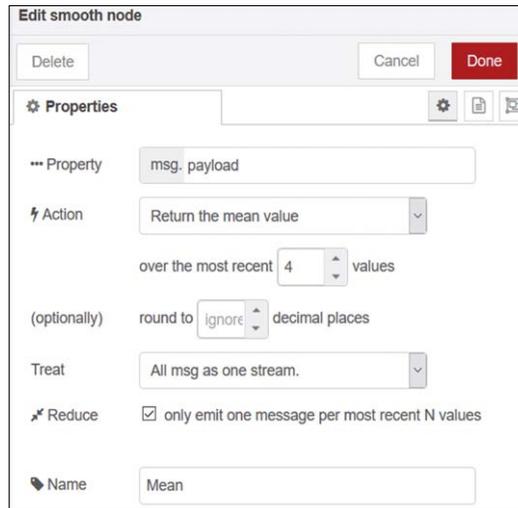


Figure 4.37 Créer un nœud lisse

- Créer un **débogueur** noeud nommé **Déboguer** et l'activer.
- Cliquez **Déployer** et vérifier qu'il n'y a pas de messages d'erreur
- Cliquez sur les boutons des quatre **injecter** nodes, en commençant par le nœud supérieur. Vous remarquerez qu'aucun id de sortie n'est produit avant que le dernier nœud injecté ne soit cliqué. Le résultat est $(3+7+8+2) / 4 = 5$ affiché dans la fenêtre de débogage comme indiqué à la figure 4.38.

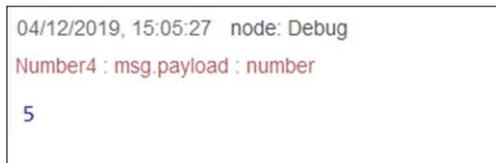


Figure 4.38 Affichage de la valeur moyenne des nombres

4 . 16 Projet 8 – Carrés de nombres

Dans ce projet, nous calculons et affichons les carrés des nombres de 1 à 5 dans la fenêtre Debug. Le programme de flux dans ce projet se compose de 3 nœuds : un **injecter** noeud, un **nœud de fonction**, et un **débogueur** nœud comme indiqué à la figure 4.39.



Figure 4.39 Le programme de flux du projet 8

Les étapes sont les suivantes :

- Créer un **injecteur** noeud avec la charge utile appelée **Cliquez**
- Créer un **fonction** noeud nommé **Carrés** et entrez les déclarations suivantes dans- côté de cette fonction (figure 4.40). La fonction affiche l'en-tête **CARRÉS DE NOMBRES** puis calcule et affiche les carrés des nombres de 1 à 5 :

```

var k;
var m

pour(k=0; k <= 5; k++) {

    si (k == 0)
    {
        msg.payload = &quot;CARRÉS DE NOMBRES&quot;;
    }
    autrement
    {
        msg.payload=&quot;&quot;;
        msg.payload = msg.payload + &quot;N= &quot;+k + &quot;      N*N= &quot;+k*k;
    } node.send(msg);
}
renvoie null;
    
```

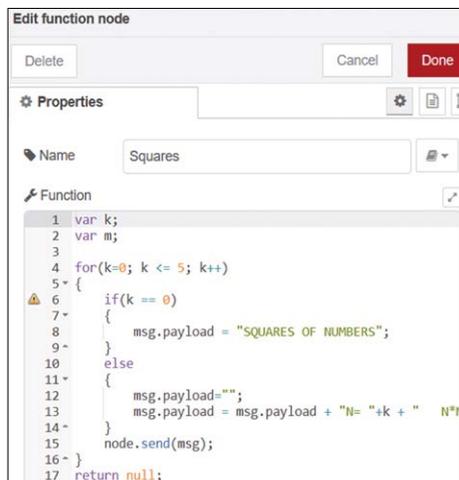


Figure 4.40 Contenu de la fonction

- Créer un noeud de débogage nommé **Débuguer**. Joindre les 3 nœuds et cliquer **Déployer**
- Cliquez sur le bouton d'injection de nœuds et vous devriez voir les carrés des nombres dis- Joué comme indiqué à la figure 4.41.

```
05/12/2019, 09:21:04 node: Debug
msg.payload : string[18]
"SQUARES OF NUMBERS"

05/12/2019, 09:21:04 node: Debug
msg.payload : string[13]
"N= 1 N*N= 1"

05/12/2019, 09:21:04 node: Debug
msg.payload : string[13]
"N= 2 N*N= 4"

05/12/2019, 09:21:04 node: Debug
msg.payload : string[13]
"N= 3 N*N= 9"

05/12/2019, 09:21:04 node: Debug
msg.payload : string[14]
"N= 4 N*N= 16"

05/12/2019, 09:21:04 node: Debug
msg.payload : string[14]
"N= 5 N*N= 25"
```

Figure 4.41 Affichage des carrés de nombres

Dans ce programme de flux, le **envoyer** La commande est utilisée pour envoyer des messages à la sortie de la fonction indépendamment des valeurs de retour. i.e. la sortie n'est pas retournée par la fonction, Ici la fonction renvoie un null mais les données réelles de la fonction sont sorties en utilisant l'instruction send.

4 . 17 Projet 9 – Obtenir les bulletins météorologiques – Afficher le bulletin météo local

Dans ce projet, nous allons obtenir le bulletin météo local et l'afficher dans la fenêtre de débogage. Tout d'abord, nous devons installer un nœud appelé **openweathermap**. Les étapes d'installation de ce nœud sont les suivantes :

- Cliquez **Menu -> Gérer la palette**
- Cliquez sur le **Installer** tab et entrer ce qui suit dans le chemin de recherche (voir la figure 4.42)

node-red-node-openweathermap

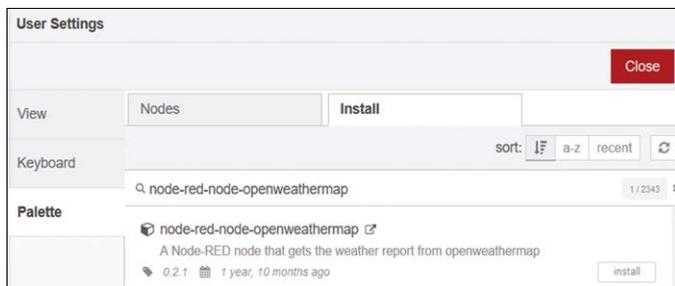


Figure 4.42 Recherche de la carte openweathermap

- Cliquez **installer** à l'intérieur du côté droit de la boîte affichant **node-red-node-openweathermap**
- Cliquez **Fermer**

Vous devriez maintenant voir les nœuds **openweathermap** et **ouvrir la carte météo** affiché dans votre palette de nœuds à l'écran Node-RED. S'il y a un problème, vous pouvez installer le nœud **openweathermap** manuellement en saisissant la commande suivante dans le répertoire racine de votre Node-RED :

```
npm install node-red-node-openweathermap
```

Maintenant que le **openweathermap** node est installé, nous devons nous inscrire et obtenir une clé API avant de pouvoir utiliser ce nœud pour obtenir les bulletins météorologiques locaux. Une clé API peut être obtenue de la façon suivante :

- Aller au site web : <https://openweathermap.org/appid>
- Cliquez sur le **API** onglet en haut de l'écran
- Cliquez **Inscrivez-vous** et remplissez le formulaire pour créer un nouveau compte
- Une nouvelle clé API sera envoyée à l'adresse électronique que vous avez fournie

Nous sommes maintenant prêts à créer et à utiliser notre programme de flux. Les étapes sont indiquées ci-dessous :

- Cliquez, faites glisser et déposez un **injecter** noeud
- Cliquez sur le bouton, faites-le glisser et déposez-le **openweathermap** nœud à l'espace de travail
- Double-cliquez sur le nœud et entrez votre clé API, sélectionnez la météo actuelle et entrez votre ville et votre pays comme indiqué dans la figure 4.43. Nommez le nœud **Météo locale**. Cliquez sur **Terminé**.

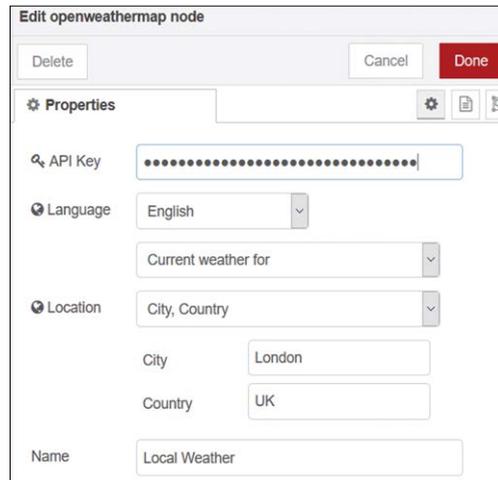


Figure 4.43 Configuration du nœud openweathermap

- Cliquez, faites glisser et déposez un nœud de débogage. Joignez les trois nœuds comme indiqué dans la figure 4.44. **Cliquez sur Déployer**



Figure 4.44 Joindre les deux nœuds

- Cliquez **injecter** bouton de nœuds. Vous devriez voir le rapport météo local au format JASON affiché dans la fenêtre Debug comme indiqué sur la figure 4.45

```
05/12/2019, 12:24:30 node: e9103ab4.83ce8 ▼
msg.payload: Object
  ◂ object
    weather: "Clouds"
    detail: "overcast clouds"
    icon: "04d"
    tempk: 279.25
    tempc: 6.1
    temp_maxc: 7.7
    temp_minc: 4.4
    humidity: 87
    pressure: 1020
    maxtemp: 280.93
    mintemp: 277.59
    windspeed: 3.1
    winddirection: 250
    location: "London"
    sunrise: 1575532133
    sunset: 1575561206
    clouds: 99
    description: "The weather in London
at coordinates: 51.51, -0.13 is
Clouds (overcast clouds)."
```

Figure 4.45 Rapport météorologique au format JASON

Comme vous pouvez le voir, le rapport est très complet, il montre que le temps était nuageux, la température au moment de faire la demande était de 6,1 °C, l'humidité était de 87%, la pression atmosphérique était de 1020 mbars, la vitesse du vent était de 3,1 m/s et ainsi de suite. La ville pour laquelle le rapport est produit et ses coordonnées sont indiquées dans la partie inférieure du rapport. Dans le prochain projet, nous ajouterons un nœud de fonction et verrons comment nous pouvons afficher uniquement la température actuelle dans la fenêtre Debug.

4.18 Project 10 – Affichage de la température actuelle

Dans ce projet, nous allons obtenir le bulletin météo local et afficher uniquement la température actuelle dans la fenêtre Debug. Ici, les nœuds suivants sont utilisés : **injecter** le nœud pour démarrer l'écoulement, **openweathermap** nœud pour obtenir le bulletin météo actuel, **fonction** nœud pour extraire la température actuelle, et **déboguer** nœud pour afficher la température actuelle dans la fenêtre de débogage.

Les étapes sont les suivantes :

- Créer un **injecter** nœud et un **openweathermap** nœud comme dans le projet précédent
- Créer un **fonction** nœud et le nommer **Température**. Double-cliquer sur ce nœud et saisir les instructions suivantes (voir la figure 4.46) :

```
var T;
T = msg.payload.tempc;
msg.payload = "Current temperature= "+T+ " Degrees C"; renvoie msg;
```

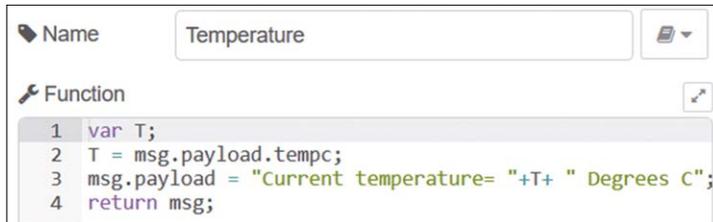


Figure 4.46 Contenu des nœuds de fonction

- Créer un **déboguer** nœud. Joignez les 4 nœuds comme dans la figure 4.47. Cliquez **Déployer**.

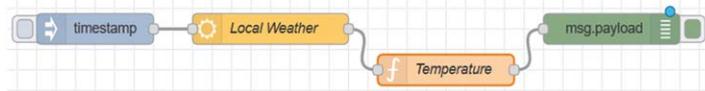


Figure 4.47 Joindre les quatre nœuds

- Cliquez **injecter** bouton de nœuds. Vous devriez voir la température actuelle affichée dans la fenêtre de débogage comme indiqué sur la figure 4.48

```
05/12/2019, 12:45:20 node: e9103ab4.83ce8
msg.payload : string[34]
"Current temperature= 6.2 Degrees C"
```

Figure 4.48 Affichage de la température actuelle

4 . 19 Projet 11 – Envoi de la température actuelle à un courriel

Dans ce projet, nous allons obtenir le bulletin météo local et envoyer la température actuelle à votre adresse courriel. Ici, les nœuds suivants sont utilisés : **injecter** le nœud pour démarrer l'écoulement, **openweathermap** nœud pour obtenir le bulletin météo actuel, **fonction** nœud pour extraire la température actuelle et le **e-mail** nœud pour envoyer le courriel.

Si le **e-mail** Le nœud n'est pas disponible dans votre palette de nœuds, vous pouvez l'installer comme suit :

- Cliquez **Menu -> Gérer la palette**
- Cliquez sur le **Installer** tab
- Saisissez ce qui suit et cliquez sur installer

```
node-red-node-email
```

- Fermer **Gérer la palette**. Vous devriez maintenant avoir 2 nœuds de messagerie dans la palette de nœuds de l'écran Node-RED

Vous pouvez également installer le nœud de messagerie en utilisant les commandes suivantes :

```
cd ~/ node-red
npm i node-red-node-email
```

Maintenant que le nœud de messagerie est installé, nous pouvons continuer à créer notre programme de flux. Les étapes sont données ci-dessous :

- Créer des nœuds : **injecter**, **openweathermap**, et **fonction** exactement comme dans le pre- projet vieux
- Cliquez, faites glisser et déposez le **e-mail** nœud (celui avec une image d'enveloppe sur son côté droit) à l'espace de travail. Double-cliquez et configurez comme indiqué dans la figure 4.49

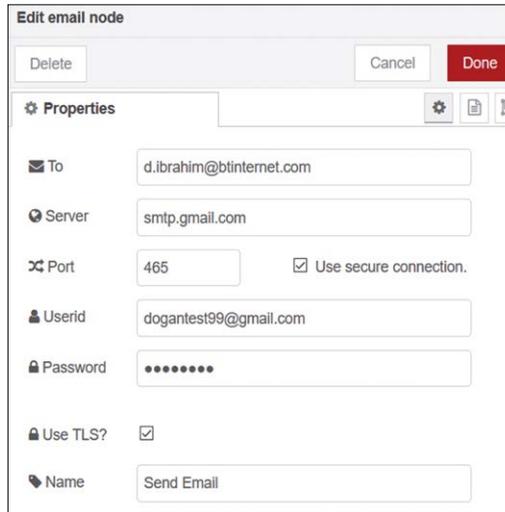


Figure 4.49 Configurer le nœud de messagerie

- Définir le **Pour** champ à l'adresse de messagerie de destination (vous pouvez spécifier plus d'une adresse de messagerie en les séparant par des virgules), définissez le **Userid** au compte de messagerie d'où le message sera envoyé, et enfin, définir le mot de passe de ce **Userid** compte.
- Cliquez **Terminé**
- Rejoindre les 4 nœuds comme indiqué dans la figure 4.40 et cliquer **Déployer**

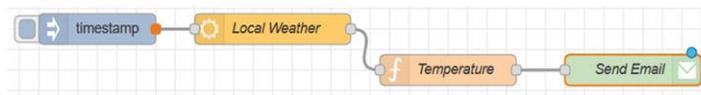


Figure 4.50 Joindre les 4 nœuds

- Si vous utilisez un compte Gmail, vous devrez activer **accès moins sécurisé** dans les paramètres de votre compte Google. Vous pouvez également saisir ce qui suit dans votre navigateur :

mienaccount.google.com/u/1/lesssecureapps

- Cliquez sur le **injecter** bouton de nœud. La température actuelle sera envoyée à l'adresse e-mail que vous avez spécifiée. La figure 4.51 montre le courrier envoyé à l'adresse e-mail de l'auteur.



Figure 4.51 Courriel reçu

Vous pouvez modifier ce projet si vous le souhaitez afin que la température soit envoyée à intervalles réguliers à votre adresse e-mail. Pour ce faire, modifiez le nœud inject et définissez l'intervalle de répétition sur la valeur requise.

4 . 20 Projet 12 – Envoi de la température et de la pression atmosphérique actuelles vers votre compte Twitter

Dans ce projet, nous allons obtenir le bulletin météo local et envoyer les relevés de température et de pression atmosphérique actuels à votre compte Tweeter. Ici, les nœuds suivants sont utilisés : **injecter** le nœud pour démarrer l'écoulement, **openweathermap** nœud pour obtenir le weath actuel- le rapport; **fonction** nœud pour extraire la température actuelle et la pression atmosphérique actuelle, et **twitter** nœud à tweeter.

Si le **twitter** Le nœud n'est pas disponible dans votre palette de nœuds, vous pouvez l'installer comme suit :

- Cliquez **Menu -> Gérer la palette**
- Cliquez sur le **Installer** tab
- Saisissez ce qui suit et cliquez sur installer

```
node-red-node-twitter
```

- Fermer **Gérer la palette**. Vous devriez maintenant avoir 2 nœuds twitter dans la palette de nœuds de l'écran Node-RED : **twitter dans** et **twitter out**

Vous pouvez également installer le nœud twitter en utilisant les commandes suivantes :

```
cd ~/. node-red  
npm i node-red-node-twitter
```

Maintenant que le nœud twitter est installé, nous pouvons continuer à créer notre programme de flux. Les étapes sont données ci-dessous :

- Créer des nœuds : **injecter**, et **openweathermap** exactement comme dans le projet précédent
- Créer un **fonction** nœud (figure 4.52), nommez-le comme suit **Presse à effleurement**, et entrez les instructions suivantes dans ce nœud :

```

var T;
T = msg.payload.tempc;
P = msg.payload.pressure; msg.payload =
"T= "+T+ " Degrees return msg;           P="+P+" mbars";

```

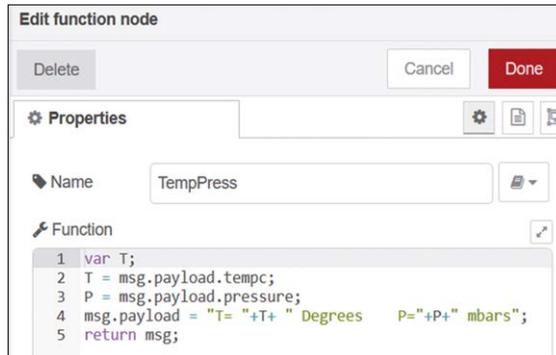


Figure 4.52 Contenu du nœud de fonction

- Cliquez, faites glisser et déposez un **twitter out** node et le nommer **Tweeter**. Double-cliquez pour configurer ce nœud. Entrez votre identifiant Twitter (p. ex., @Doganbrahim7) comme indiqué dans la figure 4.53



Figure 4.53 Entrez votre identifiant Twitter

- Cliquez sur le symbole du stylo à droite de l'identifiant Twitter. Vous serez présent- Avec un écran où vous devez entrer vos identifiants Twitter. Pour cela, il faut s'inscrire sur le site Twitter (developer.twitter.com/fr/apps) et obtenir les quatre clés suivantes propres à votre application :

Clé API, Clé secrète API, Jeton d'accès, Secret de jeton d'accès

- Entrez votre identifiant Twitter, puis copiez et collez les quatre clés que vous avez obtenues du site Twitter (voir la figure 4.54). Cliquez **Terminé** pour fermer les données du nœud

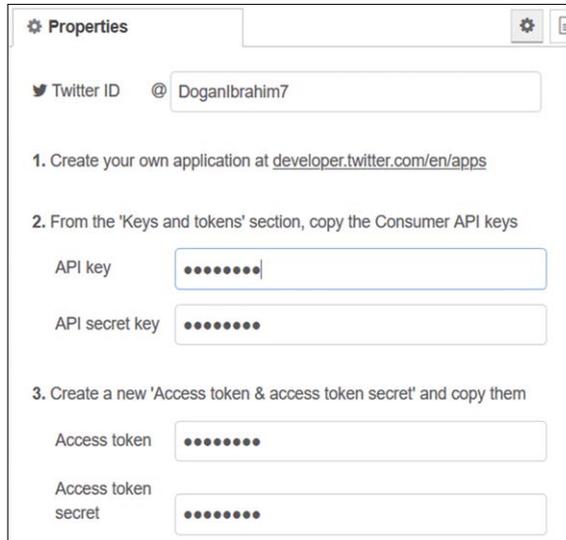


Figure 4.54 Entrez vos identifiants

- Maintenant, joignez les 4 noeuds comme indiqué dans la figure 4.55 et cliquez **Déployer**
- Cliquez sur le **injecter** bouton de noeuds. Les relevés de température et de pression atmosphérique locaux seront envoyés à votre compte Twitter, comme indiqué dans la figure 4.55 qui a été copiée du téléphone portable de l'auteur.



Figure 4.55 Température et pression atmosphérique envoyées au compte Twitter

4 . 21 Configuration du nœud-RED

Node-RED peut être configuré à l'aide d'un fichier de paramètres. Lorsque Node-RED démarre, il recherche un fichier appelé **paramètres. js** dans le répertoire utilisateur par défaut `~/ . node-red`, et ce fichier est chargé dans l'environnement d'exécution en tant que **Node . js** module. Il est également possible de définir notre fichier de paramètres en utilisant l'option **- paramètres** dans la ligne de commande.

Le fichier de paramètres par défaut a la plupart de ses lignes commentées. Vous pouvez activer le commentaire- Les lignes commentés, mais il est recommandé de savoir exactement ce que vous voulez faire avant d'activer les lignes commentées. Pour activer une ligne de commentaire, il suffit de supprimer les deux caractères (`//`) en face de la ligne requise.

En outre, il y a un **Paramètres** option dans le **Menu** lorsque Node-RED est opérationnel,

où l'utilisateur **Voir**, **Clavier**, et **Palette** options peuvent être configurées comme indiqué dans la figure- ure 4.56.

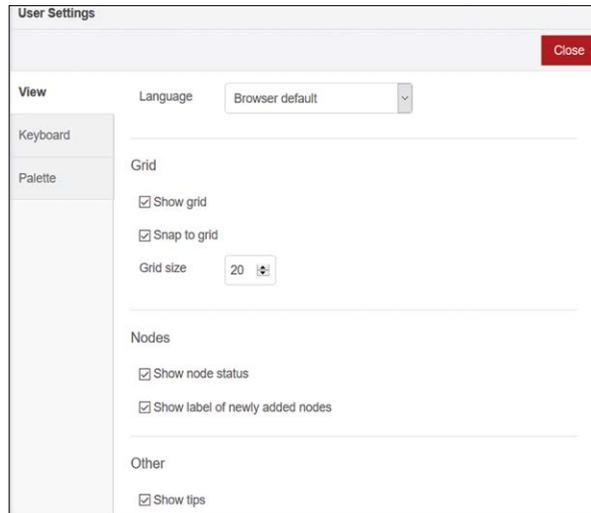


Figure 4.56 Option de menu des paramètres

4 . 22 Résumé

Dans ce chapitre, nous avons appris comment démarrer le logiciel Node-RED sur Raspberry Pi. En-tion, nous avons appris à utiliser certains nœuds Node-RED dans divers programmes de flux. Plusieurs exemples et projets sont donnés dans ce chapitre pour familiariser le lecteur avec certains des concepts importants de Node-RED.

Dans le prochain chapitre, nous apprendrons comment utiliser Node-RED pour accéder aux broches GPIO de Raspberry Pi, et comment faire l'interface des capteurs et actionneurs à Raspberry Pi, et comment les contrôler à partir de programmes de flux basés sur Node-RED.

Chapitre 5 • Projets Raspberry Pi basés sur Node-RED utilisant GPIO

5.1 Aperçu

Toutes les versions de Raspberry Pi incluent une prise pour connecter des périphériques externes, tels que- Les moteurs, LED, actionneurs, etc. Dans ce chapitre, nous allons voir comment les périphériques externes connectés à la Raspberry Pi peuvent être accessibles depuis des programmes de flux Node-RED, en développant des projets utilisant Node-RED.

Tous les projets présentés dans ce chapitre comprendront les sections suivantes pour permettre au lecteur de comprendre comment ces projets sont élaborés :

- Titre
- Description du projet
- Viser
- Informations générales (si nécessaire)
- Schéma de principe (si nécessaire)
- Schéma de circuit
- Programme de flux Node-RED avec une description complète
- Suggestions pour des travaux supplémentaires (si nécessaire)

Avant d'entrer dans les détails des projets, il est intéressant de regarder le connecteur GPIO Raspberry Pi. Il s'agit d'un connecteur à double ligne de 40 broches de 2,54 mm de largeur, comme indiqué sur la figure 5.1.

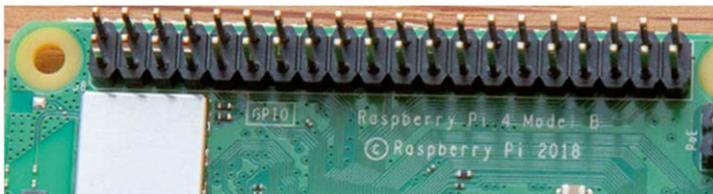


Illustration 5.1 Connecteur GPIO Raspberry Pi 4

5.2 GPIO – Interface parallèle

Lorsque le connecteur GPIO se trouve sur le côté opposé de la carte, les broches en bas, à partir de la gauche du connecteur, sont numérotées 1, 3, 5, 7, etc., tandis que celles à droite du connecteur sont numérotées 2, 4, 6, 8 et ainsi de suite (voir la figure 5.2).

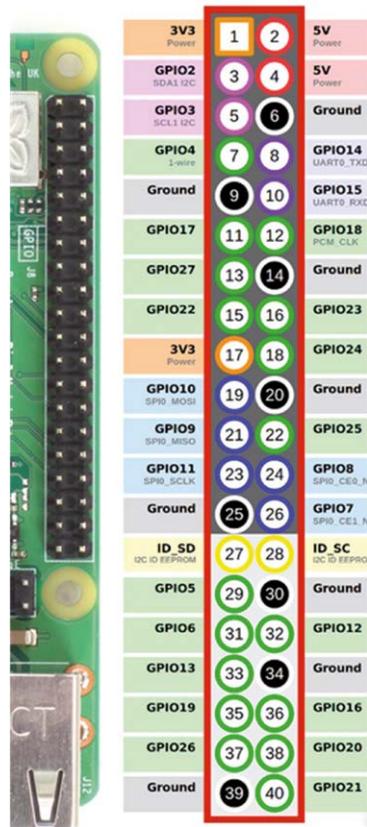


Figure 5.2 Configuration de la broche GPIO Raspberry Pi

Le GPIO fournit 26 broches d'E/S bidirectionnelles à usage général. Certaines des broches ont des fonctions multiples. Par exemple, les broches 3 et 5 sont respectivement les broches d'entrée-sortie GPIO2 et GPIO3. Ces broches peuvent également être utilisées comme bus I2C SDA I2C et I2C SCL. De même, les broches 9, 10, 11, 19 peuvent être utilisées comme entrées-sorties à usage général ou comme broches du bus SPI. Les broches 8 et 10 sont réservées pour la communication série UART.

Deux sorties de puissance sont fournies : +3.3V et +5.0V. Les broches GPIO fonctionnent à des niveaux logiques de +3,3 V (pas comme beaucoup d'autres circuits informatiques qui fonctionnent avec +5 V). Une broche peut être soit une entrée ou une sortie. Lorsqu'elle est configurée comme sortie, la tension de broche est soit 0V (logique 0) ou +3.3V (logique 1). Le Raspberry Pi est normalement alimenté par une alimentation externe (p. ex., un adaptateur secteur) avec une sortie de +5 V et une capacité de courant de 2 A au minimum. Une broche de sortie 3,3 V peut fournir jusqu'à 16 mA de courant. Le courant total tiré de toutes les broches de sortie ne doit pas dépasser la limite de 51 mA. Il faut faire attention lors de la connexion d'appareils externes aux broches GPIO car tirer des courants excessifs ou court-circuiter une broche peut facilement endommager votre Raspberry Pi. La quantité de courant pouvant être fournie par la broche 5V dépend de nombreux facteurs tels que le courant requis par le Pi lui-même, le courant pris par les périphériques USB, le courant de la caméra, le courant du port HDMI, etc.

Lorsqu'elle est configurée comme entrée, une tension supérieure à +1,7V sera prise en compte comme logique 1 et une tension inférieure à +1,7V sera prise en compte comme logique 0. Il faut veiller à ne pas fournir de tensions supérieures à +3,3V à n'importe quelle broche d'E/S car des tensions importantes peuvent facilement endommager votre Pi.

5.3 Projet 13 – Commande LED

Description : Dans ce projet, nous allons connecter une LED à l'une des broches GPIO de Raspberry Pi et ensuite allumer ou éteindre la LED d'un programme de flux Node-RED.

Objectif : Le but de ce projet est de montrer comment une LED peut être connectée à un port GPIO et comment elle peut être contrôlée depuis un programme de flux Node-RED.

Renseignements généraux : La tension avant d'une LED typique est d'environ 1,8 V. Le courant de passage de la DEL dépend de l'intensité lumineuse requise et du type de DEL utilisé. En général, 3mA devrait donner assez de lumière visible pour les petites LED. Parce que la tension de sortie d'une broche GPIO est +3.3V, nous devons utiliser une résistance de limitation de courant en série avec la LED. La valeur de la résistance de limitation du courant est calculée comme suit :

$$R = (3,3V - 1,8V) / 3mA = 500 \text{ Ohm. Nous pouvons choisir une résistance de } 470 \text{ Ohm}$$

Veiller à ce que la DEL comporte deux broches : anode (broche longue) et cathode (broche courte). La broche de la cathode doit être connectée à la terre (voir la figure 5.3).

Schéma de circuit : La figure 5.4 montre le schéma du projet où l'anode de la LED est connectée à la broche GPIO 2 du Raspberry Pi par une résistance de limitation de courant.

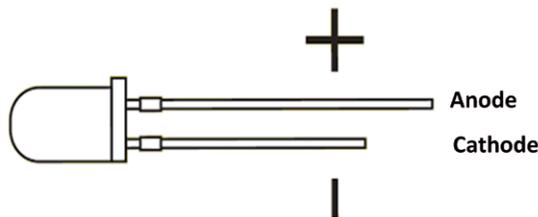


Figure 5.3 Broches d'une DEL

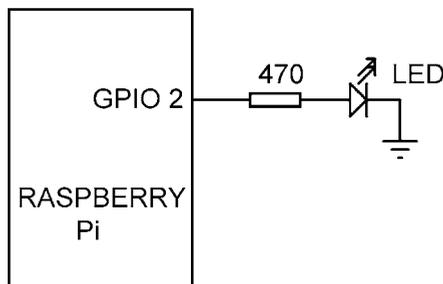


Illustration 5.4 LED connectée à la broche GPIO 2

Programme de flux Node-RED : La figure 5.5 montre le programme de flux. Dans ce programme, deux **injecter** les nœuds et une **rpi gpio out** nœuds sont utilisés.

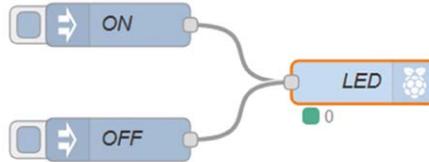


Figure 5.5 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud avec la configuration comme indiqué à la figure 5.6

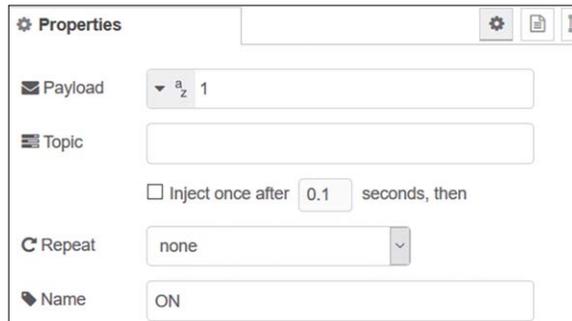


Figure 5.6 Créer un nœud d'injection

- Créer un autre **injecter** nœud avec la configuration comme indiqué à la figure 5.7

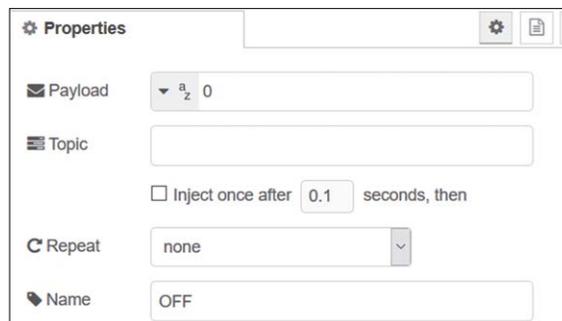


Figure 5.7 Créer un autre nœud d'injection

- Créer un **rpi gpio out** nœud avec la configuration comme indiqué dans la Figure 5.8, définissez le Type sur la sortie numérique, Initialiser l'état de broche à 0 et nommez-le comme LED. Cliquez **Terminé**

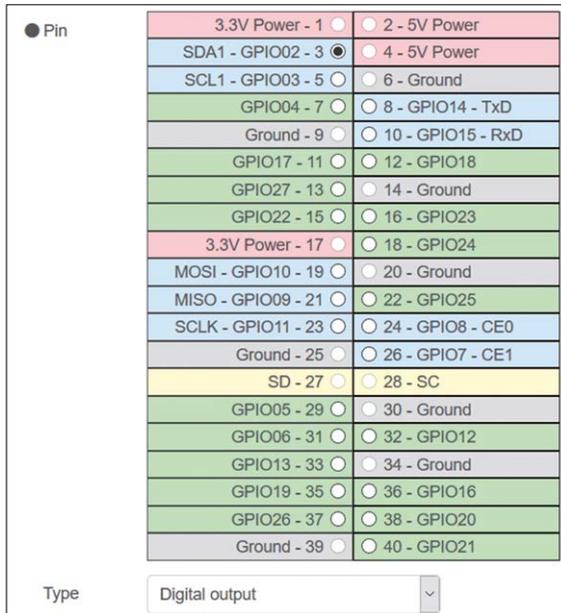


Figure 5.8 Créer un nœud rpi gpio out

- Raccorder les 3 nœuds ensemble comme indiqué dans la figure 5.5, et cliquer **Déployer**.
- Connecter la LED à Raspberry Pi comme indiqué sur la figure 5.4. Le circuit a été construit sur une planche à pain et connecté au connecteur GPIO Raspberry Pi en utilisant des fils de cavalier comme indiqué dans la figure 5.9.

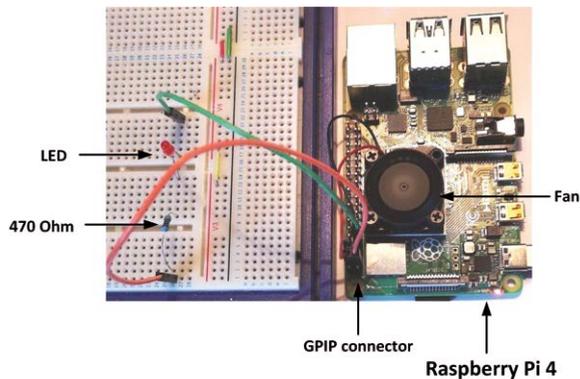


Figure 5.9 Circuit construit sur une planche à pain

- Cliquez sur le bouton de **injecter** Le voyant LED doit être allumé. Cliquez sur le bouton de la **injecter** node OFF, et cette fois la LED doit être éteinte

Comme le montre la figure 5.10, vous pouvez, si vous le souhaitez, connecter **déboguer** nœuds au flux de sorte que l'état de la LED puisse être affiché à tout moment dans la fenêtre Debug. La figure 5.11 montre l'état de la DEL lorsque celle-ci est allumée ou éteinte.

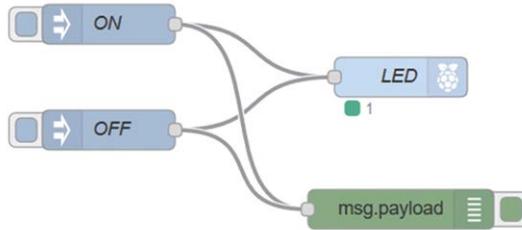


Figure 5.10 Connexion des nœuds de débogage au flux

```

06/12/2019, 14:15:02 node: 742ade48.ded8a
msg.payload : string[1]
"1"

06/12/2019, 14:15:04 node: 742ade48.ded8a
msg.payload : string[1]
"0"
    
```

Figure 5.11 État de la DEL affichée

5 . 4 Projet 14 – LED clignotante

Description : Dans ce projet, nous allons connecter une LED à l'une des broches GPIO de Raspberry Pi comme dans le projet précédent, puis faire clignoter la LED toutes les secondes.

Objectif : Le but de ce projet est de montrer comment une LED connectée à une broche de port GPIO peut être mise en clignotant en utilisant un **déclencher** nœud.

Schéma de circuit : La LED est connectée à Raspberry Pi comme dans la figure 5.4.

Programme de flux Node-RED : La figure 5.12 montre le programme de flux. Dans ce programme, un **injecternœud**, un déclencheur **nœud** et un **rpi gpio out nœuds** sont utilisés.



Figure 5.12 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud qui s'exécute régulièrement toutes les 2 secondes, dont la configuration est illustrée à la figure 5.13. Cliquez sur **Terminé**.

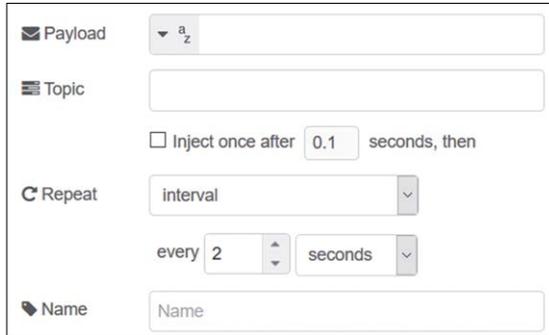


Figure 5.13 Créer un nœud d'injection

- Créer un **déclencher** nœud pour envoyer 1, attendre 1 seconde puis envoyer 0, avec la configuration comme illustré dans la Figure 5.14. Cliquez sur **Terminé**.

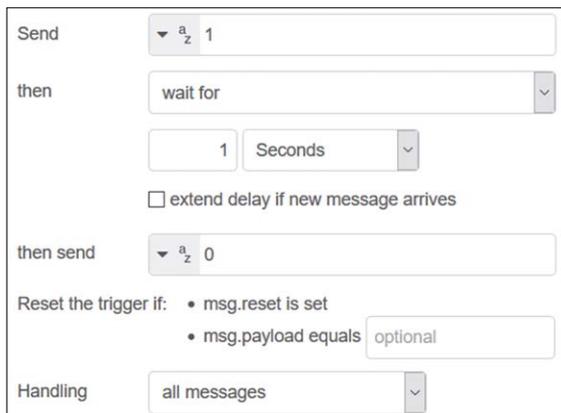


Figure 5.14 Création d'un nœud de déclenchement

- Créer la **rpi gpio out** nœud comme dans le projet précédent. Joindre les 3 nœuds, et cliquer **Déployer**.
- Cliquez sur le **injecter** bouton de nœud. Vous devriez voir la LED clignoter chaque seconde.

Dans ce projet, les **injecter** le noeud active la **déclencher** noeud toutes les deux secondes, qui à son tour contrôle la LED en la mettant en marche pendant une seconde et en l'éteignant pendant une seconde.

Vous pourriez insérer un **déboguer** Le flux de l'affichage est affiché dans la fenêtre du programme d'écoulement (figure 5.15), de sorte que l'état de la DEL peut être affiché. De plus, vous pouvez voir la précision du calendrier du projet dans la fenêtre Debug. Ceci est illustré à la figure 5.16.



Figure 5.15 Insertion d'un noeud de débogage



Figure 5.16 Affichage de l'état du LED

5 . 4 . 1 Utiliser une variable de contexte pour clignoter la LED

Dans le programme précédent, nous avons utilisé un **déclencher** noeud avec un **injecter** noeud et un **rpi gpio out**. Dans cette section, nous allons apprendre à utiliser une **fonction** noeud au lieu de **déclencher** Le point de clignotement de la LED.

La fonction de cet exemple utilisera des variables de contexte. Une variable de contexte est une variable de stockage statique dans un noeud qui ne perd pas sa valeur entre les appels.

La figure 5.17 montre le programme de flux à l'aide d'une fonction. Dans cet exemple, le **injecter** le noeud est donné **Toutes les secondes** et il est configuré pour répéter chaque seconde. Le **fonction**le noeud est donné **Toggle LED** et son contenu sont les suivants :

```

si(context.led == 0)
    contexte.led = 1;
autrement
    context.led = 0;
msg.payload = context.led; return
msg;
    
```



Figure 5.17 Programme de flux du projet

5 . 5 Projet 15 – LED clignotantes à la fois

Description : Dans ce projet, nous allons connecter deux LED à nos broches de port Raspberry Pi GPIO et ensuite clignoter ces LED alternativement chaque seconde.

Objectif : Le but de ce projet est de montrer comment plus d'une LED connectée aux broches du port GPIO et aussi comment ces LED peuvent être clignotées alternativement en utilisant **déclencher** nœuds.

Schéma de circuit : Les LED sont connectées aux broches de port Raspberry Pi GPIO 2 et GPIO 3 comme indiqué sur la figure 5.18.

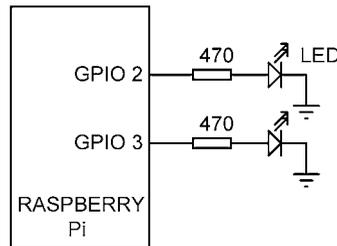


Figure 5.18 Schéma du projet

Programme de flux Node-RED : La figure 5.19 montre le programme de flux. Dans ce programme, un **injecternœud**, deux déclencher **nœuds** et deux rpi gpio out **nœuds** sont utilisés, un pour chaque LED.

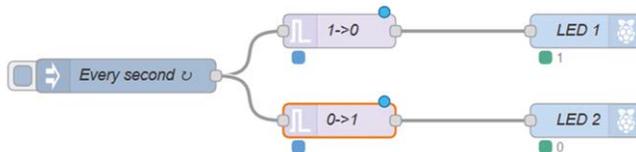


Figure 5.19 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud qui s'exécute régulièrement toutes les 2 secondes
- Créer un **déclencher** nœud pour envoyer 1, attendre 1 seconde et ensuite envoyer 0, avec la configuration comme illustré dans la Figure 5.14, nommez-le 1->0, puis cliquez sur **Terminé**.
- Créer un autre **déclencher** nœud pour envoyer 0, attendre 1 seconde et ensuite envoyer 1, nommez-le comme 0->1 et cliquez sur **Terminé**.
- Créer un **rpi gpio out** nœud comme dans le projet précédent et définir la broche GPIO num- Plus de 2
- Créer un autre **rpi gpio out** nœud comme précédemment, et définir le numéro de broche GPIO sur 3

- Cliquez **Déployer**
- Cliquez sur le **injecter** bouton de nœud. Vous devriez maintenant voir les deux LED clignotant alter- Chaque seconde.

Dans ce projet, les **déclencher** le nœud 1->0 produit un niveau de signal en baisse, tandis que le nœud 0->1 sort- augmente le niveau du signal. Le résultat est que lorsqu'une LED est allumée, l'autre est éteinte et vice versa. La **injecter** le noeud active la **déclencher** noeud toutes les deux secondes comme auparavant.

Comme dans le projet 13, vous pouvez créer deux **déboguer** les nœuds et se rattachent aux sorties des deux **déclencher** Les nœuds pour afficher l'état des deux LED.

5 . 5 . 1 Utilisation de variables de contexte pour clignoter les LED alternativement

Dans le programme précédent, nous avons utilisé **déclencher** nœuds avec un **injecter** nœud et deux **rpi gpio out** les deux DEL en alternance. Dans cette section, nous allons apprendre à utiliser une **fonction** nœud au lieu des deux **déclencher** les nœuds pour clignoter les LED.

La fonction de cet exemple est configurée pour avoir deux sorties et utilisera deux variables de contexte comme décrit dans la section 5.4.1.

La figure 5.20 montre le programme de flux à l'aide d'une fonction. Dans cet exemple, le **injecter** le nœud est donné **Toutes les secondes** et il est configuré pour répéter chaque seconde.

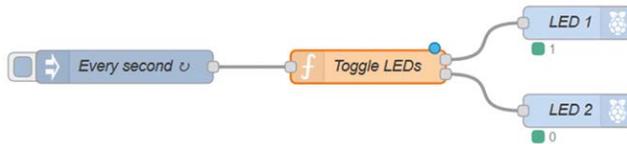


Figure 5.20 Programme de flux du projet

Le **fonction** le nœud est donné **Basculer les LED** et son contenu est le suivant (voir figure 5.21) :

```

si (context.led1 == 0) {
    context.led1 = 1;
    context.led2 = 0;
}
autrement
{
    context.led1 = 0;
    context.led2 = 1;
}

var msg1 = {payload:context.led1}; var
msg2 = {payload:context.led2}; return
[msg1, msg2];

```

```

Function
1 if(context.led1 == 0)
2 {
3   context.led1 = 1;
4   context.led2 = 0;
5 }
6 else
7 {
8   context.led1 = 0;
9   context.led2 = 1;
10 }
11
12 var msg1 = {payload:context.led1};
13 var msg2 = {payload:context.led2};
14 return [msg1, msg2];
15
Outputs 2
    
```

Figure 5.21 Configuration des fonctions

Cliquez sur le **injecter** Bouton du nœud. Vous devriez voir les deux LED clignoter alternativement chaque seconde.

Notez comment les données LED pour chaque LED sont envoyées aux sorties correspondantes de la fonction dans l'instruction de retour. La figure 5.22 montre le projet construit sur une planche à pain.

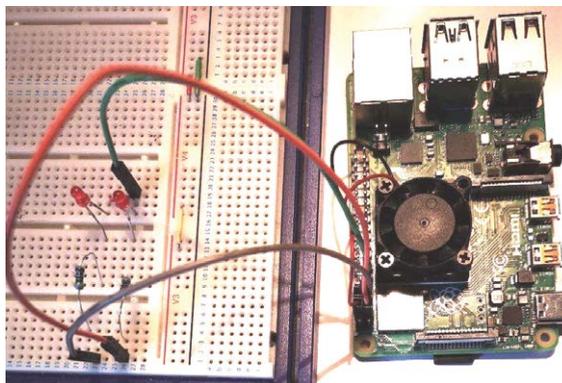


Figure 5.22 Le projet construit sur une planche à pain

5 . 6 Projet 16 – Alarme de température avec buzzer

Description : Il s'agit d'un système d'alarme de température simple. Un petit buzzer actif est connecté- Connecté à l'un des ports GPIO du Raspberry Pi. La température locale est lue à partir du site météorologique comme dans le projet 9, et si elle dépasse une valeur seuil prédéfinie, le buzzer est activé. La valeur seuil est considérée comme étant de 30 oC dans ce projet, mais elle peut être modifiée à tout autre moment si on le souhaite. La température est vérifiée toutes les minutes.

Objectif : Le but de ce projet est de montrer comment un système d'alarme simple peut être conçu avec Node-RED, en utilisant Raspberry Pi et un petit buzzer comme matériel.

Renseignements généraux : Les petits buzzers (également appelés buzzers piézoélectriques) sont couramment utilisés dans les projets d'alarme électronique. Ces avertisseurs sont utilisés avec une tension de +3V à +5V. Il existe deux types d'avertisseurs : les avertisseurs actifs et les avertisseurs passifs. Les avertisseurs actifs sont dotés de circuits électroniques intégrés et ils sonnent à une fréquence fixe lorsqu'une tension continue est appliquée sur leurs bornes. La fréquence de ces avertisseurs est généralement d'environ 1 kHz. Il est très facile d'utiliser les buzzers actifs car ils peuvent être facilement connectés à la broche du port de sortie du Raspberry Pi. Les buzzers passifs, d'autre part, nécessitent une vague- forme (généralement PWM ou sinusoïdal) à appliquer sur leurs bornes, et la fréquence du son généré dépend de la fréquence de la forme d'onde appliquée. Bien qu'il soit plus difficile d'utiliser des avertisseurs passifs, ils ont les avantages que le son- Les fréquences peuvent être facilement modifiées, par exemple, elles peuvent être utilisées dans des projets d'orgue électronique simples pour générer des sons à différentes fréquences. Dans ce projet, un buzzer actif est utilisé.

Schéma de principe : La figure 5.23 montre le schéma du projet.

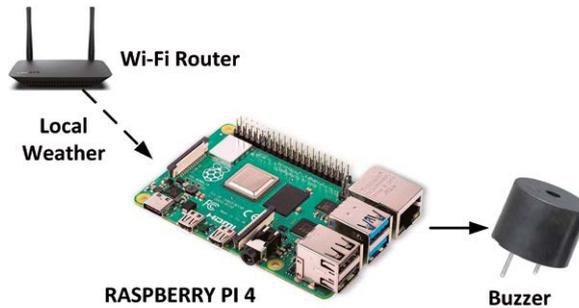


Figure 5.23 Schéma du projet

Schéma de circuit : Le schéma de circuit du projet est illustré à la figure 5.24. Le buzzer est connecté directement au port GPIO 2 du Raspberry Pi 4.

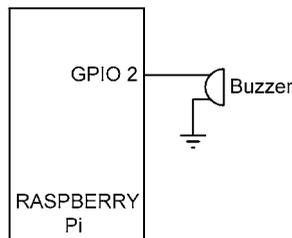


Figure 5.24 Schéma du projet

Programme de flux Node-RED : La figure 5.25 montre le programme de flux. Dans ce programme, 4 nœuds sont utilisés **injecter** nœud qui se répète chaque minute, un **openweathermap** nœud qui reçoit le bulletin météo local, un **fonction** nœud qui extrait la température actuelle de

Le bulletin météo et envoie 1 à **rpi gpioout** le nœud pour activer le buzzer si le- perature est supérieure à 30oC, sinon 0 est envoyé au rpi gpioout Node pour désactiver le buzzer.

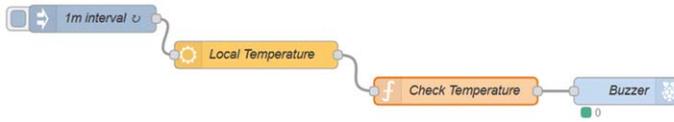


Figure 5.25 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** noeud avec le nom **intervalle de 1 m**, et régler l'intervalle de répétition sur 1 minute
- Créer un **openweathermap** noeud avec le nom **Température locale**, entrez la clé API (voir Projet 9), et définissez le nom de votre ville locale
- Créer un **fonction** noeud avec le nom **Vérifier la température**, et entrez les instructions suivantes dans ce nœud (voir la figure 5.26). Ici, si la température est supérieure à 30oC alors le buzzer est activé, sinon, le buzzer est déconnecté- tivé :

```
var T;
T = msg.payload.tempc; si(T
> 30)
    msg.payload = 1;
autrement
    msg.payload = 0;
renvoie msg;
```

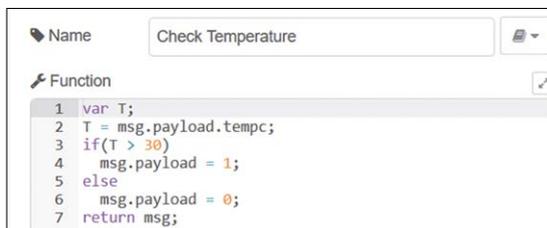


Figure 5.26 Configuration du nœud de fonction

- Créer un **rpi gpio out** noeud nommé **Buzzer**, et définissez la broche GPIO sur 2
- Rejoindre les 4 nœuds comme dans la figure 5.25, et cliquer **Déployer**
- Cliquez sur le **injecter** Le bouton du nœud. Le rapport de température locale sera maintenant obtenu toutes les minutes et si elle est supérieure à 30 o C, alors le buzzer sonnera

Vous pouvez modifier le matériel facilement, par exemple en remplaçant le buzzer par un relais pour qu'un chauffage puisse être mis ON ou OFF, ou une LED peut être utilisée pour indiquer visuellement quand une condition d'alarme se produit. En outre, un **déboguer** Le nœud peut être connecté à la sortie de l' **fonction** node pour afficher l'état du buzzer (1 ou 0 seront affichés dans la fenêtre de débogage)

5 . 7 Projet 17 – Contrôle à distance d'un port GPIO par courriel

Description : Dans ce projet, nous allons contrôler un port GPIO Raspberry Pi à distance en envoyant un email. Un petit buzzer actif est connecté à l'un des ports GPIO du Raspberry Pi. Le buzzer est activé en envoyant la commande ON par email. De même, le buzzer est désactivé en envoyant la commande OF par mail.

Objectif : Le but de ce projet est de montrer comment un port GPIO Raspberry Pi peut être contrôlé à distance en envoyant un email.

Schéma de principe : Le schéma du projet est présenté à la figure 5.23.

Schéma de circuit : Le schéma du projet est comme indiqué dans la figure 5.24, où le buzzer est connecté directement à la broche de port GPIO 2 du Raspberry Pi 4.

Programme de flux Node-RED : La figure 5.27 montre le programme de flux. Dans ce programme, 3 nœuds sont utilisés : **envoyer par courriel** nœud qui reçoit des courriels à intervalles réguliers **fonction** noeud qui envoie 1 ou 0 à sa sortie selon que la commande utilisateur est ON ou OF respective- Le CEDEFOP a **rpi gpioout nœud pour contrôler le buzzer**.



Figure 5.27 Programme de flux du projet

Les étapes sont les suivantes :

- Cliquez, faites glisser et déposez le **envoyer par courriel** nœud à l'espace de travail. Configurez ce nœud comme indiqué dans la figure 5.28 et cliquez **Terminé**. Le nœud est configuré pour vérifier la présence d'incom- Envoi d'e-mails toutes les 10 secondes. Les e-mails sont censés-compte avec le nom d'utilisateur : dogantest99@gmail.com, où le mot de passe doit également être spécifié (vous devrez remplacer ceux-ci à vos propres détails de compte Gmail)

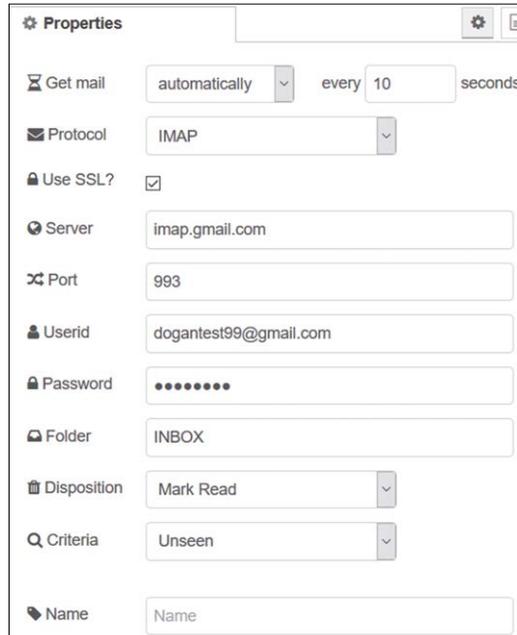


Figure 5.28 Configurer l'e-mail dans le nœud

- Créer un nœud de fonction nommé **SUR/DE**, ayant les déclarations suivantes (voir Figure 5.29). Ici, la fonction **soustraction** est utilisé pour extraire les deux premiers caractères de la commande reçue. Si la commande est **Sur**, puis 1 est retourné pour activer le buzzer. Sinon, si la commande est **De**, puis 0 est renvoyé pour désactiver le buzzer. Cliquez **Terminé** :

```
var cmd = msg.payload.substr(0,2); var On =
{payload : "1"};
var Désactivé = {payload : "0"};
```

```
si(cmd == "ON")
    retour à l'action;
sinon si(cmd == "OF")
    renvoie Off;
```



Figure 5.29 Configuration du nœud de fonction

- Créer un **rpi gpio out** et définissez le numéro de port sur GPIO 2. Joignez les 3 nœuds et cliquez **Déployer**

Le programme devrait maintenant vérifier le compte de messagerie toutes les 10 secondes. Vous devriez voir les mots **connexion** et **attirant** affichés sous **envoyer par courriel** nœud juste avant l'ac- Le nombre est vérifié. Dans ce projet, le buzzer doit sonner lorsqu'un courriel est envoyé au compte dogantest@gmail.com avec le contenu du courrier ayant le mot **Sur**. De même, le buzzer doit cesser de sonner lorsque le mot **De** (notez qu'il est OFF et non OFF) est envoyé par courriel.

Juste un rappel, si vous utilisez un compte Gmail, vous devrez activer **accès moins sécurisé** dans les paramètres de votre compte Google. Vous pouvez également saisir ce qui suit dans votre navigateur :

[mienaccount.google.com/u/1/lesssecureapps](https://myaccount.google.com/u/1/lesssecureapps)

Le buzzer dans ce projet peut être remplacé par exemple par un relais si désiré afin que l'électro- Les équipements électriques (p.ex. appareils électroménagers) peuvent être contrôlés à distance en envoyant des e-mails de n'importe où dans le monde.

5 . 8 Projet 18 – Confirmation de l'état du buzzer

Description : Dans le projet précédent, nous avons vu comment contrôler un port GPIO Raspberry Pi à distance en envoyant des emails. Le problème ici est que lorsque nous envoyons une commande au Raspberry Pi, nous ne sommes pas sûrs si le port GPIO a pris la valeur requise. Dans ce projet, nous lisons l'état du port GPIO et envoyons un email pour confirmer l'état de la broche. La confirmation est très importante dans certaines applications, par exemple si nous voulons activer notre chaudière domestique à l'aide d'un relais, nous voulons nous assurer que la chaudière a bien été activée.

Si la commande **Sur** est envoyé pour activer le buzzer, puis le message **Buzzer est activé** sera envoyé à l'adresse e-mail spécifiée lorsque le buzzer est réellement activé. Si d'autre part la commande **De** est envoyé, puis le message **Buzzer est OFF** sera envoyé à l'adresse e-mail indiquée.

Objectif : Le but de ce projet est de montrer comment l'état d'une broche de port GPIO peut être lu et puis un message peut être envoyé à un compte de messagerie pour confirmer l'état de la broche de port GPIO.

Schéma de principe : Le schéma du projet est présenté à la figure 5.23.

Schéma de circuit : Le schéma du projet est présenté à la figure 5.30. Ici, le buzzer est connecté directement à la broche de port GPIO 2 du Raspberry Pi 4 comme dans le projet précédent, et GPIO 2 est connecté à GPIO 3 où GPIO 3 est configuré comme une entrée en Node-RED. L'état du buzzer sera lu par le GPIO 3 et un message sera envoyé à un compte de messagerie.

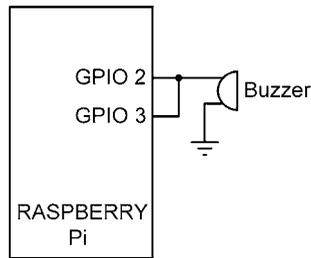


Figure 5.30 Schéma du projet

Programme de flux Node-RED : La figure 5.31 montre le programme de flux. Dans ce programme, 6 nœuds sont utilisés : 3 nœuds pour recevoir le courrier électronique et contrôler le buzzer en conséquence, et 3 nœuds pour envoyer l'état du port de buzzer à l'adresse e-mail spécifiée.

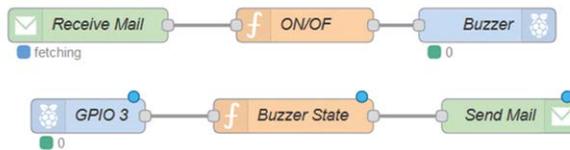


Figure 5.31 Programme de flux du projet

Les étapes sont les suivantes :

- Les 3 premiers nœuds utilisés pour recevoir des emails sont identiques à ceux du projet précédent et ne sont pas répétés ici (seul le nom de l'email dans le nœud est modifié)
- Les 3 nœuds suivants sont utilisés pour recevoir l'état du port GPIO 3 et envoyer un e-mail à l'adresse spécifiée.
- Cliquez **rpi gpio** dans node, faites-le glisser et déposez-le dans l'espace de travail. Nommez ce **GPIO 3** et le configurer pour recevoir l'état du port GPIO 3 comme indiqué dans la Figure 5.32, cliquez sur **Terminé**. Ce port recevra l'état de la broche du port GPIO 2 qui est connecté à GPIO 3.

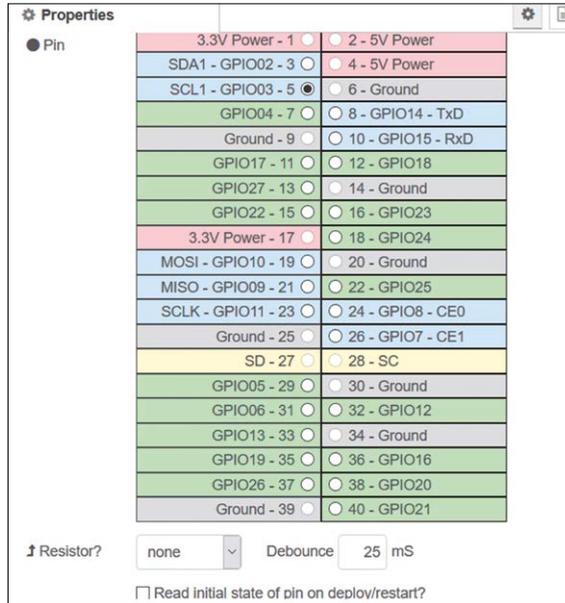


Figure 5.32 Configurer le rpi gpio dans le noeud

- Créer un **fonction** nœud, nommez-le comme **Buzzer State**, entrez l'état suivant- ments et cliquez sur **Terminé** (figure 5.33) :

```

if(msg.payload == "1")
    msg.payload = "Buzzer is ON"
else
    msg.payload = "Buzzer is OFF"
return msg;
    
```

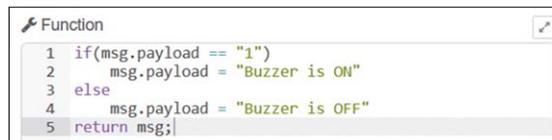


Figure 5.33 Configuration du nœud de fonction

- Cliquez sur **envoyer par courriel** node, le glisser-déposer dans l'espace de travail et le nommer comme **Envoyer le courrier**. Configurer ce nœud comme indiqué dans la figure 5.34. L'état du buzzer est envoyé à l'adresse de courriel : d.ibrahim@btinternet.com à partir de l'adresse électronique : dogantest99@gmail.com (vous devez saisir vos propres adresses e-mail et mots de passe)

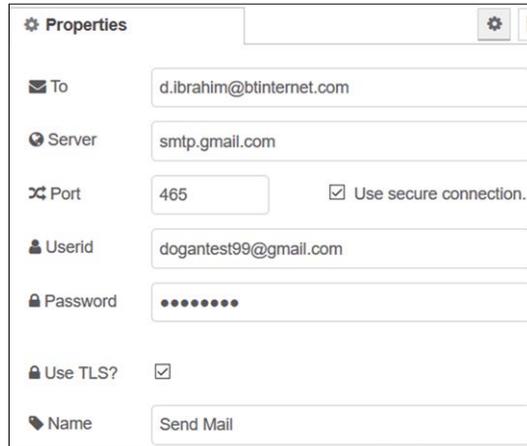


Figure 5.34 Configurer le nœud de sortie des e-mails

- Rejoindre les nœuds comme dans la figure 5.31, et cliquer **Déployer**. Pour tester le programme, envoyer la commande **Sur** à l'adresse e-mail dogantest99@gmail.com, et le buzzer doit être activé. En même temps, le message **Buzzer est activé** sera envoyé à l'adresse de courriel d.ibrahim@btinternet.com (vous devez utiliser vos propres adresses e-mail et mots de passe).

5 . 9 Projet 19 – Contrôle à distance de plusieurs ports GPIO par courriel

Description : Dans ce projet, nous allons contrôler 4 ports GPIO Raspberry Pi à distance en envoyant un email. Quatre LED sont connectées aux ports GPIO du Raspberry Pi par l'intermédiaire de résistances de limitation de courant. Les LED sont allumées/éteintes en envoyant des commandes par courriel de la manière suivante :

ON=1	Allumez LED1
ON=2	Allumez LED2
ON=3	Allumez LED3
ON=4	Allumez LED4
DE = 1	Éteignez LED1
DE = 2	Éteignez LED2
DE = 3	Éteignez LED3
DE = 4	Éteignez LED4

Objectif : Le but de ce projet est de montrer comment plusieurs ports GPIO du Raspberry Pi peuvent être contrôlés à distance en envoyant des messages électroniques.

Schéma de principe : La figure 5.35 montre le schéma du projet.

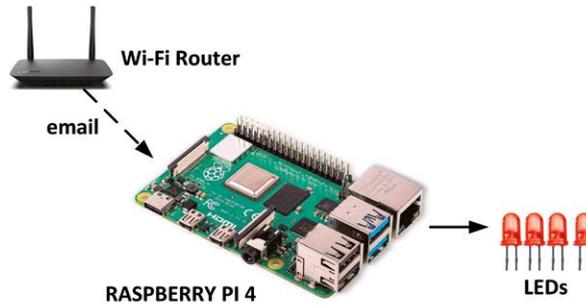
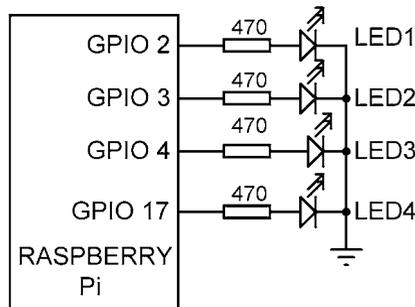


Schéma de circuit : Le schéma du projet est présenté à la figure 5.36. Les DEL sont connectées aux broches de port Raspberry Pi suivantes par l'intermédiaire de résistances de limitation de courant de 470 ohms :

LED	Raspberry Pi Nom du port	Raspberry Pi Port Pin Number 3
LED1	GPIO 2	5
LED2	GPIO 3	7
Led3	GPIO 4	11
Led4	GPIO 17	

La broche de port numéro 9 du Raspberry Pi est la broche de mise à la terre qui est connectée aux broches cathodiques de toutes les LED.



Programme de flux Node-RED : La figure 5.37 montre le programme de flux. Dans ce programme, 6 nœuds sont utilisés **envoyer par courriel**. Le nœud pour vérifier et recevoir l'e-mail à intervalles réguliers, un **fonctionnœud** avec 4 sorties pour décoder les commandes de l'utilisateur et 4 rpi gpio out **nœuds pour contrôler les 4 LED**.

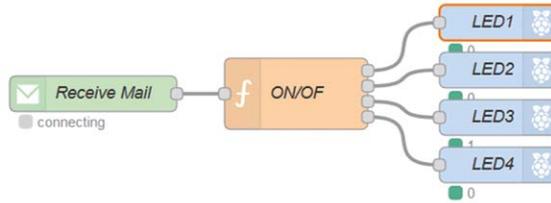


Figure 5.37 Programme de flux du projet

Les étapes sont les suivantes :

- Créer **e-mail** dans le nœud comme dans les projets précédents et le nommer **Recevoir un courriel**
- Créer un **fonction** nœud nommé **ON/OFF** avec les énoncés suivants (voir la figure 5.38). Variable **cmd** extrait les 3 premiers caractères de la commande et pour des commandes valides, elle a la valeur **ON=** ou **OF=**. Variable **port** extrait le nombre de DEL requis et pour les commandes valides, il a les valeurs de **1**, **2**, **3**, ou **4**. Notez comment les sorties multiples sont traitées dans la fonction :

```

var cmd = msg.payload.substr(0,3); var port =
msg.payload.substr(3,1); var On = {payload :
"1"};
var Désactivé = {payload : "0"};

si(cmd == "ON=")
{
    si (port == "1")
        retour [On, null, null, null]; sinon
    si(port == "2")
        retour [null, On, null, null]; sinon
    si(port == "3")
        retour [null, null, On, null]; sinon
    si(port == "4")
        retour [null, null, null, On]
}

if(cmd == "OF=") {

    si (port == "1")
        retour [Désactivé, null, null, null]; sinon
    si(port == "2")
        return>null, Off, null, null]; sinon if(port
    == "3")
        retour [null, null, Off, null]; sinon
    si(port == "4")
        retour [null, null, null, off]
}
    
```

```

Name ON/OFF

Function
1 var cmd = msg.payload.substr(0,3);
2 var port = msg.payload.substr(3,1);
3 var On = {payload: "1"};
4 var Off = {payload: "0"};
5
6 if(cmd == "ON=")
7 {
8     if(port == "1")
9         return [On,null,null,null];
10    else if(port == "2")
11        return [null,On,null,null];
12    else if(port == "3")
13        return [null,null,On,null];
14    else if(port == "4")
15        return [null,null,null,On]
16 }
17
18 if(cmd == "OF=")
19 {
20     if(port == "1")
21        return [Off,null,null,null];
22    else if(port == "2")
23        return [null,Off,null,null];
24    else if(port == "3")
25        return [null,null,Off,null];
26    else if(port == "4")
27        return [null,null,null,Off]
28 }

```

Figure 5.38 Configuration de la fonction (seule une partie du code est affichée)

- Créer 4 **rpi gpio out** , un pour chacune des LED, LED1, LED2, LED3 et LED4. Définissez les noms de broches de port sur **GPIO 2**, **GPIO 3**, **GPIO 4** et **GPIO 17** respectivement et délimitez les états initiaux des ports à 0. Un exemple pour **GPIO 17** est indiqué dans la figure 5.39. Cliquez sur Terminé **pour chaque nœud**.

Properties

SCL1 - GPIO03 - 5	<input type="radio"/>	6 - Ground	<input type="radio"/>
GPIO04 - 7	<input type="radio"/>	8 - GPIO14 - TxD	<input type="radio"/>
Ground - 9	<input type="radio"/>	10 - GPIO15 - RxD	<input type="radio"/>
GPIO17 - 11	<input checked="" type="radio"/>	12 - GPIO18	<input type="radio"/>
GPIO27 - 13	<input type="radio"/>	14 - Ground	<input type="radio"/>
GPIO22 - 15	<input type="radio"/>	16 - GPIO23	<input type="radio"/>
3.3V Power - 17	<input checked="" type="radio"/>	18 - GPIO24	<input type="radio"/>
MOSI - GPIO10 - 19	<input type="radio"/>	20 - Ground	<input type="radio"/>
MISO - GPIO09 - 21	<input type="radio"/>	22 - GPIO25	<input type="radio"/>
SCLK - GPIO11 - 23	<input type="radio"/>	24 - GPIO8 - CE0	<input type="radio"/>
Ground - 25	<input type="radio"/>	26 - GPIO7 - CE1	<input type="radio"/>
SD - 27	<input type="radio"/>	28 - SC	<input type="radio"/>
GPIO05 - 29	<input type="radio"/>	30 - Ground	<input type="radio"/>
GPIO06 - 31	<input type="radio"/>	32 - GPIO12	<input type="radio"/>
GPIO13 - 33	<input type="radio"/>	34 - Ground	<input type="radio"/>
GPIO19 - 35	<input type="radio"/>	36 - GPIO16	<input type="radio"/>
GPIO26 - 37	<input type="radio"/>	38 - GPIO20	<input type="radio"/>
Ground - 39	<input type="radio"/>	40 - GPIO21	<input type="radio"/>

Type:

Initialise pin state?

initial level of pin - low (0)

Figure 5.39 Configuration du port pour LED4

- Rejoindre les nœuds comme indiqué dans la figure 5.37 et cliquer **Déployer**. Pour tester le programme, connectez le matériel comme dans la figure 5.36 et envoyez un email par exemple avec le contenu de **ON=2**. LED2 doit être activé. Envoyer un courriel avec **DE = 2** et cette fois, LED2 doit être désactivé

Les LED utilisées dans ce projet peuvent être remplacées par des relais si on le souhaite afin d'équiper électriquement- ment peut être facilement contrôlé à distance en envoyant des e-mails.

La figure 5.40 montre le projet construit sur une planche à pain où les LED sont connectées au Raspberry Pi à l'aide de fils de cavaliers.

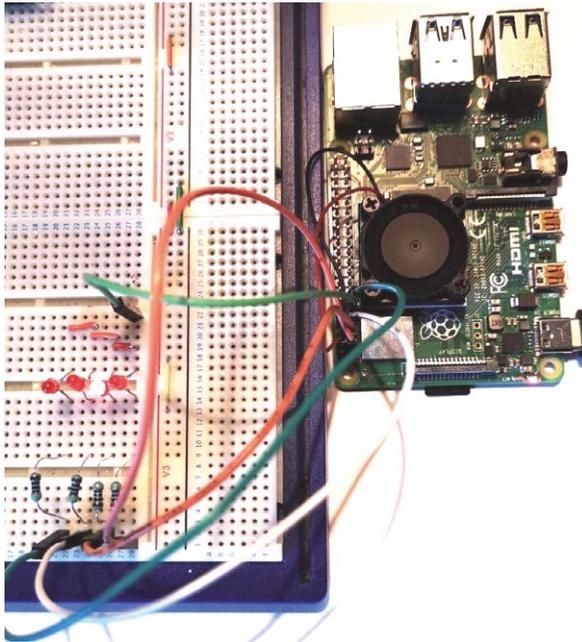


Figure 5.40 Construction du projet sur une planche à pain

Juste un rappel, si vous utilisez un compte Gmail, vous devrez activer **accès moins sécurisé** via les paramètres de votre compte Google, comme décrit dans les projets précédents utilisant les nœuds d'e-mail.

5 . 10 Projet 20 – Simulateur de feux de circulation

Description : Dans ce projet, nous allons simuler un ensemble de feux de circulation. 3 LED sont connectées aux ports GPIO du Raspberry Pi : une LED rouge, une LED orange et une LED verte. Les DEL sont allumées et éteintes comme dans la séquence PDL suivante :

FAIRE POUR TOUJOURS

Allumer la DEL rouge pendant 17 secondes

Allumer les DEL rouge et orange pendant 3 secondes

Allumez la DEL verte pendant 15
secondes Allumez la DEL orange
pendant 3 secondes

Objectif : Le but de ce projet est de montrer comment un simple programme de simulation de feux de circulation peut être développé avec Node-RED en utilisant 3 LEDs connectées à un Raspberry Pi.

Schéma de principe : La figure 5.41 montre le schéma du projet.

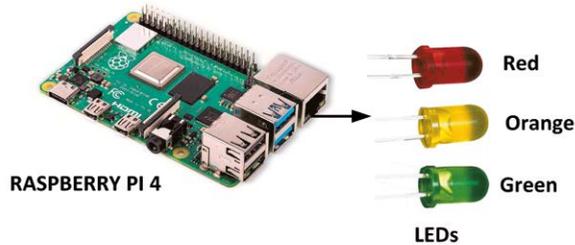


Figure 5.41 Schéma du projet

Schéma de circuit : Le schéma du projet est présenté à la figure 5.42. Les DEL sont connectées aux broches de port Raspberry Pi suivantes par l'intermédiaire de résistances de limitation de courant 470 ohms :

LED	Raspberry Pi Nom du port	Raspberry Pi Port Pin Number 3
Rouge	GPIO 2	5
Orange	GPIO 3	7
Vert	GPIO 4	

La broche de port numéro 9 du Raspberry Pi est la broche de mise à la terre qui est connectée aux broches cathodiques de toutes les LED.

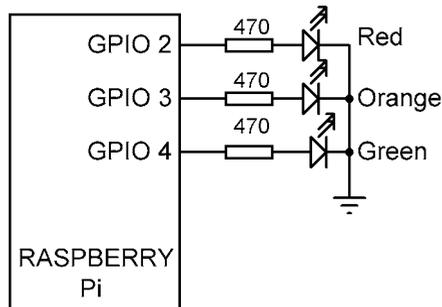


Figure 5.42 Schéma du projet

Programme de flux Node-RED : La figure 5.43 montre le programme de flux. Dans ce programme, 11 nœuds sont utilisés **injecter** nœud pour démarrer la simulation, 4 **retard** nœuds pour générer les délais de temporisation requis, 3 **déclencher** les nœuds, et 3 **rpi gpio out** nœuds.

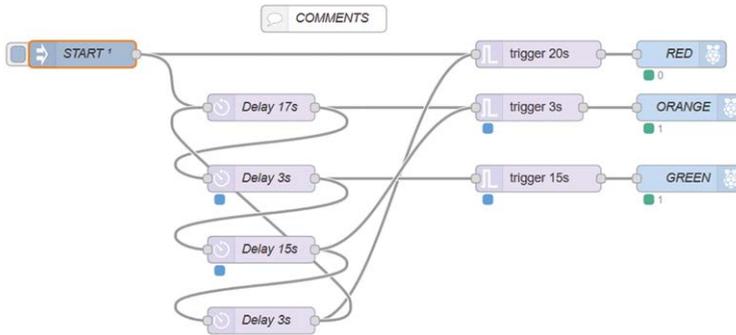


Figure 5.43 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecteur** nœud et le nommer **Démarrer**
- Créer 4 **retard** nœuds comme indiqué dans la figure 5.43. Ces nœuds généreront les délais requis. Par exemple, lorsque la LED rouge est déclenchée pour rester allumée pendant 20 secondes. Après un délai de 17 secondes, la LED orange est également allumée afin que le rouge et l'orange soient allumés pendant 3 secondes. Après ce temps, la LED verte est déclenchée pour rester allumée pendant 15 secondes, et après ce temps, la LED orange est allumée pendant 3 secondes, puis de nouveau à la LED rouge. Cette séquence est répétée pour toujours. La figure 5.44 montre un exemple typique **retard** configuration de nœud.

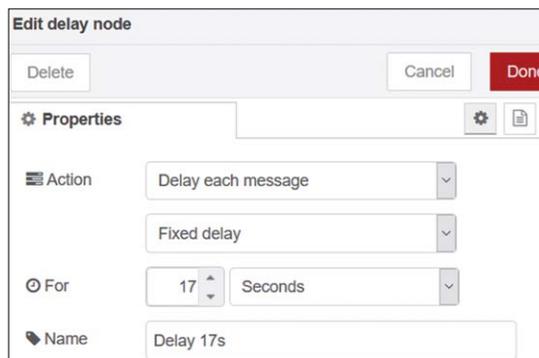


Figure 5.44 Nœud de retard de 17 secondes

- Créer 3 **déclencher** Les nœuds qui envoient 1, attendent le temps spécifié, puis envoient 0. La figure 5.45 montre un exemple de la façon dont les 20 **déclencher** le nœud est configuré.

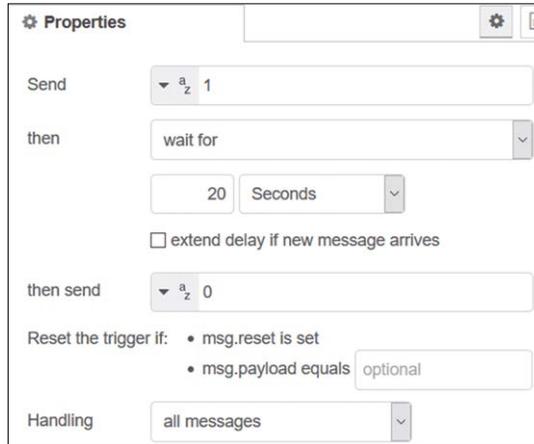


Figure 5.45 Configuration d'un nœud de déclenchement

- Créer 3 **rpi gpio out** les nœuds des broches GPIO GPIO 2, GPIO 3 et GPIO 4, et nommez-les comme **Rouge**, **Orange**, et **Vert** respectivement. À titre d'exemple, la figure 5.46 montre la configuration du nœud GREEN.

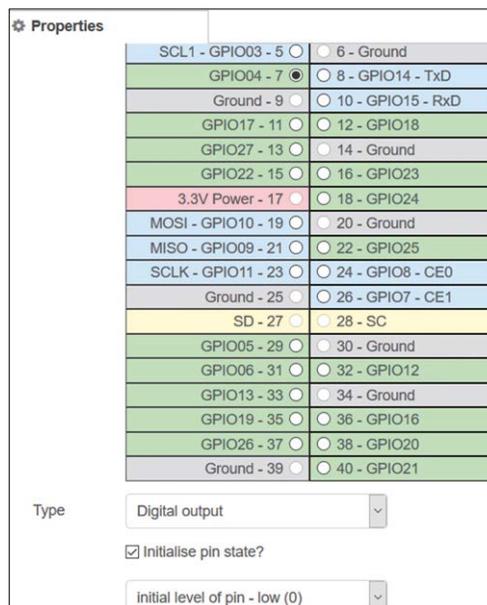


Figure 5.46 Configuration du nœud GREEN

- Joindre tous les nœuds comme indiqué dans la figure 5.43, et cliquer **Déployer**. Cliquer sur le bouton de nœud **injecter** pour démarrer la simulation. Vous devriez voir les 3 LED simulant les feux de circulation avec le calendrier spécifié au début de ce projet.

Notez que dans ce projet, le **commentaire** nœud est également introduit. Ce nœud nous aide à écrire des commentaires sur un projet. Dans cet exemple de projet, le contenu du **commentaire** nœuds sont présentés à la figure 5.47.

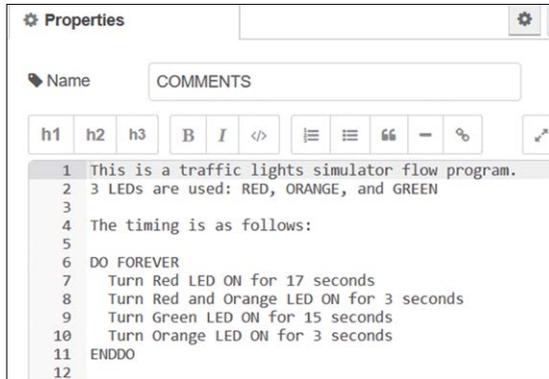


Figure 5.47 Contenu du nœud de commentaire

Ce projet a été construit sur une planche à pain comme dans les projets précédents.

5 . 11 Projet 21 – Entrée du commutateur à bouton-poussoir

Description : Il s'agit d'un projet très simple où un interrupteur à bouton-poussoir (bouton pour abrégé) et une LED sont connectés au Raspberry Pi. Le projet allume la LED lorsque le bouton est enfoncé.

Objectif : Le but de ce projet est de montrer comment un bouton peut être utilisé dans un projet basé sur Node-RED.

Renseignements généraux : Les boutons sont couramment utilisés dans les projets basés sur des microcontrôleurs afin de changer l'état d'une broche de port d'entrée. Le processeur détecte lorsque le bouton est enfoncé et exécute ensuite le code requis.

Les boutons peuvent être utilisés en deux modes : normalement LOW, et normalement HIGH. Dans un mode LOW normal, l'état de sortie du bouton est à la logique 0 par défaut et passe à la logique 1 lorsque le bouton est enfoncé. La sortie reste à cet état jusqu'à ce que le bouton soit relâché. Le schéma de circuit de la figure 5.48 montre un bouton LOW normal.

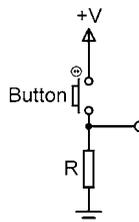


Figure 5.48 Bouton LOW normalement

Dans un bouton normalement HIGH, l'état du bouton est à la logique 1 par défaut et va à la logique 0 lorsque le bouton est enfoncé. La sortie reste à cet état jusqu'à ce que le bouton soit relâché. Le schéma de circuit de la figure 5.49 montre un bouton normalement HIGH.

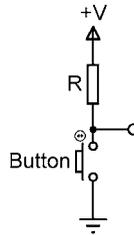


Figure 5.49 Bouton HIGH normalement

Le bouton utilisé dans ce projet est un petit composant ayant 4 broches comme indiqué sur la figure 5.50. En fait, le bouton est un dispositif à 2 broches où les broches sont parallèles pour plus de commodité.



Figure 5.50 Petit bouton et schéma de raccordement

Les commutateurs mécaniques ont des problèmes de rebond. Cela se produit parce que lorsqu'un- Les contacts mécaniques rebondissent (se déplacent d'un état à l'autre) jusqu'à ce qu'ils s'installent. Bien que ce temps de dépose puisse sembler très petit (environ 10ms), il est en fait un temps très long quand on considère le temps de traitement d'un microcontrôleur. Des circuits spéciaux de débogage peuvent être utilisés pour éliminer cet effet de rebond. Heureusement, le **rpi gpio dans node** fournit un paramètre pour définir le temps de désaisonnalisation du commutateur afin que les contacts du commutateur ne soient pas lus jusqu'à ce que ce temps expire. Ceci permet de s'assurer que le microcontrôleur lit l'état correct du commutateur.

Schéma de principe : La figure 5.51 montre le schéma du projet.



Figure 5.51 Schéma du projet

Schéma de circuit : Le schéma de circuit du projet est montré dans la figure 5.52 où la LED est connectée à la broche de port GPIO 2. La broche de port GPIO 3 est connectée à un côté du bouton, tandis que l'autre côté de la touche est connectée à la broche de terre du Raspberry Pi.

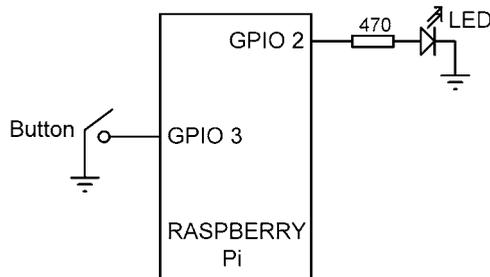


Figure 5.52 Schéma du projet

Programme de flux Node-RED : La figure 5.53 montre le programme de flux. Dans ce programme, 2 nœuds sont utilisés :

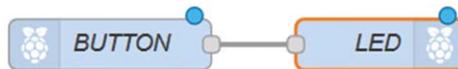


Figure 5.53 Programme de flux du projet

Les étapes sont :

- Créer un **rpi gpio dans** le noeud et le crabe comme **Bouton**, définir le nom de la broche sur GPIO 3 et définir la résistance sur Pulldown.
- Créer un **rpi gpio out** node et le nommer **LED**, définissez le nom de la broche sur GPIO 3 et initialisez le niveau de la broche à 0
- Rejoindre les deux nœuds comme indiqué dans la figure 5.51, et cliquer sur **Déployer**.

Vous devriez voir que la LED est allumée et qu'en appuyant sur le bouton, la LED est éteinte. Notez que dans ce programme de flux, nous voulons le contraire. Par exemple, la DEL doit être normalement éteinte et en appuyant sur le bouton, elle doit être allumée. Un programme de flux modifié qui fonctionnera correctement est montré à la figure 5.54

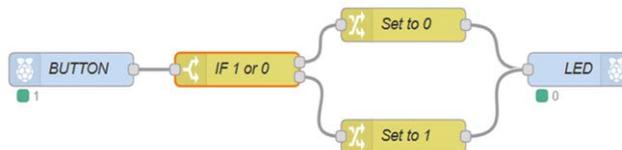


Figure 5.54 Programme de débit modifié

Cette modification peut être effectuée comme expliqué dans les étapes suivantes :

- Créer un **rpi gpio dans** Node comme ci-dessus et nommez-le comme suit **Bouton**, définir le nom de la broche sur GPIO 3 et définir la résistance sur
- Créer un **commutateur** node et le nommer **Si 1 ou 0**. Configurez ce nœud de telle sorte que si l'entrée est 1, dirigez-le vers la sortie supérieure, sinon, dirigez-le vers la deuxième sortie- put. Pour passer à la deuxième sortie, cliquez sur ajouter et sélectionnez **autrement** comme indiqué dans la figure 5.54. Cliquez sur **Terminé**.



Figure 5.55 Configuration du nœud de commutation

- Créer un **changement** node et le nommer **Régler à 0**. Configurer ce nœud pour retourner 1 comme indiqué dans la figure 5.56. Cliquez sur **Terminé**



Figure 5.56 Configuration du nœud de changement

- Créer un autre **changement** node et le nommer **Régler à 1** et définissez la sortie sur 1. Cliquez **Terminé**
- Créer un **rpi gpio out** node et le nommer **LED**, définissez le nom de la broche sur GPIO 3 et initialisez le niveau de la broche à 0
- Rejoindre les 5 nœuds comme indiqué dans la figure 5.54, et cliquer **Déployer**.

Vous devriez maintenant voir que la LED est OFF, et appuyer sur le bouton permet d'allumer la LED.

La figure 5.57 montre le circuit construit sur une planche à pain.

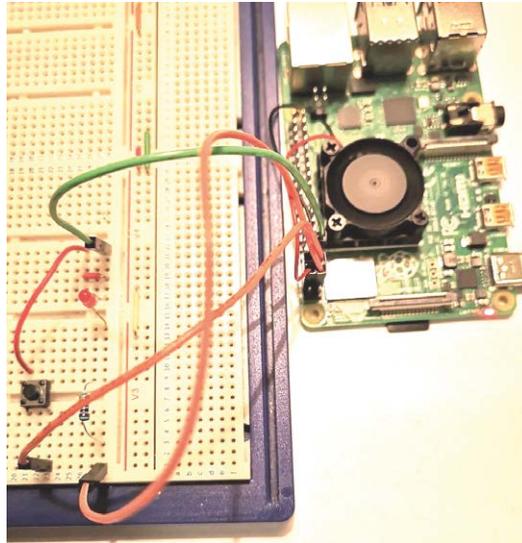


Figure 5.57 Circuit construit sur une planche à pain

5 2 12 Projet 22 – Changement de luminosité des DEL – La sortie PWM

Description : Il s'agit d'un projet très simple où la luminosité d'une LED est modifiée en changeant la tension moyenne et donc le courant moyen à travers la LED.

Objectif : Le but de ce projet est de montrer comment une forme d'onde PWM peut être générée en utilisant Node-RED.

Renseignements généraux : PWM (Pulse Width Modulation) est une technique couramment utilisée dans les circuits électroniques afin de modifier la tension moyenne fournie à une charge. Par exemple, la tension moyenne fournie à un chauffage peut être modifiée en lui envoyant une forme d'onde de tension PWM à amplitude fixe et en modifiant ensuite les caractéristiques de cette forme d'onde.

Une forme d'onde PWM est fondamentalement une forme d'onde carrée positive où le temps ON (aussi appelé le MARK) et le temps OFF (aussi appelé l'ESPACE) peuvent être changés. La figure 5.58 montre une forme d'onde PWM typique avec un point de la période T . Le rapport entre le temps ON et le temps est connu comme le cycle de travail de la forme d'onde et en changeant le cycle de travail, nous pouvons effectivement changer la tension moyenne fournie à la charge.

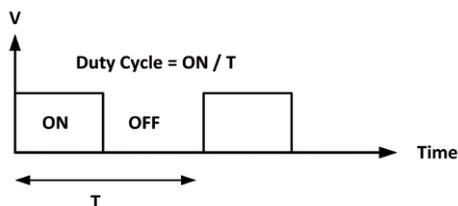


Figure 5.58 Forme d'onde PWM typique

Le cycle de travail est généralement représenté en pourcentage. Par exemple, un cycle de travail à 50 % correspond à une situation où le temps ON est égal au temps OFF. Dans un cycle de travail à 100 %, la forme d'onde est toujours activée (c.-à-d. qu'il n'y a pas de temps OFF). De même, dans un cycle de travail à 25 %, la durée du temps ON est le quart de la période de la forme d'onde. La figure 5.59 montre les formes d'onde PWM avec différents cycles de travail.

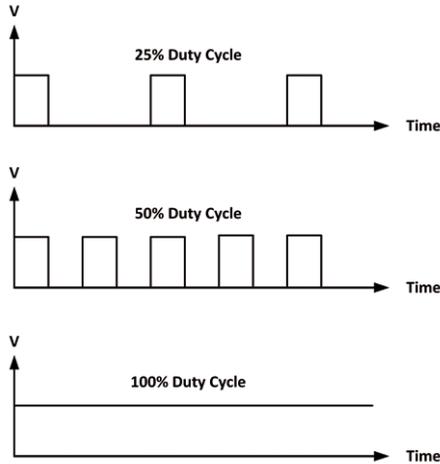


Figure 5.59 Forme d'onde PWM avec différents cycles de travail

Schéma de circuit : Le schéma de circuit du projet est comme dans la figure 5.4 où une LED est connectée à la broche port GPIO 2 de Raspberry Pi.

Programme de flux Node-RED : La figure 5.60 montre le programme de flux. Dans ce programme, 6 nœuds sont utilisés : 5 **injecter** nœuds avec différents paramètres de cycle d'évaluation et un **rpi gpio out** nœud.

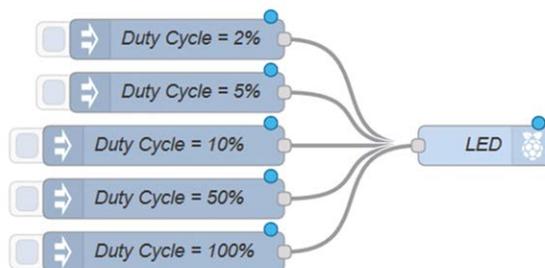


Figure 5.60 Programme de flux du projet

Les étapes sont :

- Créer 5 **injecter** , définissez les charges utiles sur des valeurs numériques de 2, 5, 10, 50, 100. Ce sont les cycles d'utilisation de la forme d'onde PWM. Nommez chaque nœud d'injection avec les cycles de service correspondants comme indiqué sur la figure 5.60. Par exemple, le con- Figure 5.61 montre la représentation du nœud avec un cycle de travail à 50

Illustration 5.61 Configuration pour un cycle de travail à 50

- Créer un nœud rpi gpio out et le nommer comme LED, définir le nom du port sur GPIO 2, et définir le type à la sortie PWM comme indiqué dans la figure 5.61. Cliquez sur Terminé

3.3V Power - 1	2 - 5V Power
SDA1 - GPIO02 - 3	4 - 5V Power
SCL1 - GPIO03 - 5	6 - Ground
GPIO04 - 7	8 - GPIO14 - TxD
Ground - 9	10 - GPIO15 - RxD
GPIO17 - 11	12 - GPIO18
GPIO27 - 13	14 - Ground
GPIO22 - 15	16 - GPIO23
3.3V Power - 17	18 - GPIO24
MOSI - GPIO10 - 19	20 - Ground
MISO - GPIO09 - 21	22 - GPIO25
SCLK - GPIO11 - 23	24 - GPIO8 - CE0
Ground - 25	26 - GPIO7 - CE1
SD - 27	28 - SC
GPIO05 - 29	30 - Ground
GPIO06 - 31	32 - GPIO12
GPIO13 - 33	34 - Ground
GPIO19 - 35	36 - GPIO16
GPIO26 - 37	38 - GPIO20
Ground - 39	40 - GPIO21

PWM output

Figure 5.62 Configuration du nœud rpi gpio out

Joindre les nœuds comme dans la figure 5.60. Cliquez sur injecter le nœud 2% et vous devriez voir que la LED est très faible. Cliquer sur des cycles de service plus élevés devrait rendre la LED plus lumineuse.

5.13 Résumé

Dans ce chapitre, nous avons appris comment connecter des composants externes aux ports GPIO de Raspberry Pi et ensuite comment accéder à ces composants depuis les programmes Node-RED.

Dans le prochain chapitre, nous allons voir comment connecter un écran LCD à notre Raspberry Pi et nous développerons également d'autres projets utilisant un écran LCD.

Chapitre 6 • Utilisation des écrans LCD avec Node-RED

6.1 Aperçu

Dans les systèmes de microcontrôleur, la sortie d'une variable mesurée est généralement affichée à l'aide de LED, d'affichages à 7 segments ou d'affichages de type LCD. Les écrans LCD présentent l'avantage de pouvoir afficher des données alphanumériques ou graphiques. Certains écrans LCD ont 40 caractères ou plus avec la possibilité d'afficher plusieurs lignes. Certains autres écrans LCD peuvent être utilisés pour afficher des images graphiques. Certains modules offrent des affichages en couleur tandis que d'autres intègrent un rétroéclairage pour pouvoir être vus dans des conditions de faible luminosité.

Il existe essentiellement deux types de LCD en ce qui concerne la technique d'interface : parallèle LCD et LCD série. Les LCD parallèles (p. ex., Hitachi HD44780) sont connectés à un microcontrôleur utilisant plus d'une ligne de données et les données sont transférées en parallèle. Il est courant d'utiliser 4 ou 8 lignes de données. Utiliser une connexion à 4 fils permet d'économiser les broches E/S mais c'est plus lent car les données sont transférées en deux étapes. L'un des standards les plus utilisés pour les écrans LCD parallèles est le HD44780.

La programmation d'un écran LCD parallèle est généralement une tâche complexe et nécessite l'état du fonctionnement interne des contrôleurs LCD, y compris les diagrammes de synchronisation. Heureusement, la plupart des langages de haut niveau fournissent des commandes de bibliothèque spéciales pour afficher les données sur les LCD alphanumériques et graphiques. Tout ce que l'utilisateur a à faire est de connecter l'écran LCD au microcontrôleur, de définir la connexion de l'écran LCD dans le logiciel, puis d'envoyer des commandes spéciales pour afficher les données sur l'écran LCD.

Les LCD série sont connectés au microcontrôleur en utilisant seulement quelques lignes de données. Les écrans LCD série sont beaucoup plus faciles à utiliser, mais ils coûtent plus cher que les écrans parallèles. L'un des écrans LCD série les plus utilisés utilise le ²Protocole de bus C où seuls deux fils sont nécessaires pour la communication, à savoir SDA (données) et SCL (horloge). Il y a un périphérique maître sur le bus et il peut y avoir un certain nombre de périphériques esclaves où chaque périphérique a une adresse unique afin qu'il puisse être identifié. La figure 6.1 montre un ²Bus C avec un maître et deux dispositifs esclaves. Notez que le bus doit être relié à l'alimentation par des résistances de traction. Les lecteurs intéressés devraient pouvoir trouver de nombreux articles, notes d'application et fiches techniques sur Internet à l'adresse suivante²C. Dans ce chapitre, nous utiliserons un ²LCD basé sur C dans les projets Raspberry Pi basés sur Node-RED.

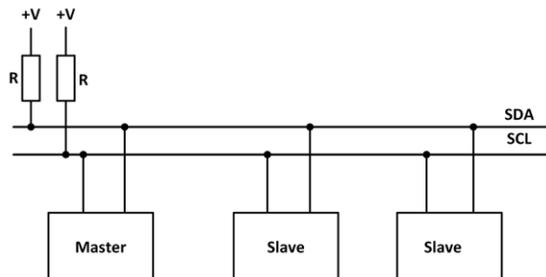


Figure 6.1 I²C Structure de bus avec un maître et 2 dispositifs esclaves

6 . 2 Raspberry Pi I²Ports C

Raspberry Pi fournit deux ports I2C SCL0 et SCL1 sur les broches GPIO suivantes :

I ² Nom du C	Broche de port GPIO	Signal
SCL0	GPIO 0	SDA0
SCL0	GPIO 1	SCL0
SCL1	GPIO 2	SDA1
SCL1	GPIO 3	SDA2

Il est recommandé d'utiliser SCL1 dans I²Applications basées sur C. Les deux lignes I2C sont alimentées en interne à +3,3V avec des résistances 1,8K.

Le I²Le C sur votre Raspberry Pi 4 doit être activé à l'aide de l'utilitaire raspi-config avant de pouvoir l'utiliser. Les étapes suivantes sont à suivre :

- Lancez raspi-config sur votre Raspberry Pi 4 en saisissant la commande suivante

```
pi@framberrypi :~$ sudo raspi-config
```

- Descendre à l'élément **5 Options d'interfaçage**
- Descendre à **P5 I2C** (voir la figure 6.2) et appuyer sur Entrée
- Sélectionner **Oui** Pour permettre le I²Interface C, et cliquez sur **D'accord** et quitter l'utilitaire raspi-config

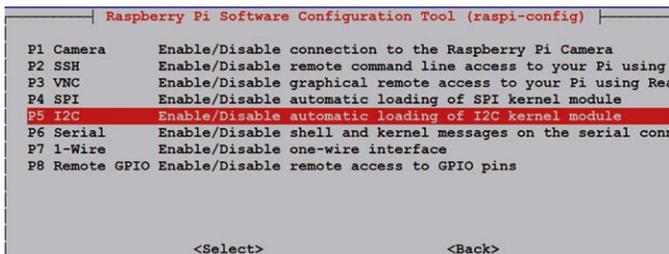


Figure 6.2 Déplacement à P5 I²C

Certains des I²Les appareils C fonctionnent avec +5 V. Il est important de faire attention lors du branchement de tels appareils à votre Raspberry Pi car la tension aux broches I/O du Raspberry Pi ne doit pas dépasser +3.3V, sinon le Raspberry Pi peut être endommagé. Il est sûr de connecter un I²C appareil qui fonctionne avec +5V à votre Raspberry Pi si l'appareil n'a pas de résistances internes pull-up sur son I²Lignes C. Si le I²Les lignes C de l'appareil sont connectées en interne à +5V, vous ne devez pas connecter un tel appareil à votre Raspberry Pi sans utiliser des circuits convertisseurs de niveau de tension pour abaisser la tension de +5V à +3.3V.

6.3 I²C LCD

Les données disponibles²C LCD se compose d'un char compatible HD44780 standard 16x2- Un écran tactile fonctionnant en mode 4 bits. Une petite carte est branchée à l'arrière de la carte LCD. Cette carte a une puce PCF8574 qui convertit les données²Données série C en forme parallèle pour l'affichage LCD. Bien que l'adresse esclave de l'écran LCD puisse être changée de 0x20 à 0x27, sa valeur par défaut est 0x27. Le module LCD fonctionne avec +5V et il n'a pas de résistances internes²Lignes C. Le module peut donc être directement connecté au Raspberry Pi I²Lignes C.

Pour tester le module LCD et trouver son adresse esclave, suivre les étapes ci-dessous :

- Connecter les broches SDA et SCL des modules LCD aux broches de port GPIO 2 et GPIO 3 du Raspberry Pi respectivement
- Connecter les broches Vcc et GND du module LCD aux broches +5V et GND du Raspberry Pi respectivement
- Entrez la commande suivante sur votre Raspberry Pi :

```
pi@framberrypi :~ $ sudo i2cdetect -y 1
```

- Vous devriez obtenir un affichage similaire à celui de la figure 6.3. Notez dans cette figure que l'adresse esclave de votre module LCD est 0x27

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  27  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

Figure 6.3 L'adresse du module LCD est 0x27

- Si l'adresse du module LCD n'est pas affichée, cela signifie qu'il y a des erreurs. Vous devez vérifier vos connexions et réessayer. Vous devez peut-être aussi installer le I²Outils C en saisissant les commandes suivantes :

```
pi@framberrypi :~ $ sudo apt-get install python-smbus
pi@framberrypi :~ $ sudo apt-get install i2c-tools
```

6.4 Installation du I²Logiciel LCD C sur Node-RED

Il y a plusieurs I²Dispositifs LCD C. Celui que nous utiliserons dans ce projet est le HD44780 com- Affichage de 16 caractères sur 2 lignes, avec une petite carte à l'arrière et une puce PCF8574 (voir la figure 6.4).

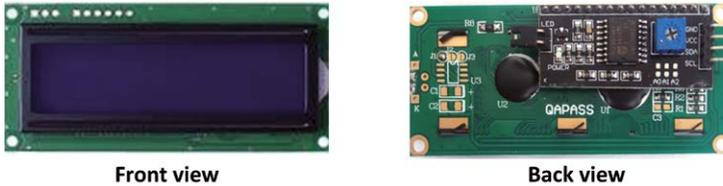


Figure 6.4 I²C LCD utilisé dans le projet

Un nœud LCD I2C compatible doit être installé pour qu'il soit disponible dans la palette Node-RED. Les étapes d'installation sont les suivantes :

- Démarrer le noeud-RED et cliquer **Menu -> Gérer la palette** puis cliquer **Installer**
- Entrer **node-red-contrib-lcd20x4-i2c** dans le chemin de recherche et cliquer **installer**
- Vous devriez maintenant voir un nouveau nœud appelé **LCD20x4 I2C**

Le nœud LCD peut être utilisé pour des écrans LCD ayant jusqu'à 4 lignes, et il accepte un objet **msg . payload. msg**. Si l'objet passé ne contient pas 4 lignes, la différence est remplie de lignes vides. Les points suivants sont importants lors de l'utilisation de ce nœud LCD :

- **msg** doit être une chaîne de texte ne dépassant pas 20 caractères. Le défilement se fera si plus de 20 caractères sont envoyés à l'écran LCD.
- A **position** l'objet peut être spécifié entre 1 et 20 (la valeur par défaut est 1 si non spécifiée) qui est utilisé pour compenser le texte (des espaces peuvent être insérés au lieu d'utiliser cet objet)
- A **centre** l'objet est optionnel et il doit être spécifié comme une valeur booléenne, passée sous forme de chaîne

Si une erreur s'est produite dans les données envoyées à l'écran LCD, un message d'erreur s'affiche sur l'écran LCD.

Un exemple est donné ci-dessous pour montrer comment le nœud LCD peut être utilisé.

Exemple

Développer un programme de flux pour afficher le texte **Bonjour de** dans la rangée du haut et **Mon écran LCD** dans la deuxième rangée.

Solution

Le programme de débit requis est indiqué à la figure 6.5.



Figure 6.5 Programme de flux de l'exemple

Les étapes sont les suivantes :

- Créer un **injecter** node et le nommer **Cliquez**
- Créer un **fonction** node et le nommer **Affichage**. Entrer les instructions suivantes dans ce nœud (voir la figure 6.6).

```
msg.payload={msgs:[
    {msg:"Hello From"},
    {msg:"My LCD"}
]};
retourner le message;
```

Le message de la première partie de l'énoncé joint à {} est envoyé à la rangée supérieure, et le deuxième message joint à {} est envoyé à la rangée inférieure du LCD.



Figure 6.6 Contenu du nœud de fonction

- Cliquez, faites glisser et déposez le **LCD20x4 I2C** Modifier le nom du nœud de l'espace de travail **CLS** Définir l'adresse I2C sur 0x27 et cliquer **Terminé**
- Rejoindre les nœuds comme indiqué dans la figure 6.5, et cliquer **Déployer**. Le texte affiché devrait être celui de la figure 6.7

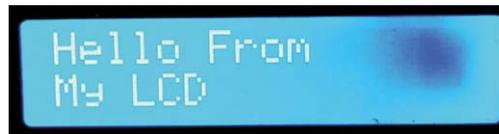


Figure 6.7 Texte affiché sur l'écran LCD

Le format d'objet LCD pour un écran LCD à deux rangées (16 x 2) est le suivant (voir lien : <https://flows.nodered.org/node/node-red-contrib-lcd20x4-i2c>) :

```
msg.payload = {
  msgs : [
    {
      msg : "string",
```

```

        pos : nombre, centre :
        "boolean"
    }
    {
        msg : "string", pos :
        number, center :
        "boolean"
    }
    ]
};

```

Les données à afficher sont saisies après le mot-clé **msg**. Le premier message est pour la rangée supérieure et le deuxième message est pour la rangée inférieure de l'écran LCD.

6 . 5 Projet 23 – Affichage de l'heure actuelle sur le LCD

Description : Dans ce projet, l'heure actuelle s'affiche sur l'écran LCD toutes les secondes au format suivant :

```

Heure actuelle
Heure=hh:mm:ss

```

Viser : Le but de ce projet est de montrer comment l'heure actuelle peut être extraite et affichée sur l'écran LCD à chaque seconde.

Schéma de circuit : La figure 6.8 montre le schéma du projet. L'écran LCD est directement relié au²Les broches C du Raspberry Pi.

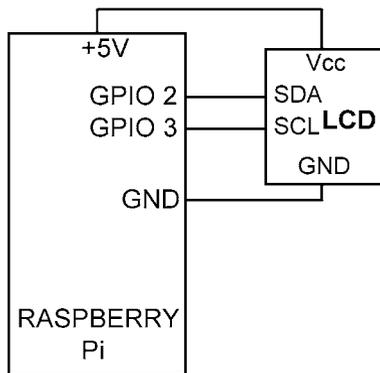


Figure 6.8 Schéma du projet

Programme de flux Node-RED : La figure 6.9 montre le programme de flux qui se compose de 3 nœuds : **injecter** nœud, un **fonction** nœud et un **LCD20x4 I2C** nœud.



Figure 6.9 Schéma du projet

Les étapes sont les suivantes :

- Créer un **injecter** noeud nommé **Cliquez**, et définissez son intervalle à une seconde
- Créer un **fonction** noeud nommé **Affichage**, et entrez les commandes suivantes (voir la figure 6.10), cliquez sur **Terminé**. Variable **LocalTime** extrait l'heure locale actuelle. Variable **Tim** construit la chaîne Time=hh:mm:ss. La première ligne de l'écran LCD affiche le texte **Heure actuelle**, et la deuxième ligne affiche l'heure réelle :

```
var LocalTime = new Date();
var Tim = "Time=" + LocalTime.toLocaleTimeString();
msg.payload={msgs:[
    {msg:"Current Time"},
    {msg:Tim, pos:1}
]};
```

retourner le message;

```
1 var LocalTime = new Date();
2 var Tim = "Time=" + LocalTime.toLocaleTimeString();
3 msg.payload={msgs:[
4     {msg:"Current Time"},
5     {msg:Tim, pos:1}
6 ]};
7
8 return msg;
9
```

Figure 6.10 Configuration du nœud de fonction

- Créer un **LCD20x4 I2C** noeud nommé **CLS** et définissez l'adresse I2C sur 0x27
- Rejoindre les 3 nœuds comme indiqué dans la figure 6.9, cliquez sur **Déployer**. Cliquez sur le bouton de la **dans- jeter** noeud. Vous devriez voir l'heure affichée sur l'écran LCD toutes les secondes comme indiqué dans la figure 6.11



Figure 6.11 Affichage de l'heure toutes les secondes

6 . 6 Projet 24 – Affichage de la température et de l'humidité locales sur l'écran LCD

Description : Dans ce projet, la prévision météorologique actuelle est reçue et le tempérament local- La température et l'humidité sont affichées sur l'écran LCD toutes les 10 secondes.

Viser : Le but de ce projet est de montrer comment la température et l'humidité locales peuvent être affichées sur l'écran LCD.

Schéma de circuit : Le schéma de circuit du projet est identique à celui de la figure 6.8.

Programme de flux Node-RED : La figure 6.12 montre le programme de flux qui se compose de 4 nœuds : **injecter** nœud pour demander une lecture toutes les 10 secondes, un **openweathermap** node pour obtenir la carte météorologique locale, un **fonction** nœud pour extraire la température et l'humidité locales, et un **LCD20x4 I2C** nœud pour afficher la température et l'humidité.



Figure 6.12 Schéma du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud avec un intervalle de 10 secondes et le nommer comme **Cliquez**
- Créer un **openweathermap** node et le nommer **Temp & Hum**. Définissez la clé API et définissez les noms de votre ville et pays locaux. Cliquez **Terminé**
- Créer un **LCD20x4 I2C** node et le nommer **CLS**. Définir l'adresse I2C sur 0x27 comme précédemment et cliquer **Terminé**
- Rejoindre les 4 nœuds et cliquer **Déployer**.
- Cliquez sur le bouton de la **injecter** Le noeud. Vous devriez voir la température et l'humidité locales affichées sur l'écran LCD, comme indiqué à la figure 6.13.



Figure 6.13 Affichage de la température et de l'humidité locales

6 . 7 Project 25 – Affichage des nombres de dés

Description : Dans ce projet, deux nombres aléatoires sont générés entre 1 et 6 et sont affichés sur l'écran LCD chaque fois que le bouton du nœud inject est cliqué. Les numéros sont affichés pendant 5 secondes. Après ce temps, l'affichage est **Prêt** est affiché. À ce stade, l'utilisateur peut cliquer sur le bouton du nœud inject pour générer deux nouveaux

Nombres aléatoires. Ce projet peut être utilisé dans les jeux joués avec des dés.

Viser : Le but de ce projet est de montrer comment deux nombres aléatoires peuvent être générés et ensuite combinés et affichés sur l'écran LCD.

Schéma de circuit : Le schéma de circuit du projet est identique à celui de la figure 6.8.

Programme de flux Node-RED : La figure 6.14 montre le programme de flux qui se compose de 8 nœuds : **injecter** nœud pour commencer à générer deux nombres aléatoires, 2 **aléatoire** nœuds pour générer deux nombres aléatoires entre 1 et 6 **rejoindre** nœud pour joindre les deux nombres générés, un **fonction** nœud pour formater les nombres à afficher sur l'écran LCD, un **retard** Node pour générer un délai de 5 secondes afin que le message Ready puisse être affiché après 5 secondes, et **fonction** nœud qui affiche réellement le message **Prêt**.

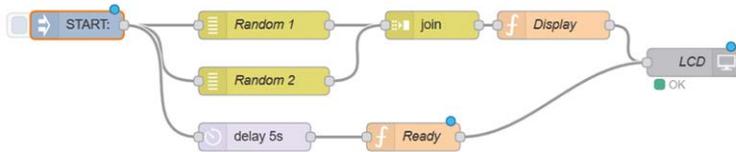


Figure 6.14 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** node et le nommer **Démarrer**
- Créer deux **aléatoire** Les nœuds et nommez-les comme **Aléatoire 1** et **Aléatoire 2**. Définir les nœuds pour générer des nombres aléatoires entre 1 et 6, cliquez sur **Terminé**
- Créer un **rejoindre** et le configurer comme indiqué sur la Figure 6.15. Ce nœud joindra les deux nombres aléatoires dans un tableau. Cliquez **Terminé**.

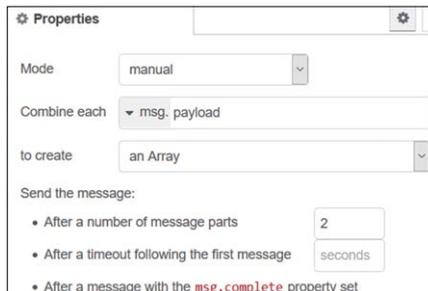


Figure 6.15 Configuration du nœud de jointure

- Créer un **fonction** node et le nommer **Affichage**. Entrez les instructions suivantes dans ce nœud (voir la figure 6.16). Ces instructions extraient les deux nombres du tableau et les combinent en une seule variable appelée **disp**. Le contenu de **disp** sont affichés sur la première rangée de l'écran LCD. Cliquez sur Terminé.

```
var first = msg.payload[0];
var second = msg.payload[1];
var disp = first + " " + second;
msg.payload={msgs:[
    {msg:disp} ]};
```

retourner le message;

- Créer un **LCD20x4 I2C** node et le nommer **CLS**. Définir l'adresse I2C sur 0x27 comme précédemment et cliquer **Terminé**
- Créer un fichier de 5 secondes **retard** noeud
- Créer un **fonction** node et le nommer **Prêt**. Entrez les instructions suivantes dans ce nœud. Ce nœud affichera le message **Prêt** sur l'écran LCD :

```
msg.payload={msgs:[
    {msg:"Ready"} ]};
```

retourner le message;

- Rejoindre tous les nœuds comme indiqué sur la figure 6.14 et cliquer **Déployer**.
- Cliquez sur le bouton de la **injecter** node. Vous devriez voir deux nombres générés-tentre 1 et 6. Après 5 secondes, l'affichage se vide et un message **Prêt** sera affiché de sorte que si vous cliquez à nouveau sur le bouton, un nouvel ensemble de nombres est généré. La figure 6.16 montre un exemple de sortie sur l'écran LCD.



Figure 6.16 Exemple de sortie sur l'écran LCD

6 8Projet 26 – Comptoir d'événements avec LCD

Description : Il s'agit d'un projet de comptage des événements. Les événements sont supposés se produire lorsque la broche de port GPIO 15 du Raspberry Pi passe de logique 1 à logique 0. Dans le projet, les événements sont simulés en connectant un commutateur à bouton-poussoir au GPIO 15 et en appuyant sur le bouton. Le programme compte les événements qui se produisent et affiche le total à tout moment sur l'écran LCD.

Viser : Le but de ce projet est de montrer comment un simple programme de comptage d'événements peut être développé- opé à l'aide de Node-RED.

Schéma de circuit : Les broches à deux boutons sont connectés au GPIO 15 et à la terre respectivement.

Programme de flux Node-RED : La figure 6.17 montre le programme de flux qui se compose de 3 nœuds : a **rpi gpio dans** nœud pour recevoir les entrées de bouton, un **fonction** le nœud pour mettre à jour les comptes, et un **LCD20x4 I2C** nœud pour afficher le nombre.



Figure 6.17 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **rpi gpio dans** nœud et le nommer **Bouton**. Définir le numéro de broche GPIO 15, définir **Résistance** à pullup et **Débounce** à 25ms. Notez que **Débounce** est réglé sur 25 ms pour éliminer les problèmes de rebond des contacts associés aux commutateurs mécaniques, comme décrit au chapitre 5. Cliquer pour lire l'état initial de la broche. Cliquez **Terminé**
- Créer un **fonction** node et le nommer **Affichage**. Entrez les instructions suivantes dans ce nœud (voir la figure 6.18). Notez que nous utilisons une variable de contexte ici. L'instruction :

```
var count=context.get('count') |
```

signifie que si **compte** n'existe pas dans l'objet contexte, puis faire variable **compte** zéro, sinon assigner la valeur stockée à **compte**. La valeur de **compte** est dans- crémé à chaque pression sur le bouton (c.-à-d. msg.payload passe à 0). **Compte** est ensuite converti en chaîne et envoyé à l'écran LCD :

```
var count=context.get('count') || 0;
si(msg.payload == 0)
{
    compte ++;
    context.set('count',count);
    var cnt = "Events = " + count.toString();
    msg.payload={msgs:[
        {msg:cnt} ]};

    retourner le message;
}
else
renvoie null;
```

```

var count=context.get('count') || 0;

if(msg.payload == 0)
{
  count++;
  context.set('count',count);
  var cnt = "Events = " + count.toString();
  msg.payload={msgs:[
    {msg:cnt}
  ]};
  return msg;
}
else
  return null;

```

Figure 6.18 Configurer le nœud de fonction

- Créer un **LCD20x4 I2C** nœud et le nommer **CLS**. Définir l'adresse I2C sur 0x27 et cliquer **Terminé**
- Rejoindre tous les nœuds comme dans la figure 6.17 et cliquer **Déployer**. Chaque fois que le bouton est enfoncé, le nombre d'événements augmente de 1 et la valeur totale s'affiche à l'écran comme indiqué sur la figure 6.19.

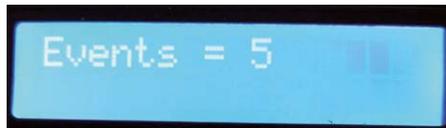


Figure 6.19 Affichage des événements totaux à tout moment

6 9Projet 27 – Capteur de température et d'humidité DHT11 avec écran LCD

Description : Dans ce projet, nous utilisons le capteur numérique DHT11 populaire pour mesurer l'am- Température et humidité de l'air ambiants et les afficher sur l'écran LCD toutes les 5 secondes.

Viser : Le but de ce projet est de montrer comment le capteur de température et d'humidité DHT11 peut être utilisé dans un projet Node-RED.

Schéma fonctionnel : La figure 6.20 montre le schéma du projet.



Figure 6.20 Schéma du projet

Informations générales : DHT 11 (ou DHT22) est l'un des capteurs de température et d'humidité numériques les plus populaires et les plus utilisés. Le capteur est essentiellement à 3 broches, comme indiqué sur la figure 6.21. Bien que certaines versions de capteurs soient à 4 broches, une des broches n'est pas utilisée. Les

DHT11 est composé de deux parties : un capteur d'humidité capacitif et un capteur de température à thermistance. Parce que la sortie est numérique, la puce comprend également un ADC pour convertir les signaux analogiques en numériques.

Les spécifications de base du DHT11 sont :

- Alimentation 3V à 5V
- Faible coût
- 2,5 mA courant max (pendant la conversion)
- 20 - 80% d'humidité avec une précision de 5
- Plage de température 0 - 50 °C avec une précision de 2 °C
- 1 Hz fréquence d'échantillonnage (une fois par seconde)

DHT22 est similaire à DHT11, il a la même taille et forme, mais il offre des plages de mesure plus larges et est également plus précis que le DHT11. Dans ce projet, nous utiliserons un DHT11.

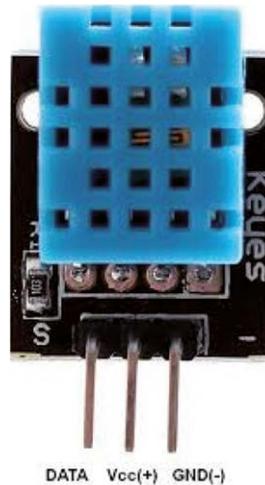


Figure 6.21 Le capteur DHT11

Installation du nœud DHTxx Node-RED : Les étapes d'installation du nœud RED DHTxx (c.-à-d. DHT11 et DHT22) sont les suivantes :

- Cliquez **Menu -> Gérer palette** et cliquez **Installer**
- Entrer **node-red-contrib-dht-sensor** et cliquez **installer** (voir figure 6.22)
- Vous devriez voir un nouveau nœud appelé **rpi-dht22** dans la palette Node

Nous sommes maintenant prêts à développer notre programme de flux pour lire et afficher la température ambiante et l'humidité.



Figure 6.22 Installation du nœud DHTxx

Schéma de circuit : Le schéma du projet est présenté à la figure 6.23. Le L'écran LCD de la console C est connecté au Raspberry Pi comme dans les projets précédents de ce chapitre. Le DHT11 est connecté au Raspberry Pi comme suit :

Broche DHT11	Raspberry Pi Pin
Données (S)	GPIO 15
Vcc	+3,3V
Gnd (-)	GND

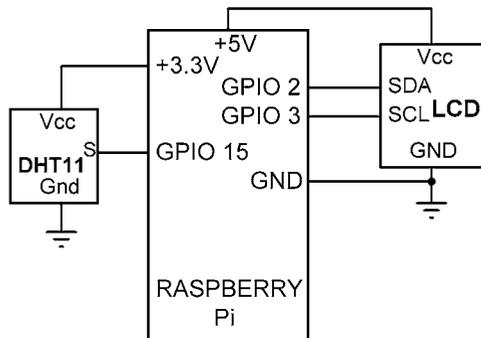


Figure 6.23 Schéma du projet

Programme de flux Node-RED : La figure 6.24 montre le programme de flux qui se compose de 3 nœuds : a **rpi gpio** dans nœud pour recevoir les entrées de bouton, un **fonction** le nœud pour mettre à jour les comptes, et un **LCD20x4 I2C** nœud pour afficher le nombre.

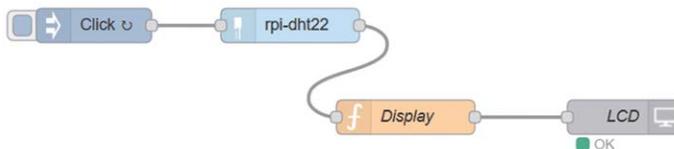


Figure 6.24 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** noeud avec le nom **Cliquez** et régler l'intervalle sur 5 secondes

- Créer un **rpi-dht22** et le configurer comme indiqué dans la figure 6.25. Définir **Sens-modèle de sor** à DHT11, **Numérotation des épingles** au GPIO BCM, et le numéro de la broche à 15 (GPIO 15), cliquez sur Terminé

The image shows the configuration interface for the 'rpi-dht22' node in Node-RED. It consists of four rows, each with a label and a dropdown menu:

- Topic:** rpi-dht22
- Sensor model:** DHT11
- Pin numbering:** BCM GPIO
- Pin number:** 15

Figure 6.25 Configuration du nœud rpi-dht22

- Créer un **fonction** node et le nommer **Affichage**. Entrez les instructions suivantes dans la fonction (voir figure 6.26). Variables **T** et **H** recevoir les lectures de température et d'humidité respectivement du DHT11 toutes les 5 secondes. Température- La température est affichée dans la rangée supérieure, tandis que l'humidité est affichée dans la rangée inférieure de l'écran LCD :

```
T = charge utile de la masse;
H = humidité relative;
Température = "T = " + T + "C";
Hum = "H = " + H + "%";
```

```
msg.payload={msg:[
    {msg : Temp},
    {msg : Hum} ]};
```

retourner le message;

```

1 T = msg.payload;
2 H = msg.humidity;
3 Temp = "T = " + T + "C";
4 Hum = "H = " + H + "%";
5
6 msg.payload={msg:[
7     {msg: Temp},
8     {msg: Hum}
9 ]};
10 return msg;
11

```

Figure 6.26 Configuration du nœud de fonction

- Créer le nœud LCD comme précédemment, joindre tous les nœuds et cliquer **Déployer**.
- Cliquez sur le bouton de la **injecter** noeud. Vous devriez voir la température et l'humidité mises à jour et affichées toutes les 5 secondes (voir figure 6.27)

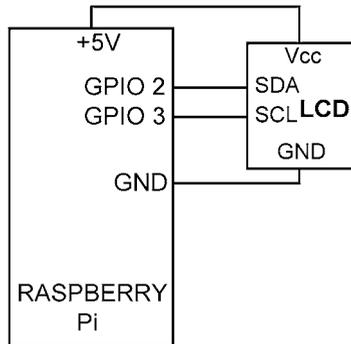


Figure 6.27 Affichage de la température et de l'humidité

Notez que si vous connectez un **déboguer** node à la sortie du nœud rpi-dht22, vous ne verrez que la température affichée dans la charge utile. Si, cependant, vous définissez la Sortie du nœud de débogage sur **objet msg complet**, vous verrez la température et l'humidité dis- Joué comme indiqué à la figure 6.28

```
12/12/2019, 09:44:08 node: a69647b5.0037e8
rpi-dht22 : msg : Object
  ▶ { topic: "rpi-dht22", payload:
    "15.00", _msgid: "23c59261.47d7ce",
    humidity: "23.00", isValid: true ... }
```

Figure 6.28 Sortie du noeud de débogage

La figure 6.29 montre le circuit construit sur une planche à pain et connecté au Raspberry Pi à l'aide de fils de cavaliers.

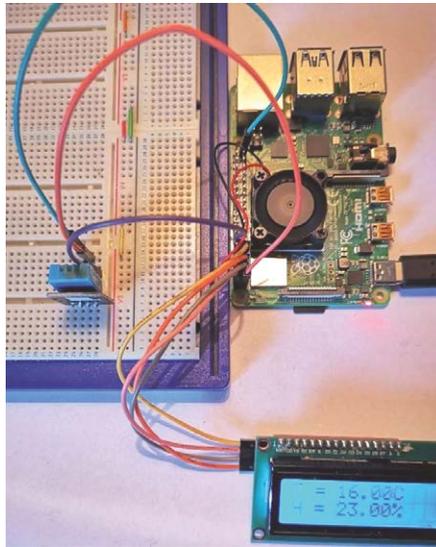


Figure 6.29 Circuit construit sur une planche à pain

6 10Projet 28 – Capteur de distance à ultrasons avec écran LCD

Description : Dans ce projet, un module de capteur de distance à ultrasons est utilisé et la distance par rapport à tout obstacle devant le capteur est affichée sur l'écran LCD.

Viser : Le but de ce projet est de montrer comment un capteur ultrasonique peut être utilisé dans un projet Node-RED.

Renseignements généraux : Les modules de capteurs de distance ultrasoniques sont utilisés dans des applications où il est nécessaire de mesurer la distance à un objet (p.ex. un obstacle) devant le module de capteur. Le capteur à ultrasons le plus utilisé est le HC-SR04 (voir la figure 6.30). Il s'agit d'un module à 4 broches avec les broches suivantes :

Echo : broche d'écho

Trig : goupille de déclenchement

Vcc : Broche d'alimentation (+5V) **GND** :

broche de mise à la terre de l'alimentation



Figure 6.30 Le module de capteur de distance à ultrasons HC-SR04

La plage de mesure du capteur HC-SR04 est de 2cm à 400cm. Le principe de fonctionnement du module capteur ultrasonique est le suivant (voir figure 6.31) :

-
- Le module envoie ensuite huit signaux d'ondes carrées 40kHz à la cible et définit la broche d'écho HAUTE
- Le programme démarre une minuterie
- Le signal atteint la cible et fait écho au module
- Lorsque le signal est renvoyé au module, la broche d'écho devient LOW
- Le minuteur est arrêté
- La durée du signal d'écho est calculée et elle est proportionnelle à la distance par rapport à la cible

La distance par rapport à l'objet est calculée comme suit :

$$\text{Distance de l'objet (en mètres)} = (\text{durée du temps d'écho en secondes} * \text{vitesse du son}) / 2$$

Par conséquent,

ou

Par exemple, si la durée du signal d'écho est de 294 microsecondes, la distance à l'objet est calculée comme suit :

$$\text{Distance de l'objet (cm)} = 294 * 0,017 = 5 \text{ cm}$$

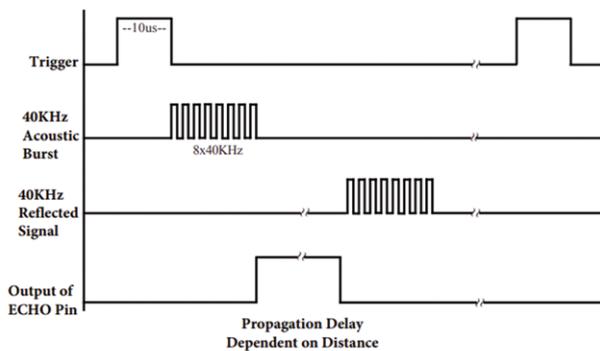


Figure 6.31 Fonctionnement du module de détection ultrasonique

Installation du capteur ultrasonique Node-RED : Les étapes d'installation du nœud HC-SR04 de capteur RED sont les suivantes :

- Cliquez **Menu -> Gérer** palette et cliquez **Installer**
- Entrer **noeud-rouge-noeud-pisrf** et cliquez **installer**
- Vous devriez voir un nouveau nœud appelé **rpi srf** dans la palette Node
- Nous sommes maintenant prêts à développer notre programme de flux pour lire et afficher la distance

Schéma de principe : Le schéma du projet est présenté à la figure 6.32.



Figure 6.32 Schéma du projet

Schéma de circuit : La figure 6.33 montre le schéma de circuit. Les broches de port Raspberry Pi GPIO 15 et GPIO 18 sont connectées aux broches Trig et Echo du capteur respectivement. Notez que le capteur fonctionne avec +5V et que son niveau de tension de sortie Echo monte jusqu'à +5V. Ceci est trop élevé pour les entrées du Raspberry Pi et, par conséquent, un circuit de diviseur de potentiel résistif est utilisé pour abaisser la tension de sortie du capteur à +3,3V. Les broches VCC et GND du capteur sont connectées à +5V et aux broches GND de Raspberry Pi respectivement.

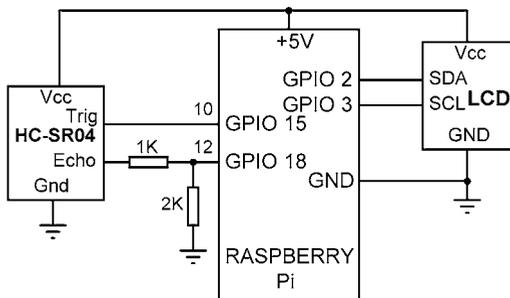


Figure 6.33 Schéma du projet

Programme de flux Node-RED : La figure 6.34 montre le programme de flux qui se compose de 3 nœuds : a **rpi srf** nœud pour lire la distance du capteur, un **fonction** nœud pour formater les données à afficher, et un **CLS** nœud.



Figure 6.34 Programme de flux du projet

Les étapes sont les suivantes :

- Cliquez, faites glisser et déposez le **rpi srf** nœud dans l'espace de travail. Configurez-le comme indiqué à la figure 6.35. Notez que par défaut la distance est mesurée toutes les 0,5 secondes, mais cela peut être modifié si vous le souhaitez. Définir les numéros de broches à l'endroit où les broches Trig et Echo sont connectées sur le Raspberry Pi. Il est important d'être prudent ici puisque les numéros de broche requis ici sont les numéros de broches physiques de l'en-tête et non les noms de broches GPIO. Dans ce projet, les broches Trig et Echo sont connectées aux broches GPIO 15 et 18 respectivement, qui correspondent aux broches physiques 10 et 12 respectivement.

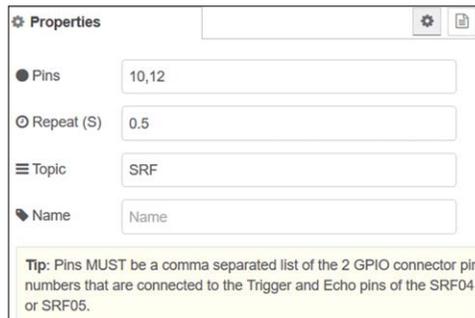


Figure 6.35 Configuration du nœud rpi srf

- Créer un **fonction** le noe et le crabe comme **Distance**. Entrez l'état suivant- Les valeurs de l'IRM dans ce nœud (voir la figure 6.36). Ici, variable **Distance** est chargé avec la distance mesurée réelle en centimètres. La première rangée de l'écran LCD affiche un en-tête, tandis que la deuxième rangée affiche la distance mesurée. Cliquez sur **Terminé** :

```
var Distance = msg.payload;
msg.payload={msgs:[
    {msg:"Distance      (cm)"},
    {msg : Distance}
  ]};

retourner le message;
```

```

Name Distance
Function
1 var Distance = msg.payload;
2 * msg.payload={msg:[
3   {msg:"Distance (cm)"},
4   {msg: Distance}
5   ]};
6 return msg;
7

```

Figure 6.36 Configuration du nœud de fonction

- Créer le nœud LCD comme précédemment, joindre les 3 nœuds et cliquer **Déployer**. Vous devriez voir la distance affichée sur l'écran LCD. Placez quelque chose devant le capteur et vous devriez voir la distance changer (voir la figure 6.37)



Figure 6.37 Affichage de la distance

La figure 6.38 montre le circuit construit sur une planche à pain (notez dans cette figure que l'auteur n'avait pas de résistances 2K et utilisait des résistances 2x1K à la place).

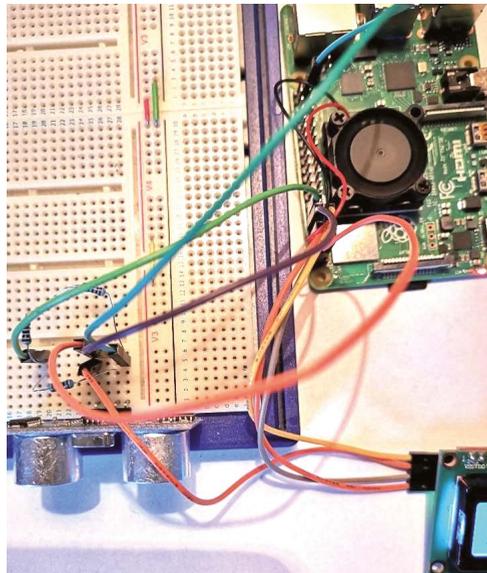


Figure 6.38 Le circuit construit sur une planche à pain

6 . 11 Projet 29 – Alarme de distance à ultrasons avec buzzer

Description : Dans ce projet, un module de capteur de distance à ultrasons est utilisé, comme dans le- Tout projet. De plus, un buzzer actif est utilisé de sorte que si la distance à tout obstacle est inférieure à 20 cm, le buzzer est activé. Ce projet peut être utilisé dans des applications robotiques mobiles ou pour aider à stationner votre voiture.

Viser : Le but de ce projet est de montrer comment un capteur ultrasonique peut être utilisé avec un buzzer dans un projet Node-RED.

Schéma de principe : La figure 6.39 montre le schéma du projet.



Figure 6.39 Schéma du projet

Schéma de circuit : Le schéma de circuit du projet est représenté à la figure 6.40, où un buzzer actif est connecté au port pin GPIO 23. Le reste du circuit est identique à celui de la figure 6.33.

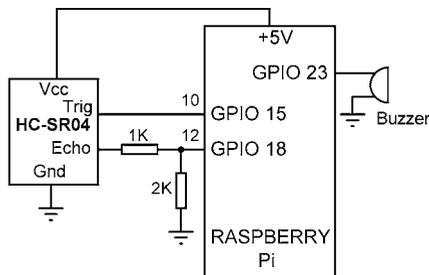


Figure 6.40 Schéma du projet

Programme de flux Node-RED : La figure 6.41 montre le programme de flux qui se compose de 3 noeuds : a **rpi srf** nœud pour lire la distance du capteur, un **fonction** nœud pour formater les données afin de contrôler le buzzer, et **rpio gpio out** le nœud qui est connecté au buzzer.



Figure 6.41 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **rpi srf** nœud comme dans le projet précédent, mais définissez l'intervalle de répétition à une valeur inférieure, p. ex., 0,25 seconde
- Créer un **fonction** Si la distance par rapport à un obstacle est inférieure à 20 cm, la sortie est réglée sur 1 pour activer le buzzer :

```

var Distance = msg.payload;
if(Distance < 20)
  msg.payload = 1;
sinon
  msg.payload = 0;
renvoie msg;

```

- Créer un **rpi gpio out** nœud et le nommer **Buzzer**. Régler le **Pin** à GPIO 23, et initialiser l'état de la broche à 0
- Joindre les 3 nœuds, et cliquer **Déployer**. Placez un objet près du module ultrasonique et vous entendrez le bourdonnement. Quand l'objet est enlevé, le bourdonnement devrait cesser de sonner.

6 . 12 Projet 30 – Système de stationnement à ultrasons avec buzzer

Description : Dans ce projet, on suppose que deux modules de capteurs de distance ultrasoniques sont fixés à l'avant et à l'arrière d'un véhicule. De plus, un buzzer actif est utilisé de sorte que si la distance à tout obstacle à l'avant ou à l'arrière du véhicule est inférieure à 20 cm, le buzzer est activé. Ce projet peut être utilisé pour aider à stationner votre voiture.

Viser : Le but de ce projet est de montrer comment deux modules de capteurs de distance ultrasoniques peuvent être utilisés avec un buzzer dans un projet Node-RED.

Schéma de principe : La figure 6.42 montre le schéma du projet.



Figure 6.42 Schéma du projet

Schéma de circuit : Le schéma du projet est présenté à la figure 6.43. Un des modules de capteur ultrasonique et le buzzer sont connectés comme dans le projet précédent. Les broches Trig et Echo du deuxième module de capteur ultrasonique sont connectées aux broches de port GPIO 24 et GPIO 25 respectivement (broches physiques 18 et 22) du Raspberry Pi.

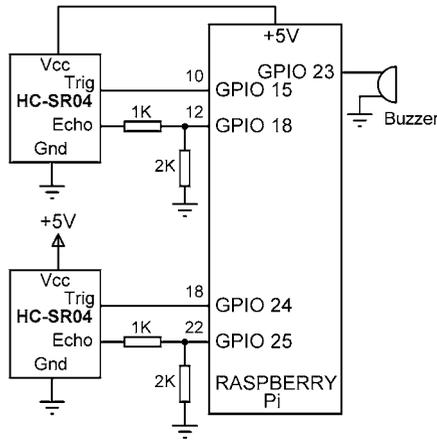


Figure 6.43 Schéma du projet

Programme de flux Node-RED : La figure 6.44 montre le programme de flux qui se compose de 5 nœuds : 2 **rpi srf** nœuds, un pour chaque module de capteur de distance à ultrasons **rejoindre** nœud pour joindre les deux sorties dans un tableau, une **fonction** nœud pour formater la sortie du buzzer, et un **rpi gpio out** nœud pour contrôler le buzzer.

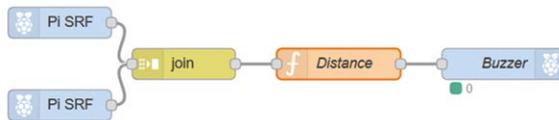


Figure 6.44 Programme de flux du projet

Les étapes sont les suivantes :

- Créer les deux **rpi srf** et réglez leurs broches sur 10,12 et 18,22. Définissez la **Répéter** fois à 0,25 seconde dans chacune
- Créer un **rejoindre** node et configurer comme indiqué dans la Figure 6.45. Ce node joint les deux entrées dans un tableau qui peut être manipulé dans le noeud de fonction

Figure 6.45 Configuration du nœud de jointure

- Créer un **fonction** node et entrez les instructions suivantes dans ce node, cliquez sur **Terminé** :

```
var Distance1 = msg.payload[0];
var Distance2 = msg.payload[1]; if(Distance1 <
20 || Distance2 < 20)
    msg.payload = 1;
autrement
    msg.payload = 0;
retourner le message;
```

- Créer un **rpi gpio out** nœud et définir le numéro de broche sur GPIO 23
- Rejoindre tous les nœuds comme dans la figure 6.44, cliquez sur **Déployer**. Placez les objets à proximité de l'un ou l'autre des capteurs et vous devriez entendre le buzzer.

La figure 6.46 montre le circuit construit sur une planche à pain prête pour les essais.

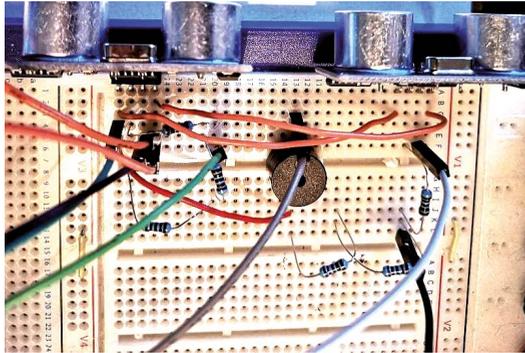


Figure 6.46 Circuit construit sur une planche à pain

6 . 13 Utilisation d'un écran LCD parallèle

Dans les projets précédents de ce chapitre, nous avons utilisé l²Écrans LCD basés sur C. Dans ces projets, l'écran LCD était un écran parallèle de type HD44780, mais une petite carte a été branchée à l'arrière de l'écran LCD afin que celui-ci puisse communiquer en utilisant les SDA et SCL l²Lignes C. Parallèle LCDs ont les avantages qu'ils sont beaucoup moins cher que le l²Versions C. Leur inconvénient est que leur contrôle nécessite au moins 6 broches, au lieu de 2 broches nécessaires pour la l²Versions C. Les écrans LCD parallèles nécessitent également un potentiomètre externe pour régler leur contraste.

Dans les parties restantes de ce chapitre, nous verrons comment un LCD parallèle de type HD44780 peut être utilisé avec des projets Node-RED.

Module LCD HD44780

HD44780 est l'un des modules LCD alphanumériques les plus populaires utilisés dans l'industrie et aussi par les amateurs. Ce module est monochrome et existe en différentes tailles. Des modules de 8, 16, 20, 24, 32 et 40 colonnes sont disponibles. Selon le modèle choisi, le nombre de rangées varie entre 1, 2 ou 4. L'écran fournit un connecteur 14 broches (ou 16 broches) à un microcontrôleur. Le tableau 6.1 présente la configuration et les fonctions des broches d'un module LCD à 14 broches. Voici un résumé des fonctions des broches :

Pin no	Nom	Fonction
1	VSS	Ground
2	DMV	+ 5 % d'alimentation
3	VEE	Contraste
4	RS	Sélectionner l'enregistrement

5	R/W	Lecture/écriture
6	E	Activer
7	D0	Bit de données 0
8	D1	Bit de données 1
9	D2	Bit de données 2
10	D3	Bit de données 3
11	D4	Bit de données 4
12	D5	Bit de données 5
13	D6	Bit de données 6
14	D7	Bit de données 7
15	A	Anode (lumière)
16	K	Cathode (lumière)

Tableau 6.1 Configuration des broches du module LCD HD44780

VSS est l'alimentation 0V ou la terre. La broche VDD doit être connectée à l'alimentation positive. Bien que les fabricants spécifient une alimentation DC 5 V, les modules fonctionneront généralement avec une tension aussi basse que 3 V ou aussi élevée que 6 V.

La broche 3 est appelée VEE et il s'agit de la broche de contrôle du contraste. Cette broche sert à régler le- l'écran et il doit être connecté à une alimentation en tension variable. Un potentiomètre est normalement connecté entre les lignes d'alimentation électrique avec son bras d'essuie-glace relié à cette broche afin que le contraste puisse être ajusté.

La broche 4 est le Register Select (RS) et lorsque cette broche est LOW, les données transférées à l'écran sont traitées comme des commandes. Lorsque RS est HIGH, les données de caractères peuvent être transférées vers et depuis le module.

La broche 5 est la ligne de lecture/écriture (R/W). Cette broche est tirée LOW afin d'écrire des commandes ou des données de caractères sur le module LCD. Lorsque cette broche est HAUTE, les données de caractère ou le statut informent- tion peut être lue à partir du module.

La broche 6 est la broche d'activation (E) qui sert à initier le transfert de commandes ou de données- tentre le module et le microcontrôleur. Lors de l'écriture sur l'écran, les données sont transférées uniquement sur la transition HIGH vers LOW de cette ligne. Lors de la lecture à partir de l'écran, les données sont- est disponible après la transition de LOW à HIGH de la broche d'activation et ces données restent valides tant que la broche d'activation est à la logique HIGH.

Les broches 7 à 14 sont les huit lignes de bus de données (D0 à D7). Les données peuvent être transférées entre le microcontrôleur et le module LCD en utilisant soit un seul octet de 8 bits, soit deux grignotines de 4 bits. Dans ce dernier cas, seules les quatre lignes de données supérieures (D4 à D7) sont utilisées. Le mode 4 bits présente l'avantage que quatre lignes d'E/S en moins sont nécessaires pour communiquer avec l'écran LCD.

Un projet simple est présenté ci-dessous pour l'exhaustivité, qui illustre comment un écran LCD parallèle peut être utilisé dans un projet basé sur Node-RED.

6 . 14 Projet 31 – Affichage du message sur un écran LCD parallèle

Description : Dans ce projet, un écran LCD parallèle est connecté au Raspberry Pi et aux- sage Bonjour Il est affiché sur l'écran LCD.

Viser : Le but de ce projet est de montrer comment un écran LCD parallèle peut être utilisé dans un projet basé sur Node-RED.

Schéma de circuit : La figure 6.47 montre le schéma du projet. L'écran LCD est- Raspberry Pi comme suit :

Broche LCD	Pi GPIO Pin	Raspberry Pi Physican Pin no
Framboise D4	GPIO 15	10
D5	GPIO 18	12
D6	GPIO 23	16
D7	GPIO 24	18
RS	GPIO 25	22
E	GPIO 8	24
R/W	GND	6
GND	GND	20
Vcc	+5V	2

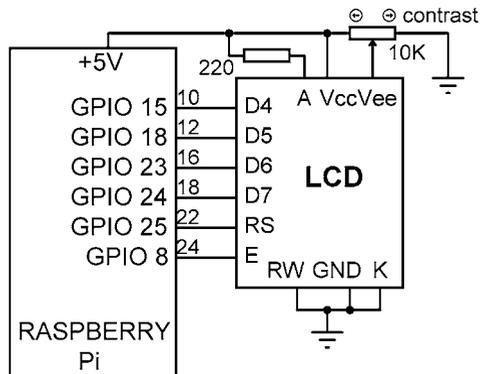


Figure 6.47 Schéma du projet

Noter qu'un potentiomètre 10K doit être connecté à la broche VEE (ou Vo) pour régler le contraste de l'écran LCD.

Installation du nœud LCD parallèle : Les étapes d'installation du nœud LCD parallèle à la palette Node-RED sont indiquées ci-dessous :

- Cliquez **Menu -> Gérer palette** et puis cliquez **Installer**
- Entrer **noeud-rouge-noeud-pilcd** dans le champ de recherche et cliquez **installer**

- Vous devriez voir un nouveau nœud nommé **rpi lcd** dans la palette Node
- Le nœud **rpi lcd** Il faut indiquer où les broches suivantes de l'écran ACL sont connectées :

RS, E, D4, D5, D6, D7

Programme de flux Node-RED : La figure 6.48 montre le programme de flux qui se compose de 4 nœuds **injecter** nœud, 2 **fonction** , où chacun envoie du texte à chaque ligne de l'écran LCD, et le **rpi lcd** nœud de contrôle.

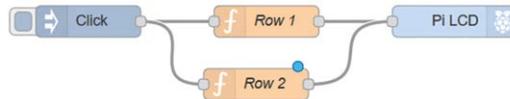


Figure 6.48 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** noeud
- Créer deux **fonction** Les nœuds et nommez-les comme **Row1** et Rangée 2. **Ces fonctions**- Les messages contiennent le texte à afficher sur l'écran LCD. La ligne 1 envoie le texte de la rangée supérieure de l'écran LCD, tandis que la ligne 2 envoie le texte de la deuxième rangée de l'écran LCD. Le contenu de Rangée 1 **fonction est** :

```
var first = "1:This is Line 1"; msg.payload  
= first;  
retourner le message;  
De même, le contenu de la fonction de la ligne 2 est :  
second="2:This is Line 2";  
msg.payload=second;  
retourner le message;
```

- Les données doivent être envoyées sous forme de chaînes au LCD. Pour la ligne 1, commencer les données par **1 :**, pour la ligne 2, commencez les données par **2 :** et ainsi de suite. String **clr** : Efface l'écran.

Les lecteurs intéressés peuvent obtenir plus d'informations en suivant le lien suivant :

<https://flows.nodered.org/node/node-red-node-pilcd>

- Créer la **rpi lcd** node. Configurer le nœud comme indiqué dans la figure 6.49.



Figure 6.49 Configuration du nœud *rpi lcd*

- Rejoindre tous les nœuds et cliquer **Déployer**. Cliquez sur le bouton de la **injecter** nœud et vous devriez voir les textes suivants affichés sur l'écran LCD :

Il s'agit de la ligne 1

Il s'agit de la ligne 2

6 . 15 Résumé

Dans ce chapitre, nous avons appris comment utiliser un écran LCD dans le Raspberry Pi pro basé sur Node-RED- jects.

Dans le prochain chapitre, nous examinerons les convertisseurs analogiques-numériques (ADC) et verrons comment nous pouvons utiliser les ADC dans des projets basés sur Node-RED.

Chapitre 7 • Utilisation de l'ADC dans les projets Raspberry Pi Node-RED

7.1 Aperçu

La plupart des capteurs dans la vie réelle ont des sorties analogiques et ces sorties ne peuvent pas être connectés directement à l'ordinateur. Les convertisseurs analogiques-numériques (ADC) sont utilisés pour convertir ces signaux analogiques en forme numérique de sorte qu'ils puissent être lus par un ordinateur. Malheureusement, un ordinateur Raspberry Pi ne possède pas de ports ADC intégrés et il n'est pas possible de l'interfacer avec des sorties analogiques à moins qu'un circuit ADC externe ne soit utilisé.

Il existe de nombreux types d'ADC qui peuvent être utilisés dans les projets. Certains noms importants des ADC - Les facteurs à prendre en compte avant d'utiliser un tel système sont les suivants :

Longueur du mot : La plupart des ADC avaient une largeur de 8 bits, avec seulement 256 niveaux de quantification. Ces DAC ne sont pas précis car seuls 256 niveaux sont utilisés pour représenter la tension d'entrée. Les DAC à usage général sont aujourd'hui généralement de 10 ou 12 bits. Un ADC 10 bits a 1024 niveaux de quantification, et un ADC 12 bits a 4096 niveaux de quantification. En considérant un ADC 8 bits, avec une tension de référence de +3.3V, chaque bit correspond à 12.89mV. Cela signifie que les tensions d'entrée inférieures à 12,89 mV ne peuvent pas être différenciées. Avec un ADC 10 bits, chaque bit représente 3,22 mV, et avec un convertisseur 12 bits, chaque bit représente 0,80 mV.

Dans ce chapitre, nous utiliserons un ADC externe sur notre Raspberry Pi pour lire les données des capteurs en utilisant Node-RED.

7.2 Le MCP3002 ADC

L'ADC que nous utiliserons dans nos projets de ce chapitre est le MCP3002. Il s'agit d'une puce DIL à 8 broches (voir la figure 7.1), qui présente les caractéristiques suivantes :

- Résolution 10 bits
- 2 canaux d'entrée
- Convertisseur d'approximation successive
- Circuit d'échantillonnage et de maintien sur puce
- Interface SPI vers l'ordinateur hôte
- +2.7 à +5.5V fonctionnement
- 200ksps taux d'échantillonnage (à +5V)
- 5nA courant de secours

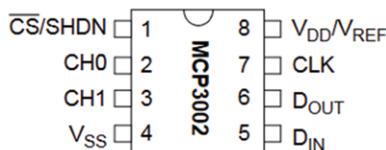


Figure 7.1 Schéma de broches MCP3002

Le MCP3002 intègre un échantillon sur la puce et des circuits de retenue qui sont requis lors de la conversion de signaux d'entrée analogiques rapides. Les fonctions de broche sont :

CS/SHDN :	Entrée de sélection de puce
CH0 :	canal d'entrée analogique 0
CH1 :	canal d'entrée analogique 1
Vss :	mise à la terre de l'alimentation
Din :	Entrée de données SPI
Dout :	Sortie des données SPI
CLK :	SPI horloge
Vdd/Vref :	alimentation et tension de référence

Installation du nœud MCP3002 : Avant d'utiliser le MCP 3002 dans une application Node-RED, nous devons installer le nœud MCP3002 dans la palette de nœuds (les étapes suivantes installent des nœuds MCP 3xxx, qui incluent d'autres puces ADC de la même famille). Les étapes sont indiquées ci-dessous :

- Cliquez **Menu -> Gérer la palette** et cliquez **Installer**
- Entrer **node-red-node-pi-mcp3008** dans le champ de recherche et cliquez **installer**
- Lorsque l'installation est terminée, vous devriez voir un nouveau nœud appelé **Con A/D- verter**

Vous pouvez également exécuter la commande suivante à partir de votre répertoire par défaut Node-RED de ~/ . **node- rouge**

```
npm install node-red-node-pi-mcp3008
```

Nous devons également activer l'interface SPI sur notre Raspberry Pi 4. Les étapes sont :

- pi@framberrypi :~ \$ **sudo raspi-config**
- Descendre à **5 . Options d'interfaçage**
- Aller **P4 SPI**, cliquez dessus et activez l'interface SPI

Le Raspberry Pi possède les deux ports SPI avec les broches suivantes :

Port SPI	Fonction SPI Raspberry Pi	Nom de la broche	Numéro du port physique
SPI0	MOSI	GPIO 10	19
SPI0	MISO	GPIO 9	21
SPI0	CLK	GPIO 11	23
SPI0	CE0	GPIO 8	24
SPI0	CE1	GPIO 7	26
SPI1	MOSI	GPIO 20	38
SPI1	MISO	GPIO 19	35
SPI1	CLK	GPIO 21	40
SPI1	CE0	GPIO 18	12
SPI1	CE1	GPIO 17	11

La communication SPI nécessite un minimum de 3 broches : MOSI (Master Out Slave In), MISO (Master In Slave Out), et CLK (horloge). Dans certaines applications, il peut également être nécessaire d'activer le périphérique SPI à l'aide de l'une des broches CE (Chip Enable).

Un projet est présenté dans la section suivante qui montre comment le MCP 3002 peut être utilisé dans un projet.

7 . 3 Projet 32 – Voltmètre avec sortie LCD

Description : Il s'agit d'un projet de voltmètre. La tension à mesurer est appliquée aux bornes d'entrée. La valeur mesurée est affichée en millivolts sur un écran LCD. Dans ce projet, assurez-vous que la tension d'entrée n'est pas supérieure à 3300mV.

Viser : Le but de ce projet est de montrer comment un ADC peut être utilisé dans un projet Node-RED avec Raspberry Pi.

Schéma de principe : Le schéma du projet est présenté à la figure 7.2.

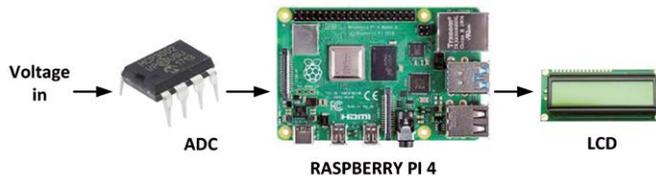


Figure 7.2 Schéma du projet

Schéma de circuit : La figure 7.3 montre le schéma du projet. L'ADC MCP 3002 est connecté au Raspberry Pi comme suit :

Broche MCP302	Raspberry Pi Port	Raspberry Pi Physical Pin No 23
CLK	GPIO 11 (CLK) GPIO	21
Dout	9 (MISO) GPIO 10	19
Din	(MOSI) GPIO 7 (CE1)	26
CS/SHDN	GND	6
Vss	+3,3V	1
Div		

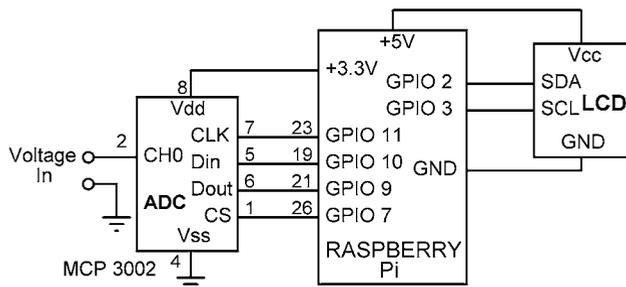


Figure 7.3 Schéma du projet

Programme de flux Node-RED : La figure 7.4 montre le programme de flux qui se compose de 4 nœuds : **injecter** nœud, un **Convertisseur A/D** nœud, **fonction** nœud et un **LCD20x4 I2C** nœud.

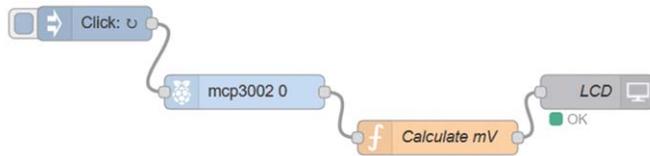


Figure 7.4 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud avec un intervalle de 1 seconde
- Créer un **Convertisseur A/D** et le configurer comme indiqué dans la Figure 7.5. Ici, définissez **Appareil** à mcp3002, **Broche d'entrée** à A0 puisque nous utilisons le canal CH0, le **Identifiant de l'appareil** à CE1, et **Bus SPI** à 0, cliquez **Terminé**

Device	mcp3002
Input pin	A0
Device ID	CE1
SPI bus	0

Figure 7.5 Configuration du nœud de convertisseur A/D

- Créer un nœud de fonction et le nommer comme **Calculer mV**, et entrez les instructions suivantes dans ce nœud. Ici, la valeur numérique lue par le port est convertie en millivolts physiques par multiplication avec $3300.0 / 1024.0$. Le résultat à virgule flottante est ensuite formaté à l'aide de la fonction **toFixed()** de sorte qu'il ne reste que 3 chiffres après la virgule :

```
var mV = msg.payload * 3300.0 / 1024.0; var V =
mV.toFixed(3) + " mV"; msg.payload={msg:[
```

```
{msg :
V} ]};
```

```
retourner le message;
```

- Créer un **LCD20x4 I2C** nœud comme dans les projets du Chapitre 6. Joindre tous les nœuds et cliquer **Déployer**.

- Raccorder l'entrée CH0 de l'ADC à une tension externe comprise entre 0V et +3,3V. La valeur de la tension sera affichée sur l'écran LCD en millivolts comme indiqué dans l'exemple de la figure 7.6.

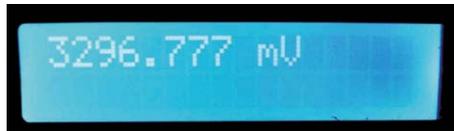


Figure 7.6 Exemple d'affichage de la tension

7 . 4 Projet 33 – Mesure de la température à l'aide d'un capteur analogique

Description : Dans ce projet, une puce de capteur de température analogique est utilisée pour mesurer la température ambiante et l'afficher toutes les secondes sur l'écran ACL.

Viser : Le but de ce projet est de montrer comment une puce de capteur analogique peut être interfacée avec Raspberry Pi via un ADC dans un projet Node-RED.

Renseignements généraux : Dans ce projet, on utilise la puce de capteur analogique populaire de type TMP36. Il s'agit d'une puce à 3 broches (figure 7.7) qui présente les caractéristiques suivantes :

- 2,7 à 5,5V fonctionnement
- précision de 2 oC
- Linéarité de 0,5 oC
- -40oC à +125oC plage de mesure
- Fonctionnement à très faible courant

La relation entre la température mesurée et la tension de sortie est exprimée comme suit :

$$T = (V_o - 500) / 10$$

Où T est la température en oC et V_o est la tension de sortie du capteur en millivolts. Par exemple, si la tension de sortie du capteur est de 800 mV, la température mesurée est $(800 - 500) / 10 = 30$ oC et ainsi de suite.

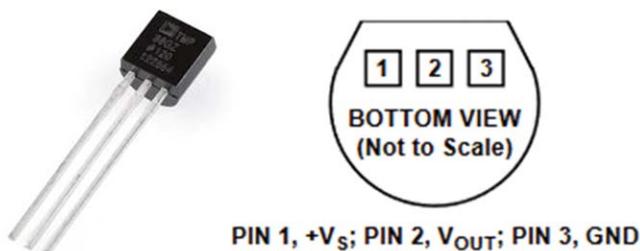


Figure 7.7 Puce de capteur de température TMP36

Lower 4 Bits	Upper 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣
xxxx0001	(2)	␣	!	1	A	Q	a	9	A	J	i	±	Á	Ñ	á	ñ	
xxxx0010	(3)	“	”	2	B	R	b	r	W	Γ	ϕ	²	À	Ò	à	ò	
xxxx0011	(4)	”	#	3	C	S	c	s	3	π	€	³	Ã	Ó	ã	ó	
xxxx0100	(5)	␣	\$	4	D	T	d	t	Μ	Σ	×	₠	Ä	Ö	ä	ö	
xxxx0101	(6)	␣	%	5	E	U	e	u	Ÿ	σ	¥	µ	Å	Ö	ä	ö	
xxxx0110	(7)	␣	&	6	F	V	f	v	J	¶	!	¶	€	Ö	ø	ö	
xxxx0111	(8)	␣	'	7	G	W	w	Π	τ	§	.	ϕ	×	ϕ	÷		
xxxx1000	(1)	↑	(8	H	X	h	x	Υ	‡	¶	ω	É	Ê	é	ê	
xxxx1001	(2)	↓)	9	I	Y	i	y	Ϸ	Θ	¹	É	Ù	é	ù		
xxxx1010	(3)	÷	*	:	J	Z	j	z	4	Ω	∞	É	Ù	é	ù		
xxxx1011	(4)	←	+	;	K	[k	[ω	∞	∞	É	Ù	é	ù		
xxxx1100	(5)	£	,	<	L	\	l	l	Ϸ	*	∞	¥	i	ü	i	ü	
xxxx1101	(6)	≥	-	=	M]	m	>	b	¶	¶	í	ý	í	ý		
xxxx1110	(7)	␣	.	>	N	^	n	~	bl	ε	Q	¼	î	ï	î	ï	
xxxx1111	(8)	␣	/	?	Q	_	o	Q	Q	Q	Q	‘	¿	ï	ï	ü	

Figure 7.10 Jeu de caractères ACL

7 . 5 Projet 34 – Contrôle de la température ON/OFF

Description : Il s'agit d'un projet de contrôle de la température ON/OFF d'une pièce. Le projet comprend une puce de capteur de température, un relais, un appareil de chauffage, une DEL et un écran LCD. Le projet contrôle le chauffage par l'intermédiaire du relais de sorte que la température dans la pièce soit à la valeur de consigne souhaitée. Si la température dans la pièce est inférieure au point de consigne, le relais est activé pour allumer le chauffage. En même temps, le LD est mis sous tension pour indiquer que le chauffage est allumé. Si, par contre, la température dans la pièce est supérieure au point de consigne, le relais est désactivé de sorte que le chauffage est mis OFF pour ramener la température ambiante à la valeur de consigne requise. En même temps, la LED est éteinte pour indiquer que le chauffage est ÉTEINT. La rangée supérieure de l'écran LCD indique la température de consigne, tandis que la température ambiante réelle est affichée dans la deuxième rangée de l'écran LCD.

Viser : Le but de ce projet est de montrer comment un projet de contrôle de température de type ON/OFF peut être développé en utilisant Node-RED avec un Raspberry Pi.

Renseignements généraux : Un **Contrôleur de température ON/OFF**, également connu sous le nom de contrôleur bang-bang, éteint le chauffage dès que la température est supérieure à un point de consigne prédéfini (c.-à-d. la valeur requise). De même, si la température est inférieure au point de consigne, le chauffage est mis sous tension. Les contrôleurs ON/OFF sont utilisés pour la simplicité puisqu'il n'est pas nécessaire de connaître le modèle de l'installation à contrôler. Le seul paramètre à définir est la valeur de consigne. La commande ON-OFF convient lorsque le délai de réponse de l'installation doit être

contrôlé est court et le taux de sortie du temps de montée est petit. Si un contrôle de température précis est requis, il peut être plus approprié d'utiliser le PID professionnel (Proportionnel + Int- gral+Derivative) des algorithmes de type contrôleur avec rétroaction négative ou d'utiliser un contrôleur intelligent basé sur fuzzy. Le problème avec la plupart des contrôleurs professionnels est qu'un modèle précis de l'usine à contrôler est normalement nécessaire avant qu'un algorithme de contrôle approprié puisse être dérivé. Le contrôleur de type ON/OFF présente l'inconvénient que le relais utilisé pour allumer et éteindre le chauffage doit fonctionner plusieurs fois, ce qui peut réduire la durée de vie du relais.

Schéma de principe : La figure 7.11 montre le schéma du projet.

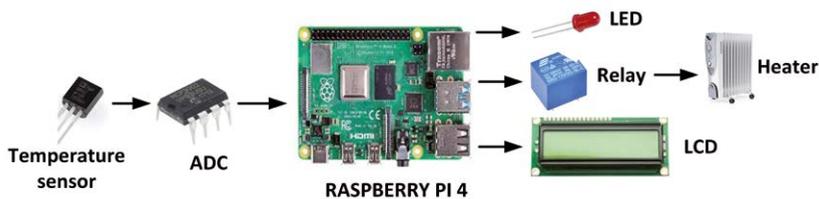


Figure 7.11 Schéma du projet

Schéma de circuit : Le schéma du projet est présenté à la figure 7.12. La plupart des parties du circuit sont similaires à la figure 7.8, mais ici en plus un relais, un chauffage et une LED sont connectés au système. Notez que certains relais fonctionnent uniquement avec +5V et si vous avez un de ces relais, alors vous aurez besoin d'avoir une puce convertisseur de niveau logique +3.3V à +5V pour conduire le relais (voir l'Internet pour un convertisseur approprié). Les connexions entre le Raspberry Pi et les composants externes sont les suivantes :

Raspberry Pi Port	Composante
GPIO 2	SDA CLD
GPIO 3	LCD SCL
GPIO 17	Relais d'anode
GPIO 27	LED
GPIO 11	MCP 3002 CLK
GPIO 10	MCP 3002 Din
GPIO 9	MCP 3002 Dout
GPIO 7	MCP 3002 CS

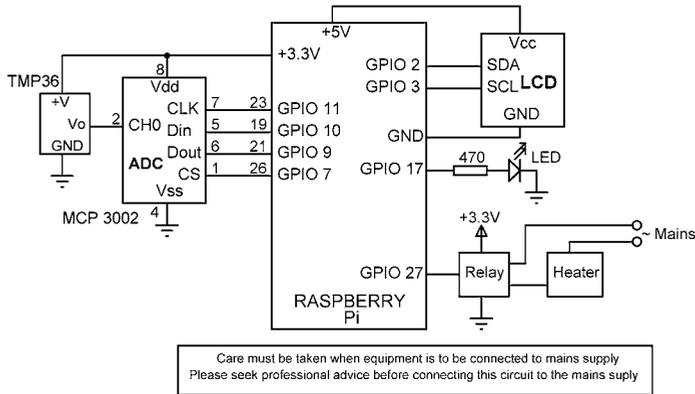


Figure 7.12 Schéma du projet

Programme de flux Node-RED : La figure 7.13 montre le programme de flux qui se compose de 6 nœuds : injecter nœud, un **Convertisseur A/D** nœud pour lire la température ambiante, un **fonction** le contrôleur ON/OFF, et un **LCD20x4 I2C** nœud pour afficher le point de consigne et les températures réelles, une **rpi gpioout** nœud connecté à la LED, et un autre **rpi gpio out** Le nœud est connecté au relais.

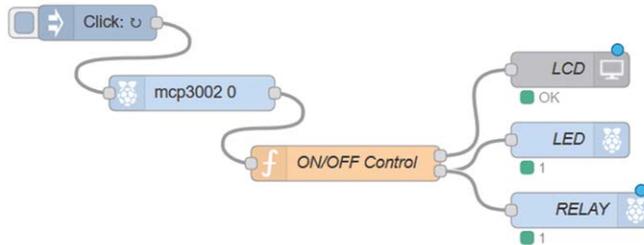


Figure 7.13 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud avec un intervalle de 1 seconde
- Créer un **Convertisseur A/D** nœud et le configurer comme indiqué à la figure 7.5
- Créer un **fonction** Node avec 2 sorties, et nommez-le comme Contrôle ON/OFF (voir la figure 7.14). Entrez les instructions suivantes dans ce node. Dans cet exemple, la température de consigne est réglée à 22°C. Variable T est chargé avec le tempérément réel- Température toutes les secondes. La température de consigne s’affiche dans la rangée supérieure de l’écran LCD. De même, la température mesurée est affichée sur la deuxième rangée de l’écran LCD. Si la température mesurée réelle est inférieure à la valeur de consigne, alors la LED et le relais sont allumés. Si, d’autre part, la mesure réelle- la température est égale ou supérieure à la valeur du point de consigne, puis la LED et le relais sont éteints. Cliquez sur **Terminé** :

```

var setpnt = 22,0;
var DegreeSymbol = String.fromCharCode(0xDF);
var Setpoint = '"Setpoint= &quot; + setpnt.toFixed(2) + DegreeSymbol;

var mV = msg.payload * 3300,0 / 1024,0;
var T = (mV - 500,0) / 10,0;
var Temp = '"Actual = &quot; + T.toFixed(2) + DegreeSymbol +&quot;C&quot;;

msg.payload={msgs:[
    {msg : Setpoint},
    {msg : Temp}
    ]};

si (T < setpnt) {

    var msg1 = {payload :
    '"1&quot;}; return [msg, msg1];
}
autrement
{
    var msg2 = {payload :
    '"0&quot;}; return [msg, msg2];
}

```

Figure 7.14 Configuration de la fonction du nœud

- Créer un **LCD20x4 I2C** node et le configurer comme avant
- Créer un **rpi gpio** et le nommez comme LED. Définissez le nom du port sur GPIO 17, et initialisez à 0
- Créer un autre **rpi gpio** nom de nœud comme RELAY. Définissez le nom du port sur GPIO 27, et initialisez à 0
- Connecter tous les nœuds comme dans la figure 7.13 et cliquer **Déployer**. Le programme doit maintenant contrôler la température de la pièce, au besoin.

La figure 7.15 montre un exemple d'affichage sur l'écran LCD où la température de consigne est de 22°C

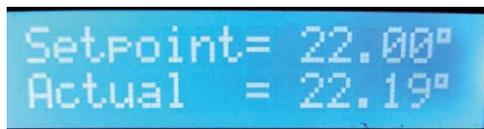


Figure 7.15 Exemple d'affichage

7 . 6 Résumé

Dans ce chapitre, nous avons appris à utiliser une puce convertisseur analogique-numérique externe dans un projet Raspberry Pi basé sur Node-RED. Dans le prochain chapitre, nous installerons et utiliserons le tableau de bord pour créer des tableaux de bord dynamiques en direct dans les projets Node-RED.

Chapitre 8 • La palette du tableau de bord

8.1 Aperçu

Dans le dernier chapitre, nous avons appris comment utiliser un convertisseur analogique-numérique dans un projet Raspberry Pi basé sur Node-RED. Dans ce chapitre, nous allons installer et utiliser le Dash- palette de cartes.

La palette Dashboard fournit un ensemble de nœuds dans Node-RED pour créer rapidement un tableau de bord de données en direct. C'est un module optionnel et non installé par défaut.

8.2 Installation du tableau de bord

Les instructions pour installer le tableau de bord sur Node-RED sont les suivantes :

- Cliquez **Menu -> Gérer la palette**, et cliquez sur **Installer**
- Entrer **noeud-rouge-tableau de bord** et cliquez **installer** (voir figure 8.1)

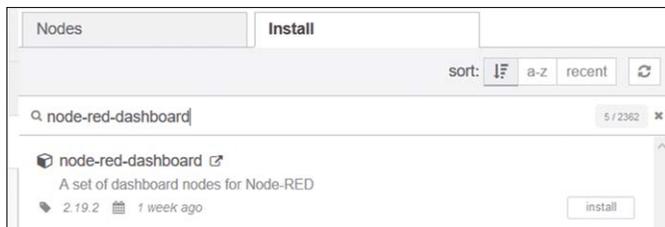


Figure 8.1 Installation du tableau de bord

- Vous verrez de nombreux nouveaux nœuds ajoutés à la palette des nœuds sous le titre **dash- conseil** comme indiqué à la figure 8.2

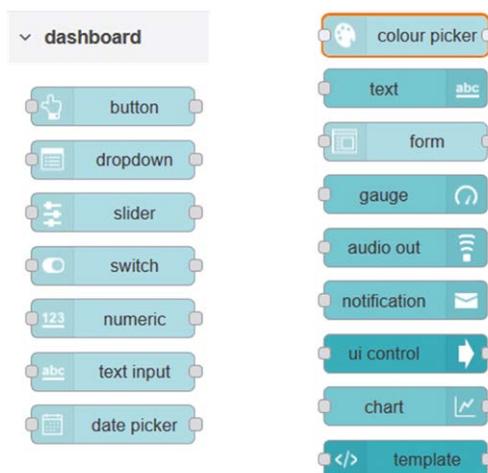


Figure 8.2 Les nœuds du tableau de bord

Nous sommes maintenant prêts à utiliser les nœuds du tableau de bord. Quelques exemples sont donnés dans le prochain paragraphe.

8 3 Projet 35 – Utiliser une jauge pour afficher la température

Description : Dans le dernier chapitre, nous avons vu comment lire et afficher la température sur un écran LCD. Dans ce projet, nous allons lire la température ambiante toutes les secondes et l'afficher sur un nœud de jauge.

Viser : Le but de ce projet est de montrer comment un nœud de jauge peut être utilisé dans un projet Node-RED

Schéma de circuit : Le schéma de circuit du projet est comme dans la figure 7.8 où une puce de capteur de température de type TMP36 est reliée à un ADC de type MCP 3002.

Programme de flux Node-RED : La figure 8.3 montre le programme de flux qui se compose de 4 nœuds : **injecter** nœud, un **Convertisseur A/D** nœud, un **fonction** nœud et un **jauge** nœud.

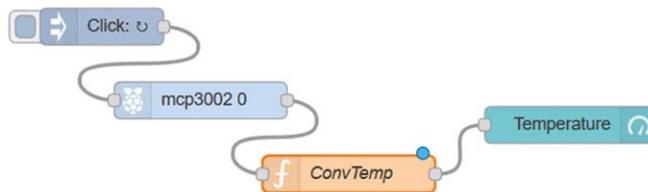


Figure 8.3 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud avec un intervalle de 1 seconde
- Créer un **Convertisseur A/D** nœud tel que décrit au chapitre 7
- Créer un **fonction** nœud et le nommer **ConvTemp**. Entrez l'état suivant- Se connecte à ce nœud pour lire la température et la convertir en degrés centigrades

```
var mV = msg.payload * 3300.0 / 1024.0; var T =
(mV - 500.0) / 10.0; msg.payload = T.toFixed(1);
retourner le message;
```

- Créer un **jauge** nœud et connecter tous les nœuds ensemble comme indiqué à la figure 8.3. Configurer le nœud comme suit :
- Double-cliquez sur le **jauge** nœud

- Cliquez sur le **Groupe** bouton de champ pour ajouter et configurer un tableau de bord. Vous devriez voir le formulaire d'édition du groupe de tableaux de bord. Configurez-le comme indiqué dans la figure 8.4. Cliquez sur **Mise à jour**

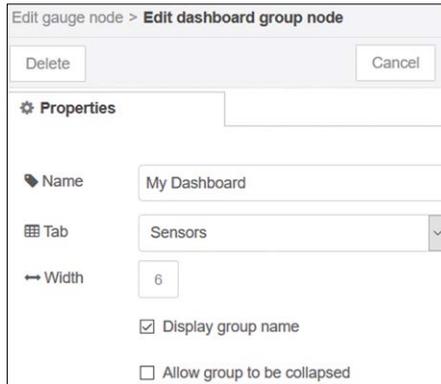


Figure 8.4 Modifier le nom du groupe de tableau de bord

- Configurer le **jauge** nœud comme indiqué à la figure 8.5. Ici, le **Étiquette** est réglé sur Température, **Unités** aux degrés C, **min** valeur à 0 et **max** valeur à 40. Cliquez **Terminé**

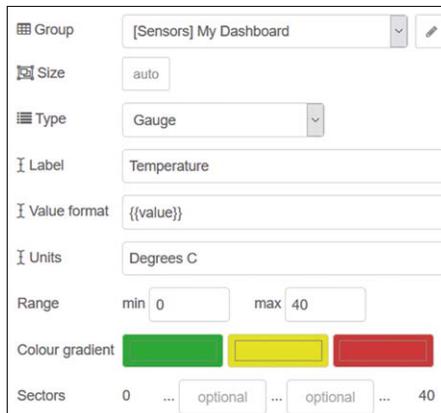


Figure 8.5 Configuration du nœud de jauge

- Cliquez sur le bouton de la **injecter** nœud. Pour voir la température sur l'écran de la jauge, allez sur le site suivant avec votre navigateur :

<http://192.168.1.202:1880/ui/>

où 192.168.1.202 est l'adresse IP de votre Raspberry Pi. L'affichage dans ce projet est indiqué sur la figure 8.6.

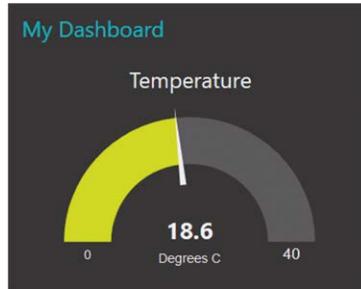


Figure 8.6 Affichage de la température

L'option par défaut de la tabulation est Accueil, et elle vous permet de spécifier sur quel onglet de la page d'interface utilisateur vous verrez l'élément d'interface utilisateur (ici notre graphique). Le champ Name est le nom du nœud Node-RED. Le champ Group permet de grouper les éléments de l'interface utilisateur.

8 . 4 Utiliser un graphique linéaire pour afficher la température

Dans le dernier projet, nous avons utilisé une jauge pour afficher la température ambiante. Dans cette section, nous allons afficher la température à l'aide d'un graphique linéaire. **Le schéma du projet est celui de la figure 7.8.**

Le programme de flux est le même que dans la figure 8.3, mais le **jauge** le nœud est remplacé par un **graphiquenœud** et un **Graphique linéaire est choisi comme indiqué à la figure 8.7.**

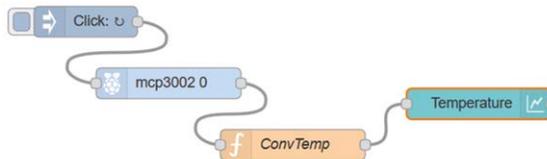


Figure 8.7 Programme de flux de l'exemple

Pour configurer le nœud graphique, double-cliquez dessus et configurez comme indiqué dans la figure 8.8

Group	[Sensors] My Dashboard
Size	auto
Label	Temperature
Type	Line chart <input type="checkbox"/> enlarge points
X-axis	last 10 minutes OR 1000 points
X-axis Label	HH:mm:ss
Y-axis	min 10 max 30
Legend	Show <input type="checkbox"/> Interpolate linear

Figure 8.8 Configurer le nœud de graphique

L'affichage de cet exemple est illustré à la figure 8.9.

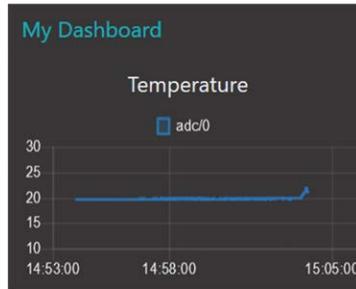


Figure 8.9 Affichage de la température

8.5 Utiliser un graphique à barres pour afficher la température

Dans le dernier projet, nous avons utilisé un graphique à lignes pour afficher la température ambiante. Dans cette section, nous allons afficher la température en utilisant un graphique à barres. **Comme il n'y a qu'un seul point** à- Le graphique à barres ne comporte qu'une seule colonne.

Le programme de flux est le même que dans la figure 8.3, mais le **jauge** le nœud est remplacé par un **graphique**. On choisit un graphique à barres comme indiqué sur la figure 8.10.

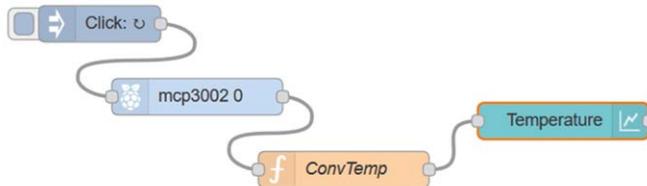


Figure 8.10 Programme de flux de l'exemple

Pour configurer le nœud graphique, double-cliquez dessus et configurez comme indiqué dans la figure 8.11

Group	[Sensors] My Dashboard
Size	auto
Label	Temperature
Type	Bar chart
Y-axis	min 10 max 30
Legend	Show

Figure 8.11 Configurer le nœud de graphique

L'affichage de cet exemple est représenté à la figure 8.12.

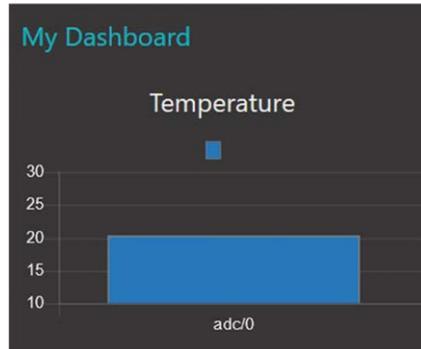


Figure 8.12 Affichage de la température

Vous avez les autres options de carte telles que la carte à barre horizontale, la carte circulaire, la carte polaire et la carte radar. La figure 8.13 montre la température affichée à l'aide d'un graphique à barres horizontal.

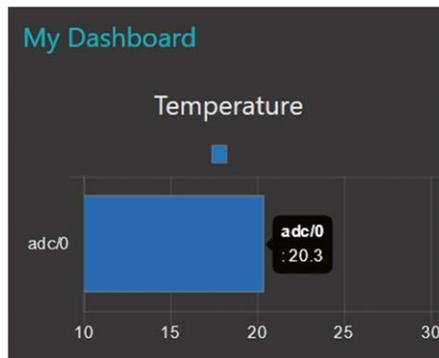


Figure 8.13 Affichage de la température à l'aide d'un graphique à barres horizontal

8 . 6 Projet 36 – Utilisation de jauges pour afficher la température et l'humidité

Description : Dans les derniers exemples, nous n'avons affiché qu'un seul élément à l'aide du tableau de bord. Dans ce projet, nous allons lire la température et l'humidité actuelles à partir du bulletin météo, puis les afficher en utilisant deux jauges.

Viser : Le but de ce projet est de montrer comment deux variables peuvent être affichées en utilisant deux séparément les jauges.

Programme de flux Node-RED : La figure 8.14 montre le programme de flux qui se compose de 5 nœuds **injecter** nœud, un **openweathermap** nœud qui lit le bulletin météo, un **fonction** nœud avec 2 sorties qui formatent les données et deux **jauges** nœuds, un **pour afficher le tem-** perature et l'autre pour afficher l'humidité.

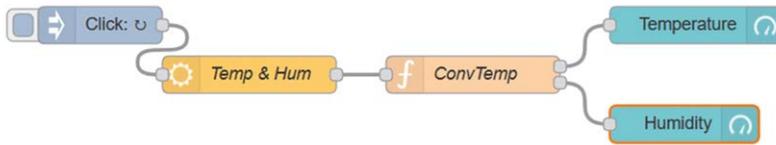


Figure 8.14 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** noeud à répéter chaque seconde
- Créer un **openweathermap** node et le nommer **Temp & Hum**, entrez votre clé API et cliquez sur **Terminé**
- Créer un **fonction** node et entrez les instructions suivantes dans ce node. Variables T et H extraient la température et l'humidité du bulletin météo actuel :

```
T = msg.payload.tempc;
H = msg.payload.humidity; var1 =
{payload : T};
var2 = {payload : H}; return [var1,
var2];
```

- Créer un **jauge** node et définir son **Étiquette** à la température, **Unités** aux degrés C, **min** à 0 et **max** à 30 cliquez sur **Terminé**.
- Créer un autre **jauge** node et définir son **Étiquette** à l'humidité, **Unités** en %, **min** à 0 et **max** à 100, cliquez **Terminé**
- Rejoindre tous les nœuds comme dans la figure 8.14 et cliquer **Déployer**

Vous devriez voir la température et l'humidité affichées sur deux jauges séparées comme dans la figure 8.15.

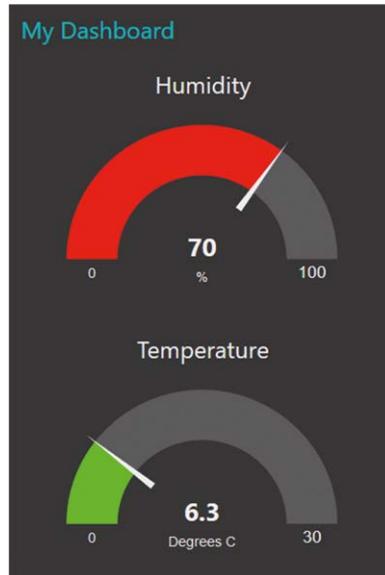


Figure 8.15 Affichage de la température et de l'humidité

8 . 7 Projet 37 – Utilisation de plusieurs jauges

Description : Dans ce projet, nous allons lire les prévisions météorologiques locales et afficher la fol- Utilisation de jauges : température minimale, température actuelle, température maximale, humidité et pression atmosphérique.

Viser : Le but de ce projet est de montrer comment plusieurs jauges peuvent être utilisées dans un programme Node-RED.

Programme de flux Node-RED : La figure 8.16 montre le programme de flux qui se compose de 8 nœuds.

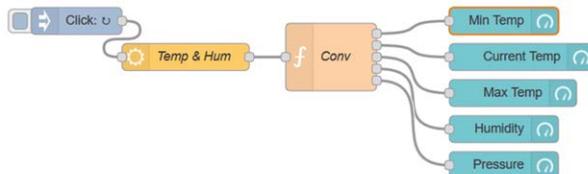


Figure 8.16 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** node à répéter chaque seconde comme dans le projet précédent
- Créer un **openweathermap** nœud comme dans le projet précédent
- Créer un **fonction** nœud avec les instructions suivantes :

```

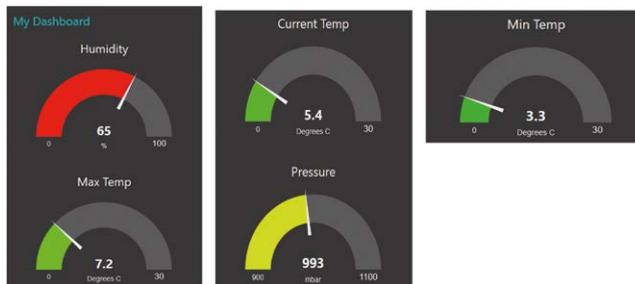
Tmin = msg.payload.temp_minc;
Tmax = msg.payload.temp_maxc;
T = msg.payload.tempc;
H = msg.payload.humidity;
P = msg.payload.pressure;
varTmin = {charge utile : Tmin};
varT = {charge utile : T};
varTmax = {charge utile : Tmax};
varH = {charge utile : H};
varP = {charge utile : P};
retour [varTmin,varT,varTmax,varH,varP];

```

- Créer 5 **jauge** nœuds pour la température minimale, la température actuelle, la température maximale, l'humidité et la pression atmosphérique. La configuration de ces nœuds est telle que décrite dans le projet précédent
- Rejoindre tous les nœuds et cliquer **Déployer**.
- Afficher le tableau de bord en saisissant (saisir l'adresse IP de votre propre Raspberry Pi) :

<http://192.168.1.202:1880/ui/>

Vous devriez voir les affichages comme dans la figure 8.17 (notez que les jauges sont affichées verticalement, et ils sont coupés et collés horizontalement dans cette figure)



8 . 8 Project 38 – Utiliser un curseur pour modifier la luminosité de la LED

Description : Dans cet exemple, une LED est connectée à l'un des ports du Raspberry Pi. La luminosité de la LED est modifiée en utilisant un nœud coulissant pour modifier le cycle d'utilisation de la forme d'onde de tension PWM envoyée à la LED.

Viser : Le but de ce projet est de montrer comment un nœud de curseur peut être utilisé.

Schéma de circuit : Le schéma de circuit du projet est le même que dans la figure 5.4 où la LED est reliée à la broche de port GPIO 2 par une résistance de limitation de courant.

Programme de flux Node-RED : La figure 8.18 montre le programme de flux qui se compose de 3 nœuds : a **curseur** Pour modifier le cycle de travail de la forme d'onde PWM (d'où l'intensité de la LED), un **rpio gpio out** le nœud de commande de la DEL, et un **jauge** noeud pour afficher la valeur actuelle du cycle d'utilisation.

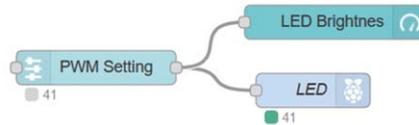


Figure 8.18 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **curseur** node et le nommer **Réglage du PWM**. Configurer ce nœud comme indiqué dans la figure 8.19, cliquez sur **Terminé**

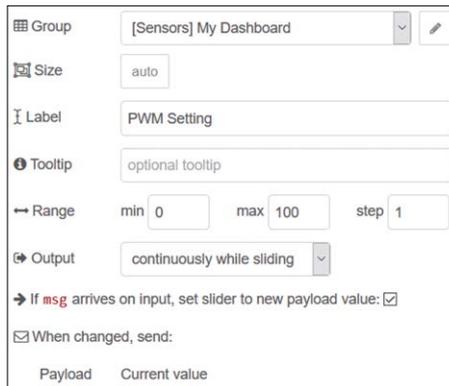


Figure 8.19 Configuration du nœud de curseur

- Créer un **rpio gpio** Sortir le nœud et le nommer **LED**. Régler la broche sur GPIO 2, et **Type** pour PWM, cliquer **Terminé**
- Créer un **jauge** noeud avec nom **Luminosité des DEL** et le configurer comme indiqué dans la figure 8.20, cliquez sur **Terminé**

Group	[Sensors] My Dashboard
Size	auto
Type	Gauge
Label	LED Brightnes
Value format	{{value}}
Units	%
Range	min 0 max 100

Figure 8.20 Configuration du nœud de jauge

- Rejoindre tous les nœuds et cliquer **Déployer**.
- Afficher le tableau de bord en saisissant (saisir l'adresse IP de votre propre Raspberry Pi) :

<http://192.168.1.202:1880/ui/>

- Déplacer le curseur au bas de la jauge (voir figure 8.21). Vous devriez voir la luminosité de la LED changer lorsque le curseur est déplacé de 0% à 100% (il est réglé à 41% dans la figure).



Figure 8.21 Déplacez le curseur pour changer la luminosité de la LED

8 . 9 Project 39 – Utilisation de nœuds à bouton pour commander une DEL

Description : Dans cet exemple, une LED est connectée à l'un des ports du Raspberry Pi comme dans le projet précédent. La LED est allumée et éteinte à l'aide des boutons du tableau de bord

Viser : Le but de ce projet est de montrer comment les nœuds du bouton Tableau de bord peuvent être utilisés

Schéma de circuit : Le schéma de circuit du projet est le même que dans la figure 5.4 où la LED est reliée à la broche de port GPIO 2 par une résistance de limitation de courant.

Programme de flux Node-RED : La figure 8.22 montre le programme de flux qui se compose de 3 nœuds : a **bouton** Pour allumer la LED, un **bouton** pour éteindre la LED, et une **rpi gpio out** bouton connecté à la LED.

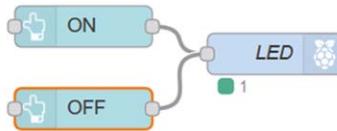


Figure 8.22 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **bouton** node et définir son **Étiquette** à **Sur**, et le **Charge utile** à 1 pour que la sortie du nœud devienne 1 lorsque vous cliquez sur le tableau de bord. Configurez ce nœud comme indiqué dans la figure 8.23, cliquez **Terminé**

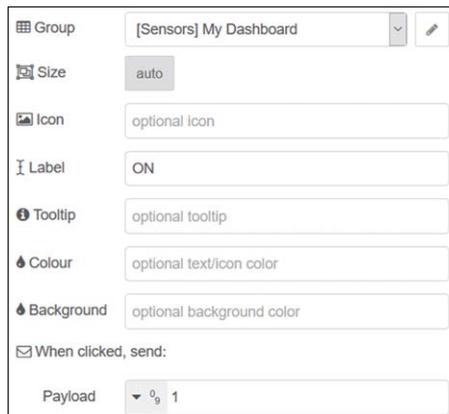


Figure 8.23 Configuration du nœud de bouton

- Créer un autre **bouton** node et définir son **Étiquette** à **OFF**, et le **Charge utile** à 0 pour que la sortie du nœud devienne 0 lorsque vous cliquez sur le tableau de bord
- Créer un **rpi gpio out** nœud et définir **Pin** à **GPIO**, **Type** à la sortie numérique, et initialiser à 0
- Rejoindre tous les nœuds et cliquer **Déployer** et afficher le tableau de bord comme indiqué à la figure 8.24. Cliquer sur **ON** pour allumer la LED, et sur **OFF** pour éteindre la LED

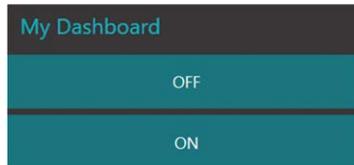


Illustration 8.24 Allumer et éteindre la DEL

8 . 10 Project 40 – Utilisation de commutateurs et de nœuds texte pour commander une DEL

Description : Dans cet exemple, une LED est connectée à l'un des ports du Raspberry Pi comme dans le projet précédent. La LED est allumée et éteinte en cliquant sur un commutateur. En outre, un texte est affiché sur le dessus du commutateur pour indiquer à l'utilisateur ce qu'il doit faire

Viser : Le but de ce projet est de montrer comment le commutateur Dashboard et les nœuds de texte peuvent être utilisés dans un projet.

Schéma de circuit : Le schéma de circuit du projet est le même que dans la figure 5.4 où la LED est connectée au port GPIO 2 de Raspberry Pi.

Programme de flux Node-RED : La figure 8.25 montre le programme de flux qui se compose de 3 nœuds : a **commutateur** Pour activer/désactiver la LED, un **texte** le noeud pour afficher un message, et **rpi gpio out** bouton connecté à la LED.

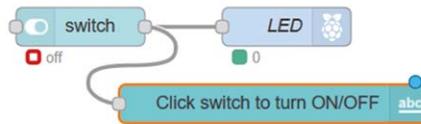


Figure 8.25 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **commutateur** nœud et définir **Sur la charge utile** à 1, et **Charge utile hors service** à 0, comme indiqué dans la figure 8.26, cliquez sur **Terminé**

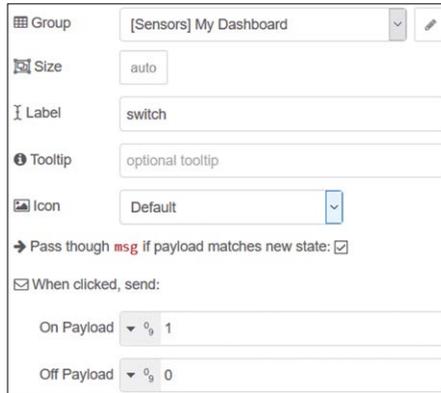


Figure 8.26 Configuration du nœud de commutation

- Créer un **texte** nœud et définir **Étiquette** pour cliquer sur le commutateur pour activer/désactiver
- Créer un **rpi gpio out** nœud et définir **Pin** à GPIO, **Type** à la sortie numérique, et initialiser à 0
- Rejoindre tous les nœuds et cliquer **Déployer** et afficher le tableau de bord comme indiqué dans la figure 8.27. Cliquez sur l'interrupteur pour allumer/éteindre la LED

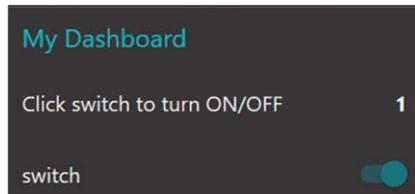


Illustration 8.27 Allumer et éteindre la DEL

8 . 11 Projet 41 – Prévisions météorologiques

Description : Il s'agit d'un projet intéressant sur la météo parlante. Dans ce projet, le bulletin météorologique local est reçu en utilisant **openweathermap** nœud et les données sont décodées puis envoyées aux haut-parleurs du PC à l'aide du tableau de bord **sortie audio** nœud. Par conséquent, vous pouvez entendre les prévisions météo sur vos haut-parleurs de PC. La prévision météorologique est répétée après 30 secondes lorsqu'elle s'arrête.

Viser : Le but de ce projet est de montrer comment le tableau de bord **sortie audio** Le nœud peut être utilisé dans un projet.

Programme de flux Node-RED : La figure 8.28 montre le programme de flux qui se compose de 12 nœuds : **injecter** node pour démarrer et répéter la prévision météo, un **openweathermap** node pour obtenir les prévisions météorologiques locales, un fonction **nœud qui extrait les divers paramètres météorologiques locaux et les envoie aux nœuds de retard**, 7 retard les nœuds qui insèrent des délais entre les messages, un rejoindre le nœud qui joint les messages, et un tableau de bord sortie audio

nœud qui génère le son sur les haut-parleurs du PC.

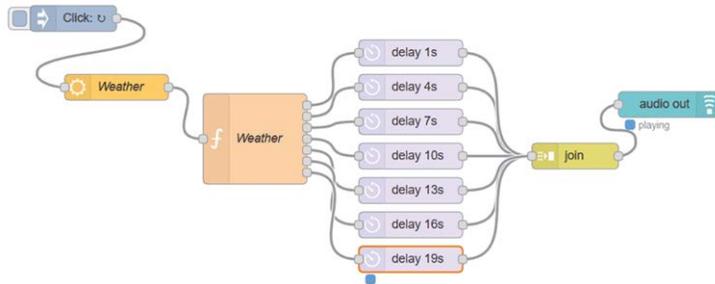


Figure 8.28 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecteur** et régler l'intervalle de répétition sur 54 secondes
- Créer un **openweathermap** node et le nommer **Météo**. Entrez la clé API, le nom de la ville et du pays (voir la figure 8.29)

Configuration du nœud openweathermap :

- API Key : [masqué]
- Language : English
- Current weather for : [masqué]
- Location : City, Country
- City : London
- Country : UK

Figure 8.29 Configuration du nœud openweathermap

- Créer un **fonction** noeud avec 7 sorties et le nommer comme **Décoder** (voir la figure 8.30). Entrez les instructions suivantes dans ce nœud. Ici, L est l'emplacement, T la température locale en degrés centigrades, H est l'humidité locale en %, P est la pression atmosphérique locale en millibars, W est la vitesse du vent local en mètres par seconde :

```
L = msg.payload.location;
T = msg.payload.tempc;
H = msg.payload.humidity;
P = msg.payload.pressure;
W = vitesse de la neige
TT = {charge utile : « La température est de "+T+" degrés centigrades »}; HH =
{charge utile : « L'humidité est de "+H+" pourcent »};
PP = {charge utile : "La pression atmosphérique est "+P+" millibars"}; WW =
{charge utile : "La vitesse du vent est "+W+" mètres par seconde"};
```

```
Head = {payload : &quot;Voici les prévisions météorologiques pour &quot; + L + &quot; aujourd'hui&quot;};
```

```
si (T < 15)
```

```
    MD = {charge utile : &quot;Il fait très froid et il est conseillé de porter une couche épaisse&quot;};
```

```
autrement
```

```
    MD = {charge utile : &quot;Le temps est doux et il n'y a pas besoin d'une couche épaisse&quot;};
```

```
Nxt = {charge utile : &quot;La prochaine prévision météorologique sera dans 30 secondes&quot;};
```

```
retour [Head, TT, HH, PP, WW, MD, Nxt];
```

Voici un exemple de prévisions météorologiques prononcées par les orateurs :

Voici les prévisions météorologiques pour London aujourd'hui. La température est de 7,1 degrés centigrades. L'humidité est de 53 %. La pression atmosphérique est de 1020 millibars. La vitesse du vent est de 3,2 mètres par seconde. Il fait très froid et il est conseillé de porter un manteau épais. La prochaine prévision météorologique sera dans 30 secondes.

```

Name: Decode
Function
1 L=msg.payload.location;
2 T=msg.payload.tempc;
3 H=msg.payload.humidity;
4 P=msg.payload.pressure;
5 W=msg.payload.windspeed;
6
7 TT={payload: "The temperature is "+T+" degrees centigrade"};
8 HH={payload: "The humidity is "+H+" percent"};
9 PP={payload: "The atmospheric pressure is "+P+" millibars"};
10 WW={payload: "The wind speed is "+W+" metres per second"};
11 Head={payload: "This is the weather forecast for " + L + " today"};
12
13 if(T < 15)
14     MD = {payload: "The weather is very cold and you are advised to wear a thick coat"};
15 else
16     MD = {payload: "The weather is mild and no need for a thick coat"};
17
18 Nxt = {payload: "The next weather forecast will be in 30 seconds time"};
19
20 return [Head, TT, HH, PP, WW, MD, Nxt];
Outputs: 7
    
```

Figure 8.30 Configuration de la fonction du nœud

- Créer un **rejoindre** nœud et le configurer comme indiqué sur la figure 8.31

Figure 8.31 Configuration de la jointure des nœuds

- Cliquez sur le **sortie audio** nœud dans la palette Dashboard, glissez et déposez dans le travail- espace. Configurer ce nœud comme indiqué sur la figure 8.32

Figure 8.32 Configuration de la sortie audio du nœud

- Rejoindre tous les nœuds comme indiqué dans la figure 8.28 et cliquer **Déployer**.
- Démarrer votre tableau de bord (entrez votre propre adresse IP) :

<http://192.168.1.202:1880/ui/>

- Cliquez sur le bouton de la **injecter** nœud. Vous devriez entendre les prévisions météorologiques re- Vous pouvez augmenter le volume de vos haut-parleurs

8 . 12 Configuration du tableau de bord

L'apparence du tableau de bord peut être facilement configurée. Cliquez **Menu -> Affichage -> Tableau de bord- conseil** et vous verrez la fenêtre Tableau de bord affichée sur le côté droit de l'écran. Cliquez sur **Thème** tab pour définir le style comme Sombre, Clair ou Personnalisé. Vous pouvez également définir la couleur du tableau de bord et la police.

8 . 13 Résumé

Dans ce chapitre, nous avons appris comment utiliser certains nœuds du tableau de bord dans les projets. Dans le prochain chapitre, nous utiliserons certains des nœuds réseau pour contrôler les appareils connectés à notre Raspberry Pi à partir d'un téléphone mobile intelligent

Chapitre 9 • Projets basés sur le réseau UDP/TCP Wi-Fi

9.1 Aperçu

Dans le dernier chapitre, nous avons appris comment utiliser le tableau de bord pour créer des projets intéressants. Dans ce chapitre, nous allons créer des projets basés sur le réseau Wi-Fi.

UDP (User Datagram Protocol) et TCP (Transmission Control Protocol) sont deux des protocoles de communication les plus couramment utilisés dans les applications basées sur le Wi-Fi pour envoyer et recevoir des paquets de données sur un réseau. TCP/IP est un protocole fiable qui inclut l'échange de données et garantit ainsi la livraison des paquets à la destination requise. UDP, par contre, n'est pas aussi fiable mais c'est un protocole rapide. Le tableau 9.1 montre une comparaison des communications de type UDP et TCP/IP.

TCP/IP	UDP
Protocole orienté connexion	Protocole sans connexion
Lent	Rapide
Transmission de données très fiable	Pas si fiable
Paquets disposés dans l'ordre	Pas de commande des paquets
Taille de l'en-tête 20 octets	Taille de l'en-tête 8 octets
Vérification et retransmission des erreurs	Pas de vérification des erreurs
Accusé de réception des données	Pas de reconnaissance
Utilisé dans HTTP, HTTPS, FTP etc	Utilisé dans DNS, DHCP, TFTP etc

Tableau 9.1 Comparaison de UDP et TCP/IP

Les programmes basés sur le protocole UDP et TCP/IP sont basés sur le serveur-client où un nœud envoie des données et l'autre nœud les reçoit et vice versa. Les données sont transférées par des ports où le serveur et les clients doivent utiliser les mêmes numéros de port.

Développer des programmes basés sur UDP et TCP nécessite de bonnes connaissances en programmation. Heureusement, Node-RED fournit des nœuds qui simplifient considérablement cette tâche. Dans ce chapitre, nous développerons des projets basés sur Node-RED pour établir la communication entre un Raspberry Pi et un téléphone mobile intelligent.

9.2 Projet 42 – Contrôle d'une DEL à partir d'un téléphone mobile – Communication basée sur UDP

Description : Dans cet exemple, une LED est connectée à l'un des ports du Raspberry Pi comme dans le chapitre précédent, dans Project 40. La LED est allumée et éteinte en envoyant des messages à partir d'un téléphone mobile en utilisant le protocole UDP. Les commandes valides sont :

Sur	Allumer la DEL
off	Éteindre la DEL

Viser : Le but de ce projet est de montrer comment la communication réseau basée sur UDP peut être établie en utilisant Node-RED.

Schéma de principe : La figure 9.1 montre le schéma du projet.

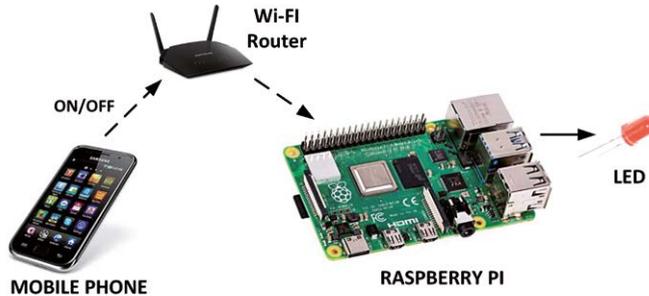


Figure 9.1 Schéma du projet

Schéma de circuit : Le schéma de circuit du projet est identique à celui de la figure 5.4 où une LED est connectée au port GPIO 2 du Raspberry Pi.

Programme de flux Node-RED : La figure 9.2 montre le programme de flux qui se compose de 3 nœuds : a **udp dans** nœud pour recevoir des paquets UDP du téléphone mobile, un **fonction** nœud pour formater les données reçues afin de contrôler la DEL, et **rpi gpio out** le nœud qui est connecté à la LED.



Figure 9.2 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **udp dans** nœud et le configurer comme indiqué dans la figure 9.3. Ici, le **Port** le nombre est réglé à 5000, et **Production** est défini sur string, cliquez **Terminé**

La capture d'écran montre la configuration d'un nœud 'udp dans'. Les paramètres sont : 'Listen for' : udp messages ; 'on Port' : 5000 ; 'using' : ipv4 ; 'Output' : a String ; 'Name' : Name.

Figure 9.3 Configuration du nœud udp dans

- Créer un **fonction** et le nommer ON/OFF. Entrez les instructions suivantes dans cette fonction. Ici, la sortie est définie sur 1 si la commande est ON et sur 0 si la commande est OFF. Cliquez **Terminé** :

```
var md = msg.payload.substr(0, 3); if(md
== "ON")
    msg.payload = 1;
sinon si(md == "OFF")
    msg.payload = 0;
renvoie msg;
```

- Créer un **rpi gpio out** node et définir la broche sur GPIO 2, nommer le nœud comme **LED**, Définissez-le sur la sortie numérique, et initialisez la broche à 0. Cliquez **Terminé**
- Rejoindre les 3 nœuds et cliquer **Déployer**.

Le programme de flux est maintenant terminé et il attend de recevoir des commandes UDP d'un téléphone mobile pour contrôler les LED. Dans ce projet, un mobile Android est utilisé pour tester le projet. Il existe de nombreuses applications UDP gratuites dans le Play Store qui peuvent être utilisées pour envoyer et recevoir des messages. Celle utilisée dans ce projet s'appelle **UDP RECEVOIR et ENVOYER** par *Wezzi Studios* (v 3.3), comme indiqué à la figure 9.4

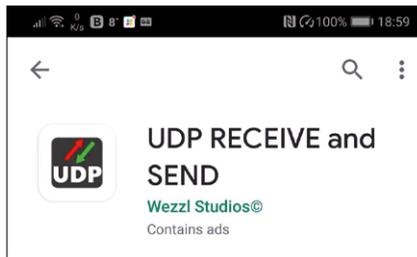


Figure 9.4 UDP RECEIVE et SEND applications

Les étapes suivantes sont indiquées pour utiliser cette application :

- Lancez les applications sur votre téléphone mobile
- Définir l'adresse IP sur celle de votre Raspberry Pi (dans ce projet, elle est 192.168.1.202)
- Définir le numéro de port à 5000
- Entrer le message **Sur** et cliquez sur ENVOYER UN MESSAGE UDP comme indiqué dans la figure 9.5 et vous devriez voir le LED allumer

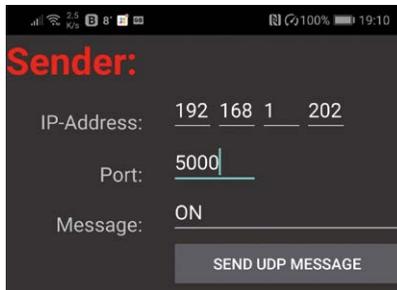


Illustration 9.5 Entrer ON pour allumer la LED

Entrer le message **off** et cliquez sur ENVOYER UN MESSAGE UDP comme indiqué dans la figure 9.6 et vous devriez voir le LED pour éteindre

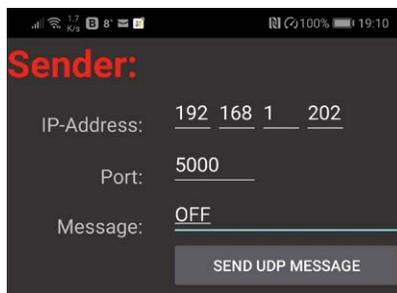


Illustration 9.6 Entrer OFF pour éteindre la LED

Bien que dans ce projet nous ayons contrôlé une simple LED, il n'y a aucune raison pour laquelle nous ne pouvons pas utiliser par exemple un relais afin que l'équipement électrique (p. ex., une chaudière, une lampe, etc.) puisse être commandé à distance.

9 . 3 Projet 43 – Contrôle de plusieurs LED à partir d'un téléphone mobile – Communication basée sur UDP

Description : Dans le projet précédent, nous n'avons contrôlé qu'une seule DEL. Il existe des applications où nous pouvons vouloir contrôler un certain nombre d'appareils à distance. Dans ce projet, nous allons contrôler 3 LED à partir d'un téléphone mobile. Les commandes valides sont :

1=ON	allumer la LED 1
2=ON	allumer la LED 2
3=ON	allumer la LED3
1=OFF	éteindre la LED 1
2=OFF	éteindre la LED 2
3=OFF	éteindre la LED 3

Viser : Le but de ce projet est de montrer comment plusieurs dispositifs peuvent être contrôlés à distance en utilisant un nœud UDP Node-RED.

Schéma de principe : La figure 9.7 montre le schéma du projet.

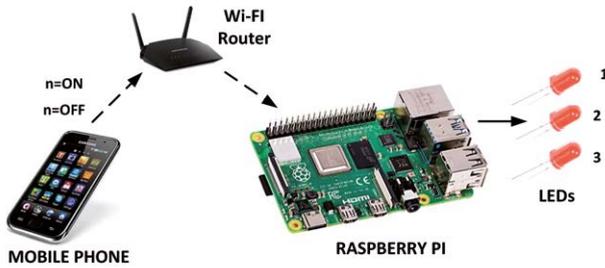


Figure 9.7 Schéma du projet

Schéma de circuit : Le schéma de circuit du projet est montré à la figure 9.8. Les LED sont connectées aux ports GPIO GPIO 2, GPIO 3 et GPIO 4.

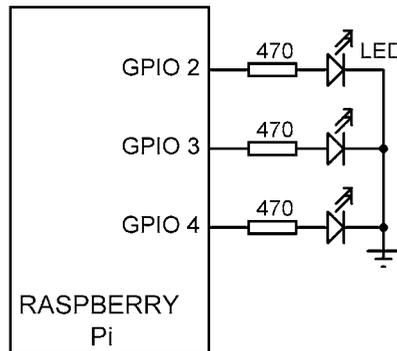


Figure 9.8 Schéma du projet

Programme de flux Node-RED : La figure 9.9 montre le programme de flux qui se compose de 5 nœuds : a **udp dans** nœud pour recevoir des paquets UDP du téléphone mobile, un nœud de fonction avec 3 sorties- met à formater les données reçues pour contrôler les 3 LED, et 3 **rpi gpio out** les nœuds qui sont connectés aux LED.



Figure 9.9 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **udp dans** nœud comme dans le projet précédent

Créer un **fonction** nœud avec 3 sorties et entrer les instructions suivantes à l'intérieur de ce nœud :

```
var no = msg.payload.substr(0, 1);
```

```

var md = msg.payload.substr(2, 3); var var1 =
null;
var var2 = null;
var var3 = null;

si(md == "ON")
{
    si (aucun == "1")
    {
        var1 = {payload : 1};
    }
    sinon si(non == "2") {

        var2 = {payload : 1};
    }
    sinon si(non == "3") {

        var3 = {payload : 1};
    }
}
sinon si(md == "OFF") {

    si (aucun == "1")
    {
        var1 = {charge utile : 0};
    }
    sinon si(non == "2") {

        var2 = {payload : 0};
    }
    sinon si(non == "3") {

        var3 = {payload : 0};
    }
}

retour [var1, var2, var3];

```

- Créer 3 **rpi gpio out** Les nœuds avec les broches suivantes :

```

LED1 : GPIO 2
LED2 : GPIO 3
LED3 : GPIO 4

```

- Rejoindre tous les nœuds comme dans la figure 9.9 et cliquer **Déployer**.

Pour tester le projet, vous pouvez utiliser les applications **UDP RECEVOIR et ENVOYER** Décrit dans le pré- vious. Par exemple, saisissez **2=ON** et LED2 doivent être activés, saisir **2=OFF** et la même LED doit être éteinte, etc.

Comme suggéré dans le projet précédent, les LED peuvent être remplacées par des relais pour contrôler l'électricité- Les équipements électriques à distance.

9 . 4 Projet 44 – Envoi de la température et de l'humidité au téléphone mobile – Communication basée sur UDP

Description : Dans les projets précédents, nous avons envoyé des données du téléphone mobile à la Raspberry Pi. Dans ce projet, nous allons faire le contraire. . Nous enverrons les données du Raspberry Pi au téléphone mobile. Le programme attendra de recevoir la commande **météo** depuis le téléphone mobile. Lorsque cette commande est reçue, la météo locale sera reçue et la température et l'humidité seront envoyées au téléphone mobile en utilisant le protocole UDP. Le fonctionnement du programme est le suivant :

Téléphone mobile : Envoie la commande **météo** à Raspberry Pi

Raspberry Pi : Obtient le bulletin météo local et envoie les données de température et d'humidité au téléphone mobile

Viser : Le but de ce projet est de montrer comment les données peuvent être envoyées de Raspberry Pi au téléphone mobile en utilisant le protocole UDP.

Schéma de principe : La figure 9.10 montre le schéma du projet.

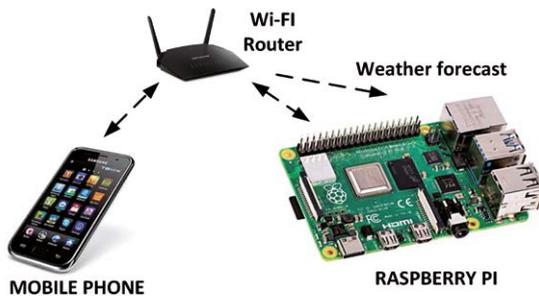


Figure 9.10 Schéma du projet

Programme de flux Node-RED : La figure 9.11 montre le programme de flux qui se compose de 6 nœuds : a **udp dans** nœud pour recevoir des paquets UDP du téléphone mobile, un **fonction** nœud qui vérifie quand commande **météo** est reçu, un **commutateur** nœud qui active **ouvrir- carte météorologique** node pour obtenir le bulletin météo local, un **fonction** node permettant d'extraire les données locales sur la température et l'humidité à partir du bulletin météo, et **udp out** nœud pour envoyer les données de température et d'humidité au téléphone mobile.

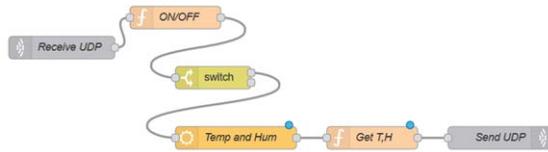


Figure 9.11 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **udp** dans nœud comme dans le projet précédent et nommez-le comme suit **Recevoir UDP**
- Créer un **fonction** node et le nommer **ON/OFF**. Entrez l'instruction suivante dans cette fonction retournera 1 lorsque la commande **météo** est reçu de l'UDP :

```
var md = msg.payload;
if(md == "weather")
    msg.payload = 1;
sinon
    msg.payload = 0;
```

retourner le message;

- Créer un **commutateur** le nœud qui active **openweathermap** nœud lorsque 1 est reçu. Configurer ce nœud comme indiqué dans la figure 9.12. Notez que la deuxième sortie de ce nœud n'est pas utilisée.



Figure 9.12 Configuration du commutateur de nœud

- Créer un **openweathermap** node et le nommer **Temp et Hum**, entrez la clé API, définissez votre ville (Londres dans ce projet), et le pays (le Royaume-Uni dans ce pro-)
- Créer un **fonction** node et le nommer **Obtenir T, H**, entrez les instructions suivantes dans cette fonction. Cette fonction extrait et renvoie la température et le hu- midité du bulletin météo :

```
T = "Temperature=" + msg.payload.tempc + "C"; H =
"Humidity=" + msg.payload.humidity + "%";
```

```
D = T + " " + H;
msg.payload = D;
return msg;
```

- Créer un **udp out** nœud, nommez-le comme **Envoyer UDP**, et le configurer comme indiqué dans la figure 9.13. Ici, l'adresse IP de votre téléphone portable doit être entrée dans le champ Adresse (dans ce projet c'était 192.168.1.178)



Figure 9.13 Configuration de la sortie du nœud udp

- Rejoindre tous les nœuds comme indiqué sur la figure 9.11 et cliquer **Déployer**

Vous pouvez tester le programme à l'aide des applications **UDP RECEVOIR** et **ENVOYER** décrit dans le précédent- Tous les projets. Saisissez le numéro de port 5000. Ensuite, saisissez la commande **météo** et cliquez sur ENVOYER MESSAGE UDP. Les données locales de température et d'humidité s'afficheront sur votre téléphone mobile, comme indiqué dans la figure 9.14.

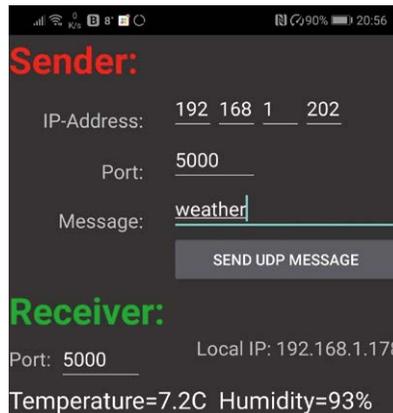


Figure 9.14 Affichage de la température et de l'humidité locales

9 . 5 Projet 45 – Contrôle d'une DEL à partir d'un téléphone mobile – communication basée sur le protocole TCP

Description : TCP est un protocole de communication fiable basé sur la connexion . Dans cet exemple, une LED est connectée à l'un des ports du Raspberry Pi comme dans le chapitre précédent, dans Project 42. La LED est allumée et éteinte en envoyant des messages à partir d'un téléphone mobile en utilisant le protocole TCP protocole. Les commandes valides sont :

Sur	Allumer la DEL
off	Éteindre la DEL

Viser : Le but de ce projet est de montrer comment une communication réseau fiable basée sur TCP peut être établie en utilisant Node-RED.

Schéma de principe : Le schéma du projet est identique à celui de la figure 9.1.

Schéma de circuit : Le schéma de circuit du projet est identique à celui de la figure 5.4 où une LED est connectée au port GPIO 2 du Raspberry Pi.

Programme de flux Node-RED : La figure 9.15 montre le programme de flux qui se compose de 3 nœuds : a **tcp dans** nœud pour recevoir des paquets TCP du téléphone mobile, un **fonction** nœud pour formater les données reçues afin de contrôler la DEL, et **rpi gpio out** le nœud qui est connecté à la LED.



Figure 9.15 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **tcp dans** et le configurer comme indiqué dans la Figure 9.16. Un nœud TCP peut être configuré soit comme serveur, soit comme client. Lorsqu'il est configuré comme un serveur, le nœud écoute pour une connexion sur un port spécifié. Lorsqu'il est configuré en tant que client, le nœud se connecte à une adresse IP distante spécifiée. Dans cet exemple, le nœud est configuré comme un serveur et le numéro de port est défini sur 5000



Figure 9.16 Configuration du nœud tcp dans

- Créer un **fonction** et nommez-le ON/OFF comme dans Project 41. Entrez la- Les instructions dans cette fonction.

```
var md = msg.payload.substr(0, 3); if(md
== "ON")
    msg.payload = 1;
sinon si(md == "OFF")
    msg.payload = 0;
renvoie msg;
```

- Créer un **rpi gpio out** node et définir la broche sur GPIO 2 comme dans Project 41
- Rejoindre les 3 nœuds et cliquer **Déployer**.

Le programme de flux est maintenant terminé et il attend pour se connecter à un client TCP. Dans ce projet, un mobile Android est utilisé pour tester le projet comme avant. Il existe de nombreuses applications TCP gratuites dans le Play Store qui peuvent être utilisées pour envoyer et recevoir des messages TCP. Celle utilisée dans ce projet s'appelle **Simple TCP Socket Tester** par *Armando Jesus Gomez Parra*, comme indiqué à la figure 9.17

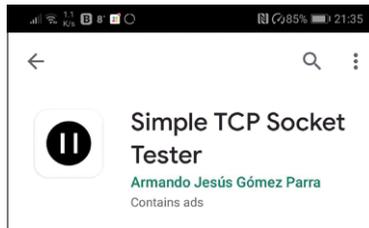


Figure 9.17 Applications de testeur de socket TCP simples

Les étapes suivantes sont indiquées pour utiliser cette application :

- Lancez les applications sur votre téléphone mobile
- Cliquez sur l'onglet CLIENT
- Saisissez l'adresse IP de votre Raspberry Pi (dans ce projet, elle est 192.168.1.202)
- Définir le numéro de port à 5000
- Entrer le message **Sur** et cliquez sur SEND comme indiqué dans la figure 9.18 et vous devriez voir le LED pour allumer

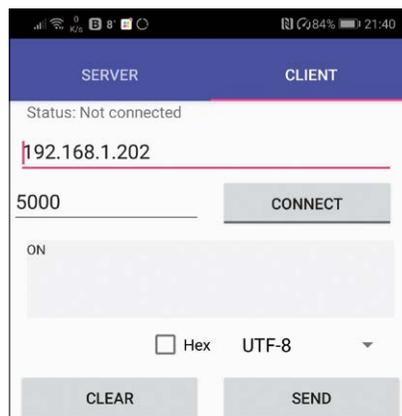


Illustration 9.18 Entrer ON pour allumer la LED

- Entrer le message **off** et cliquez sur SEND comme indiqué dans la figure 9.19 et vous devriez voir le LED pour éteindre

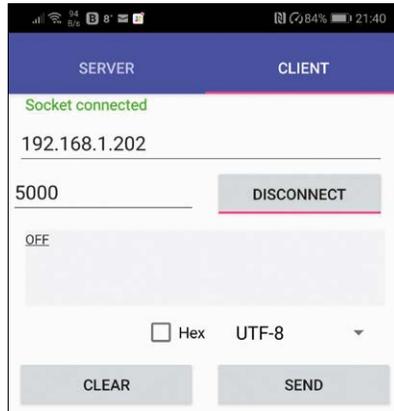


Illustration 9.19 Entrer OFF pour ÉTEINDRE la LED

9 . 6 Projet 46 – Contrôle de plusieurs LED à partir d'un téléphone mobile – Communication basée sur TCP

Description : Dans ce projet, nous allons contrôler 3 LED à partir d'un téléphone mobile comme dans le projet 43, mais maintenant nous allons utiliser le protocole TCP fiable basé sur la connexion.

Comme précédemment, les commandes valides sont :

1=ON	allumer la LED 1
2=ON	allumer la LED 2
3=ON	allumer la LED3
1=OFF	éteindre la LED 1
2=OFF	éteindre la LED 2
3=OFF	éteindre la LED 3

Viser : Le but de ce projet est de montrer comment plusieurs dispositifs peuvent être contrôlés à distance en utilisant le protocole TCP.

Schéma de principe : Le schéma du projet est présenté à la figure 9.7.

Schéma de circuit : Le schéma de circuit du projet est comme indiqué dans la figure 9.8, où les LED sont connectées aux ports GPIO GPIO 2, GPIO 3 et GPIO 4.

Programme de flux Node-RED : La figure 9.20 montre le programme de flux qui se compose de 5 nœuds : a **tcp dans** nœud pour recevoir des paquets TCP du téléphone mobile, un **fonction** noeud avec 3 sorties- met à formater les données reçues pour contrôler les 3 LED, et 3 **rpi gpio out** les nœuds qui sont connectés aux LED.



Figure 9.20 Programme de flux du projet

Les étapes sont les mêmes que dans le projet 43, mais **udp dans** Le nœud doit être remplacé par **tcp dans** nœud. Le **tcp dans** le nœud doit être configuré comme dans la figure 9.16, et le **fonction** le nœud doit être configuré comme dans Project 43.

Pour tester le projet, vous pouvez utiliser les applications **Simple TCP Socket Tester** Décrit dans le pré- vious. Par exemple, saisissez **2=ON** et LED2 doivent être activés, saisir **2=OFF** et la même LED doit être éteinte, etc.

9 . 7 Projet 47 – Envoi de la température et de l'humidité au téléphone mobile – Communication basée sur TCP

Description : Ceci est très similaire à Project 44 . Dans ce projet, nous enverrons des données de Raspberry Pi au téléphone mobile. Le programme attendra pour recevoir la commande **météo** depuis le téléphone mobile. Lorsque cette commande est reçue, la météo locale sera reçue et la température et l'humidité seront envoyées au téléphone mobile en utilisant le protocole TCP.

Le fonctionnement du programme est décrit dans le projet 44.

Viser : Le but de ce projet est de montrer comment les données peuvent être envoyées de Raspberry Pi au téléphone mobile en utilisant le protocole TCP.

Schéma de principe : Le schéma du projet est comme dans la figure 9.10.

Programme de flux Node-RED : La figure 9.21 montre le programme de flux qui se compose de 6 nœuds : a **tcp dans** nœud pour recevoir des paquets TCP du téléphone mobile, un **fonction** nœud qui vérifie quand commande **météo** est reçu, un **commutateur** nœud qui active **ouvrir- carte météorologique** node pour obtenir le bulletin météo local, un **fonction** node pour extraire les données locales de température et d'humidité du rapport météorologique, et **tcp out** nœud pour envoyer les données de température et d'humidité au téléphone mobile.

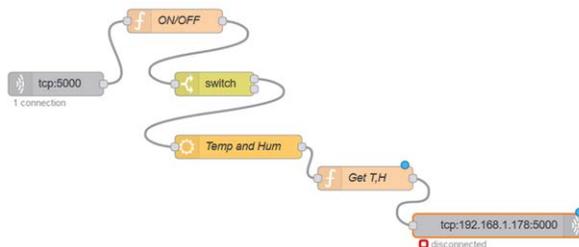


Figure 9.21 Programme de flux du projet

Les étapes sont les mêmes que dans le projet 43, sauf qu'ici le nœud TCP est utilisé à la place du nœud UDP. Le **tcp dans** Le nœud au début du programme doit être configuré comme indiqué dans la figure 9.22.

Figure 9.22 Configurer le tcp dans le nœud

La configuration des nœuds restants est la même que dans le projet 43, sauf **tcp outle** dernier nœud du programme de flux. Ce nœud doit être configuré comme indiqué sur la figure 9.23, où le Type **est défini sur Répondre au protocole TCP**.

Figure 9.23 Configurer le nœud de sortie tcp

Après avoir rejoint tous les nœuds comme indiqué dans la figure 9.21, cliquez sur **Déployer**. Vous pouvez maintenant utiliser les applications **Simple TCP Socket Tester** pour tester le programme. Les étapes sont :

- Lancer les applications
- Cliquez sur l'onglet CLIENT
- Saisissez l'adresse IP du Raspberry Pi
- Entrez le numéro de port comme 5000
- Saisir la commande weather et cliquer sur SEND

Les valeurs locales actuelles de température et d'humidité s'affichent sur votre téléphone mobile comme indiqué à la figure 9.24

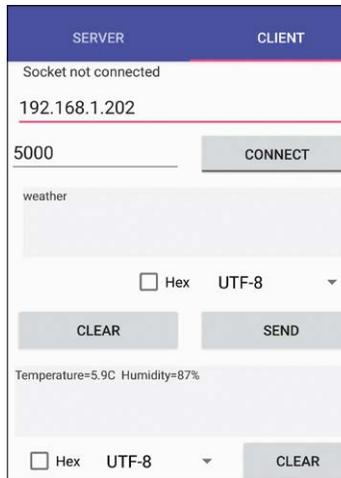


Figure 9.24 Affichage de la température et de l'humidité locales

9 8Projet 48 – Programme de clavardage – Clavardage du téléphone portable au Raspberry Pi

Description : Les programmes de chat sont très courants sur presque tous les sites de médias sociaux. Tels pro- Les messages permettent à deux personnes ou plus de communiquer en temps réel. Les- Court afin de permettre aux participants de s'adresser les uns aux autres rapidement. Certains programmes de clavardage offrent également des liens vidéo où les participants peuvent se voir tout en communiquant par texte. La plupart des programmes de chat sont basés sur le protocole UDP.

Dans ce projet, nous développons un programme de chat simple qui permet à une personne utilisant un téléphone mobile de discuter avec une autre personne utilisant un Raspberry Pi. Les nœuds du tableau de bord Node-RED sont utilisés dans ce programme pour afficher le message envoyé entre les deux parties.

Viser : Le but de ce projet est de montrer comment un simple programme de chat peut être développé en utilisant le protocole UDP sur un Raspberry Pi.

Programme de flux Node-RED : Le développement d'un programme de chat utilisant Node-RED est relatif. Ce qui est nécessaire, c'est une **udp dans** nœud pour recevoir les messages, un **udp out** un nœud pour envoyer des messages, et un outil pour afficher les messages reçus et envoyés. Par conséquent, un programme de chat est essentiellement une communication bidirectionnelle entre deux appareils sur Internet.

La figure 9.25 montre le programme de flux de notre programme de discussion. Le programme se compose de 4 nœuds : **udp dans** nœud pour recevoir des messages TCP du téléphone mobile, un tableau de bord **texte** nœud pour afficher les messages reçus, un tableau de bord **saisie de texte** nœud pour recevoir le message à envoyer au téléphone mobile, et un **udp out** nœud pour envoyer le message au téléphone mobile.



Figure 9.25 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **udp dans** nœud comme indiqué à la figure 9.26

Figure 9.26 Configurer l'udp dans le nœud

- Créer un tableau de bord **texte** nœud et configurer comme dans la figure 9.27

Figure 9.27 Configuration du nœud de texte

- Créer un tableau de bord **saisie de texte** et configurer comme dans la Figure 9.28. Notez que le paramètre Delay est réglé à 0 de sorte que les messages ne soient envoyés qu'après l'envoi du message- La touche Ter est enfoncée (sinon chaque touche sera envoyée dès qu'elle est enfoncée)

Figure 9.28 Configurer le nœud d'entrée de texte

- Créer un **udp out** nœud et le configurer comme indiqué sur la figure 9.29



Figure 9.29 Configurer le nœud de sortie udp

- Rejoindre les nœuds comme dans la figure 9.25 et cliquer **Déployer**. Pour tester le programme, utilisez les applications **UDP RECEVOIR** et **ENVOYER** comme décrit plus haut dans ce chapitre. Démarrez le tableau de bord en saisissant les informations suivantes dans votre navigateur web (remplacez l'adresse IP par votre propre adresse IP Raspberry Pi). La sortie de l'exemple sur le tableau de bord est illustrée à la figure 9.30. L'affichage correspondant sur le téléphone mobile est illustré à la figure 9.31 :

<http://192.168.1.202:1880/ui/>

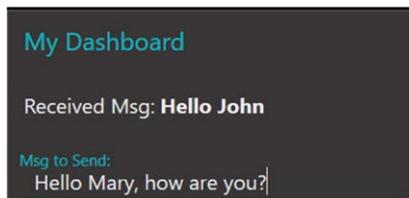


Illustration 9.30 Affichage sur le tableau de bord

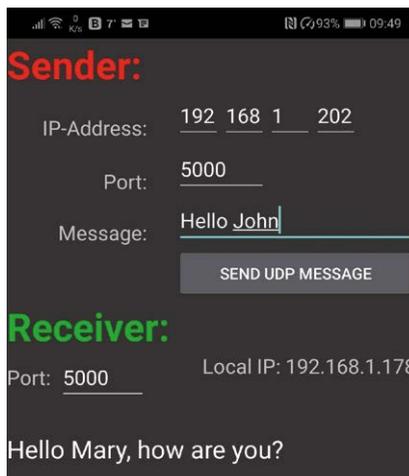


Figure 9.31 Affichage sur le téléphone mobile

Flux de programme modifié

Dans le flux de programmation de la figure 9.25, le message envoyé au téléphone mobile (Msg To Send : dans la figure 9.30) reste dans la zone de texte. Par conséquent, pour envoyer un nouveau message, nous devons supprimer le message existant. Cela peut être modifié de sorte que le message soit- automatiquement après 5 secondes d'appui sur la touche Entrée. Le programme de flux modifié est illustré à la figure 9.32.

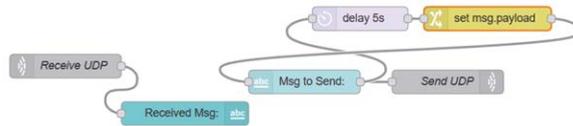


Figure 9.32 Programme de flux modifié

Dans ce programme modifié, un nœud de 5 secondes est inséré à la sortie du texte. De plus, un nœud de changement est inséré sans données (voir figure 9.33). Le message est supprimé après 5 secondes d'appui sur la touche Entrée, de sorte qu'un nouveau message peut être écrit.

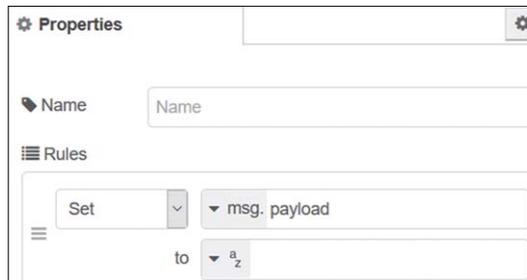


Figure 9.33 Créer un nœud de changement vide

9 . 9 Projet 49 – Utilisation du ping

Description : Ping est un utilitaire réseau utilisé pour tester l'accessibilité d'un ordinateur sur un réseau. Ping mesure le temps de trajet aller-retour pour les messages envoyés d'un ordinateur à un ordinateur de destination. Fondamentalement, un message est envoyé et le temps pour que le message fasse écho est mesuré et affiché. Le ping fonctionne si les deux ordinateurs sont disponibles sur un réseau (p. ex., Internet ou Wi-Fi). Dans ce projet, nous enverrons un message ping de notre Raspberry Pi à notre téléphone mobile et afficherons le temps d'aller-retour.

Viser : Le but de ce projet est de montrer comment l'utilitaire ping peut être utilisé dans une application Node-RED.

Programme de flux Node-RED : Le groupe de réseau des nœuds comprend un nœud ping qui peut être utilisé dans une application. La configuration la plus simple consiste à saisir l'adresse IP de l'ordinateur dans le nœud **ping** et connectez un nœud de débogage à la sortie de ce nœud. De cette façon, les données de retour ping seront affichées dans la fenêtre de débogage. Dans ce projet, nous utiliserons un tableau de bord **texte** nœud pour afficher les données de retour ping dynamiquement. La figure 9.34 montre le programme de flux du projet.

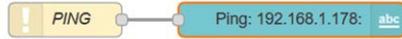


Figure 9.34 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **ping** node en cliquant, en le faisant glisser et en le déposant depuis le réseau palette. Double-cliquez pour configurer ce nœud comme indiqué dans la figure 9.35. Ici, le nœud est configuré pour produire une sortie toutes les 5 secondes. L'adresse IP du téléphone portable de l'auteur est donnée comme **Cible** (vous devez choisir votre propre adresse IP). Notez que vous pouvez également saisir une adresse web valide, par ex. www.bbc.co.uk

Figure 9.35 Configuration du nœud ping

- Créer un **texte** nœud en cliquant dessus, en le faisant glisser et en le déposant dans le tableau de bord. Configurez ce nœud comme indiqué sur la figure 9.36

Figure 9.36 Configuration du nœud de texte

- Rejoindre les deux nœuds et cliquer **Déployer**. Démarrez votre tableau de bord. Vous devriez voir le temps d'aller-retour comme indiqué à la figure 9.37

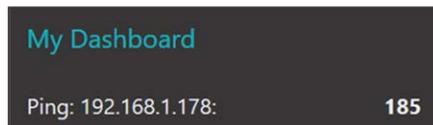


Figure 9.37 Exemple d'affichage du temps de trajet aller-retour

9 . 10 Résumé

Dans ce chapitre, nous avons appris comment utiliser les nœuds UDP et TCP dans des applications basées sur Node-RED- tions de communiquer avec un téléphone mobile.

Dans le prochain chapitre, nous allons examiner les nœuds de stockage et apprendre comment stocker des données dans un fichier.

Chapitre 10 • Nœuds de stockage

10 . 1 Aperçu

Dans le dernier chapitre, nous avons appris comment utiliser les nœuds Node-RED pour la communication réseau basée sur UDP et TCP. Dans ce chapitre, nous allons examiner l'important sujet de la façon de stocker des données dans des fichiers sur une carte SD Raspberry Pi, en utilisant les nœuds Node-RED.

Les données sur Raspberry Pi sont normalement stockées sur la carte SD qui est également chargée avec l'oper- Le système. Le dossier utilisateur par défaut est /home/pi. Dans ce chapitre, nous allons créer des fichiers dans le répertoire par défaut et ensuite stocker les données dans ces fichiers.

10 2Projet 50 – Stockage de la température et des données d'humidité dans un fichier

Description : Dans ce projet, nous allons lire le bulletin météo local à chaque minute- en **openweathermap** et ensuite extraire les données de température et d'humidité. Ces données seront alors stockées dans un fichier appelé **temphum . txt** sur la carte SD Raspberry Pi avec l'heure à laquelle les données ont été extraites.

Programme de flux Node-RED : La figure 10.1 montre le programme de flux qui se compose de 5 nœuds : **injecter** noeud qui se répète toutes les minutes, un **openweathermap** noeud qui reçoit le bulletin météo local, un **fonction** noeud qui formate les données, un **changement** noeud qui spécifie le nom du fichier, et une **fichier** noeud qui stocke (ajoute) les données dans le fichier spécifié.

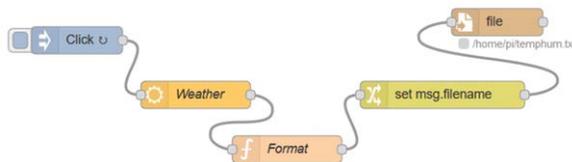


Figure 10.1 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud, nommez-le comme **Cliquez**, et le réglez pour répéter chaque minute
- Créer un **openweathermap** nœud et le nommer **Météo**. Entrez la clé API
- Créer un **fonction** noeud, nommez-le comme **Format**, et entrez les instructions suivantes dans ce nœud. Variable **premier** est initialisé à 0 comme variable de contexte. Ceci est fait pour que nous puissions insérer une rubrique dans les données la première fois que la fonction est exécutée. Sur ce, la première est non nulle et les données réelles sont stockées dans le fichier. L'heure locale est extraite et stockée dans une variable appelée **Tim** qui est stocké dans le fichier avec chaque élément de données.

```

var first = context.get('first') || 0; si (first == "0")
{
    context.set('first',1);
    Tous = " Heure      Température, humidité";
}

var LocalTime = new Date();
var Tim = LocalTime.toLocaleTimeString(); Temp =
msg.payload.tempc;
Hum = msg.payload.humidity;
if(first != "0")
    Tous = Tim + " " + Temp + "C msg.payload = All      " + Hum + "%";
retourner le message;

```

- Créer un **changement** noeud et ensemble **msg . nom de fichier** au nom du fichier où nous voulons que les données soient stockées, c.-à-d. /home/pi/temphum.txt. Notez que nous aurions pu définir le nom de fichier dans le nœud **fichier** . Si msg.filename est utilisé, le fichier sera fermé après chaque écriture. Configurez ce nœud comme indiqué dans la Figure 10.2

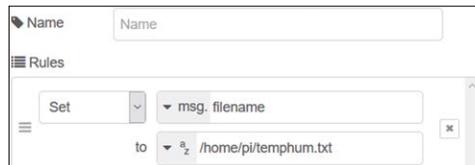


Figure 10.2 Configuration du changement de nœud

- Créer un **fichier** nœud et le configurer comme indiqué dans la figure 10.3. Notez que **Action** est défini sur Ajouter au fichier, et une nouvelle ligne doit être ajoutée après chaque enregistrement

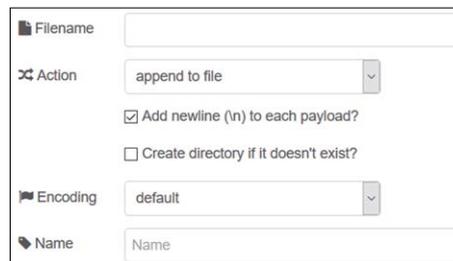


Figure 10.3 Configuration du fichier de nœud

- Rejoindre tous les nœuds comme indiqué dans la figure 10.1, et cliquez sur **Déployer**.
- Cliquez sur le bouton de la **injecter** noeud. Un nouveau fichier nommé temphum.txt sera créé dans le dossier /home/pi et les données locales sur la température et l'humidité seront stockées dans le fichier toutes les minutes.

La figure 10.4 montre les fichiers (en utilisant la commande **ls**) dans le dossier utilisateur par défaut /home/pi. Remarquez que le fichier tempnum.txt se trouve dans ce dossier. Le contenu du fichier tempnum.txt (commande **cat tempnum.txt**) sont illustrés à la figure 10.5.

```
pi@raspberrypi:~ $ ls
book      Downloads  Music      Pictures  tempnum.txt
Desktop   MagPi     node_modules Public    Templates
Documents mu_code   package-lock.json Scratch   Videos
pi@raspberrypi:~ $
```

Figure 10.4 Fichiers dans le dossier /home/pi]

```
pi@raspberrypi:~ $ cat tempnum.txt
Time      Temperature Humidity
16:40:30  7.2C       87%
16:41:30  7.2C       93%
16:42:30  7.2C       93%
16:43:30  7.2C       93%
16:44:31  7.2C       93%
16:45:30  7.1C       93%
16:46:31  7.2C       93%
16:47:30  7.2C       93%
16:48:30  7.1C       93%
16:49:30  7.1C       93%
16:50:30  7.1C       93%
16:51:30  7.1C       93%
16:52:30  7.1C       93%
```

Figure 10.5 Contenu du fichier tempnum.txt

10 . 3 Projet 51 – Lecture du contenu d’un fichier

Description : Dans le projet précédent, nous avons vu comment stocker des données dans un fichier. Dans ce pro- objet, nous allons lire le contenu du fichier que nous avons créé dans le projet précédent.

Programme de flux Node-RED : Nœud **déposer** est utilisé pour lire le contenu d’un fichier soit sous forme de chaîne ou de tampon binaire. La figure 10.6 montre le programme flow qui se compose de 3 nœuds : un **injecter** noeud, un **déposer** nœud pour lire le contenu du fichier, et un **déboguer** nœud qui décompose- lit le contenu du fichier.



Figure 10.6 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud, nommez-le comme **Cliquez**
- Créer un **déposer** nœud et le configurer comme indiqué dans la Figure 10.7. Le **Nom de fichier** est défini sur /home/pi/tempnum.txt et le **Production** est défini comme une chaîne de caractères

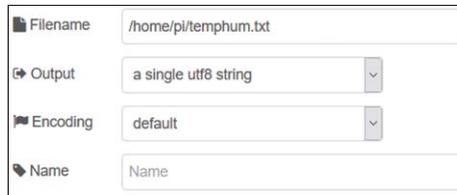


Figure 10.7 Configurer le fichier dans le nœud

- Créer un **débogueur** node et définir son **Production** pour déboguer la fenêtre. Rejoindre les nœuds et cliquer **Déployer**
- Cliquez sur le bouton de la **injecter** node. Vous devriez voir le contenu du fichier dis-
Lu dans la fenêtre de débogage comme indiqué sur la figure 10.8. Notez ici que tout le
contenu du fichier est affiché et que le texte n'est pas aligné correctement.

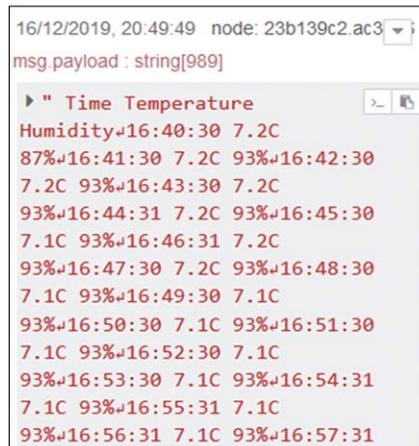


Figure 10.8 Contenu du fichier affiché dans la fenêtre de débogage

- Maintenant, double-cliquez sur le **débogueur** modifier le nœud et **Production** vers la console système. Cliquez sur le bouton du nœud d'injection. La console système est l'affichage du mode de commande du Raspberry Pi et vous devriez voir que le contenu du fichier s'affiche correctement comme indiqué dans la figure 10.9

```
To find more nodes and example flows - go to h!

Starting as a systemd service.
16 Dec 20:53:57 - [info] [debug:23b139c2.ac37b!
  Time    Temperature  Humidity
16:40:30    7.2C          87%
16:41:30    7.2C          93%
16:42:30    7.2C          93%
16:43:30    7.2C          93%
16:44:31    7.2C          93%
16:45:30    7.1C          93%
16:46:31    7.2C          93%
16:47:30    7.2C          93%
16:48:30    7.1C          93%
```

Illustration 10.9 Affichage du fichier sur la console système

- Le fichier peut être lu dans différents formats. Double-cliquez sur le nœud **déposer** et modifier le **Production** à un message par ligne. Changez également le **déboguer** nœud à afficher dans la fenêtre Debug. Cliquez sur le bouton **injecter** node et vous devriez voir le contenu du fichier affiché dans la fenêtre Debug comme indiqué sur la figure 10.10

```

"16:40:30 7.2C 87%"
16/12/2019, 20:57:46 node: 23b13
msg.payload : string[28]
"16:41:30 7.2C 93%"
16/12/2019, 20:57:46 node: 23b13
msg.payload : string[28]
"16:42:30 7.2C 93%"
16/12/2019, 20:57:46 node: 23b13
msg.payload : string[28]
"16:43:30 7.2C 93%"
16/12/2019, 20:57:46 node: 23b13
msg.payload : string[28]
"16:44:31 7.2C 93%"

```

Figure 10.10 Affichage du contenu du fichier en mode msg par ligne

- Le fichier peut également être affiché en format binaire en définissant la **Production** de nœud **déposer** à un seul objet Buffer. Dans ce mode, la sortie est représentée sur la figure 10.11. Cet affichage peut ne pas avoir beaucoup de sens. Cliquez sur les nombres et vous devriez voir le contenu du fichier affiché correctement comme indiqué dans la figure 10.12.

```

16/12/2019, 21:00:28 node: 23b139c2.ac37b6
msg.payload : buffer[989]
▶ [ 32, 32, 84, 105, 109, 101, 32,
32, 32, 32 ... ]

```

Figure 10.11 Affichage du contenu du fichier en mode objet tampon unique

16/12/2019, 21:00:28 node: 23b139c2.ac3

msg.payload : buffer[989]

▼ buffer[989] string

Time	Temperature	Humidity
16:40:30	7.2C	87%
16:41:30	7.2C	93%
16:42:30	7.2C	93%
16:43:30	7.2C	93%
16:44:31	7.2C	93%
16:45:30	7.1C	93%
16:46:31	7.2C	93%
16:47:30	7.2C	93%
16:48:30	7.1C	93%
16:49:30	7.1C	93%

Figure 10.12 Affichage correct du fichier dans la fenêtre de débogage

En résumé, la lecture d'un fichier comporte quatre options :

chaîne utf-8 : le fichier entier est converti en une chaîne de caractères et lu et présenté comme une chaîne de texte, et le fichier ne peut pas être traité avant qu'il soit terminé. Ici, les données doivent contenir des données textuelles.

une ligne de MSG : les données sont lues et présentées sous forme de lignes de texte, et le fichier peut être traité pendant la lecture

un seul objet tampon : les données sont lues comme une séquence d'octets bruts et com- Toutes les données sont lues, elles ne peuvent pas être traitées avant d'être complétées

un flot de tampons : les données sont lues et présentées comme une séquence d'octets bruts, et il peut être traité tout en la lisant

Nous pouvons facilement convertir les données binaires en format de chaîne en utilisant la fonction `toString()`. Pour ex- ample, dans le code de fonction suivant, nous pouvons définir la **déposer** format à **un flot de tamponset** convertir les données en format chaîne :

```
var data = msg.payload; srng =
data.toString(); msg.payload =
srng; return msg;
```

10 . 4 Projet 52 – Lecture d'un fichier ligne par ligne

Description : Dans le projet précédent, nous avons vu comment lire le contenu d'un fichier- en **déposer** node. Le problème était que tout le contenu du fichier était lu et qu'il n'était donc pas possible de traiter le fichier pendant qu'il était en lecture. Dans ce projet, nous allons lire le fichier créé dans Project 50 et extraire et afficher uniquement le champ de température. **Programme de flux Node-RED** : Nœud **déposer** est utilisé pour lire le contenu du fichier comme dans la

projet précédent. La figure 10.13 montre le programme de flux qui se compose de 5 nœuds : un **injecternœud**, un déposer **nœud pour lire le contenu du fichier**, un retard **nœud pour produire un message par seconde** fonction **nœud pour extraire le champ de température des données**, et une débogueurle **nœud qui affiche les données extraites dans la fenêtre Debug**.

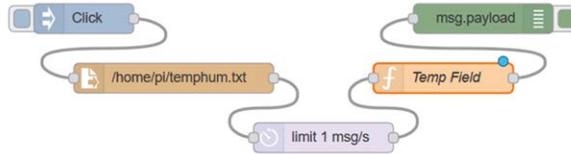


Figure 10.13 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** nœud, nommez-le comme **Cliquez**
- Créer un **déposer** nœud et définir **Nom de fichier** à /home/pi/temphum.txt et le **Production** à un msg par ligne
- Créer un **retard** le nœud définit son **Action** à la limite de débit avec 1 msg par seconde. Config- Effectuer ce nœud comme indiqué à la figure 10.14

```
pi@raspberrypi:~ $ ls
book      Downloads  Music      Pictures  temphum.txt
Desktop   MagPi     node_modules  Public   Templates
Documents mu_code   package-lock.json  Scratch  Videos
pi@raspberrypi:~ $
```

Figure 10.14 Configuration du nœud de retard

- Créer un **fonction** nœud nommé **Champ temporaire** et entrez les instructions suivantes dans ce nœud. Ce nœud va extraire et sortir le champ de température des données :

```
T = msg.payload.substr(12,4);
msg.payload = T;
retourner le message;
```

- Créer un **débogueur** node pour afficher la sortie dans la fenêtre Debug
- Rejoindre les nœuds et cliquer **Déployer**. Cliquez sur le bouton de la **injecter** nœud. Vous devriez voir la température affichée dans la fenêtre de débogage comme indiqué sur la figure 10.15

```
16/12/2019, 22:17:32 node: 5df26784.abb36
msg.payload : string[4]
" 7.2"
16/12/2019, 22:17:33 node: 5df26784.abb36
msg.payload : string[4]
" 7.2"
16/12/2019, 22:17:34 node: 5df26784.abb36
msg.payload : string[4]
" 7.2"
16/12/2019, 22:17:35 node: 5df26784.abb36
msg.payload : string[4]
" 7.2"
16/12/2019, 22:17:36 node: 5df26784.abb36
msg.payload : string[4]
" 7.2"
```

Figure 10.15 Affichage de la température

10 . 5 Résumé

Les fichiers dans Raspberry Pi sont normalement stockés sur la carte SD qui stocke également le système d'exploitation. Dans ce chapitre, nous avons appris comment utiliser les nœuds de fichiers pour écrire et lire des fichiers.

Dans le prochain chapitre, nous examinerons les nœuds utilisés pour la communication série et développerons des projets qui utilisent la communication série.

Chapitre 11 • Communication en série

11 . 1 Aperçu

Dans le dernier chapitre, nous avons appris à écrire et lire des fichiers en utilisant Node-RED, et nous avons développé plusieurs projets sur ce sujet. Ce chapitre est sur serial communication où nous apprendrons comment deux appareils peuvent communiquer en utilisant la communication série. La communication série a été l'une des premières méthodes d'établissement de la communication entre deux appareils. De nos jours, ce type de communication n'est pas très utilisé, surtout après l'invention de l'USB.

Dans sa forme la plus simple, la communication série entre deux appareils nécessite seulement 3 fils : TX (ou TXD, transmettre), RX (ou RXD, recevoir) et GND (terre). La figure 11.1 montre le modèle- forme la plus simple de communication série entre deux dispositifs où les lignes TX et RX sont croisées entre les dispositifs.

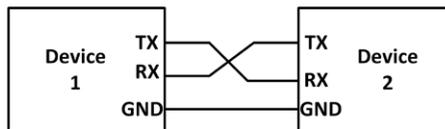


Figure 11.1 Lignes de communication série

Dans la communication série, les octets de données sont décomposés en bits et ces bits sont envoyés et reçus avec les temps corrects. Les données sont envoyées ou reçues sous forme de trames lorsqu'une trame est composée des bits suivants :

- Bit de démarrage
- 7 ou 8 bits de données
- Bit de parité optionnel
- Bit d'arrêt

Le bit de parité est utilisé pour la vérification d'erreur sur un bit et ce bit n'est pas utilisé très souvent. Les données sont normalement envoyées avec 8 bits. Par conséquent, une trame de données se compose de 10 bits, où un appareil qui souhaite communiquer avec un autre appareil envoie un bit de départ, suivi de 8 bits de données et d'un bit d'arrêt.

Le moment du bit, également connu sous le nom de débit en bauds, est très important dans la communication série et les deux côtés doivent avoir exactement les mêmes débits en bauds. Les taux de bauds typiques sont 9600, 19200, 38400, etc. Le taux de bauds détermine effectivement le nombre d'octets qui peuvent être envoyés en une seconde. Par exemple, à 9600 Baud nous pouvons envoyer ou recevoir 960 octets (ou charac-) par seconde.

Les anciens appareils de communication série utilisaient 12V comme niveau de signal, où -12 était connu sous le nom de Mark et +12 sous le nom d'Espacement. Ces niveaux de signal ont été spécifiés par la norme RS232 protocol. Deux types de connecteurs ont été utilisés dans les applications RS232 : 25-way connecteur de type D (vous pouvez voir ces connecteurs sur des ordinateurs très anciens), et 9-way connecteur de type D (usu- On trouve souvent sur les vieux ordinateurs portables). Aujourd'hui, la plupart des microcontrôleurs fonctionnent avec

+3,3V. Par conséquent, les niveaux de signaux de communication série ont été modifiés pour être compatibles avec les entrées actuelles du microcontrôleur. Parfois, les nouveaux niveaux sont également appelés à être compatibles TTL. Il est important d'être prudent et de ne pas connecter une ligne RS232 à une entrée de crocodyl que la haute tension endommagera le circuit du microcontrôleur.

Dans ce chapitre, nous utiliserons les nœuds d'entrée et de sortie série fournis par Node-RED pour communiquer avec le monde extérieur via le port série du Raspberry Pi.

11 2Projet 53 – Communication avec Arduino sur ligne série

Description : Dans ce projet, nous utilisons un Arduino Uno et un Raspberry Pi. Un- perature est connecté à l'une des entrées analogiques de l'Arduino. L'Arduino lit la température ambiante toutes les 5 secondes et l'envoie au Raspberry Pi sur la ligne série. Le Raspberry Pi lit la température de la ligne série et l'affiche dans le De- fenêtre de bug.

Viser : Le but de ce projet est de montrer comment la communication série peut être établie en utilisant le Node-RED serial in node.

Renseignements généraux : L'ordinateur Arduino peut être programmé pour utiliser soit son port de communication série dédié (broches de port GPIO 0 et 1) ou n'importe laquelle de ses broches de port peuvent être utilisées dans ce qui est connu comme communication série logicielle. Dans la communication série logicielle, tout le timing se fait dans le logiciel. Le seul inconvénient de l'utilisation du logiciel au lieu du matériel dans la communication série est que le débit en bauds maximum est plutôt faible- Dans cette application, nous utiliserons 9600 comme débit en baud. Il n'y a aucun problème de synchronisation à ce faible débit en baud.

Les ordinateurs Raspberry Pi ont deux UART intégrés : un PL011 et un mini UART. Ces modules sont implémentés en utilisant différents blocs de matériel, donc ils ont des caractères légèrement différents- téristiques. Comme les deux sont des appareils 3,3V, il faut faire particulièrement attention lors de la connexion à d'autres lignes de communication série. Sur les Raspberry Pis équipés de modules sans fil/Bluetooth (par ex. Raspberry Pi 3, Zero W, 4, etc.), l'UART PL011 est connecté par défaut au module Bluetooth, tandis que le mini UART est l'UART principal avec la console Linux dessus. Dans tous les autres modèles, le PL011 est utilisé comme UART principal. Par défaut, /dev/ttyS0 fait référence au mini-UART et /dev/ttyAMA0 à la PL011. La console Linux utilise l'UART primaire qui dépend du modèle de Raspberry Pi utilisé. Aussi, si activé, /dev/serial0 fait référence à l'UART primaire (si activé), et si activé, /dev/serial1 fait référence à l'UART secondaire

Par défaut, l'UART principal (serial0) est assigné à la console Linux. Utiliser le port série à d'autres fins nécessite que cette configuration par défaut soit modifiée. Au démarrage, systemd vérifie la ligne de commande du noyau Linux pour toutes les entrées de console et utilisera la console de- Pour arrêter ce comportement, il faut supprimer les paramètres de la console série de la ligne de commande. Ceci est facilement fait en utilisant l'utilitaire raspi-config en sélectionnant option **5** (Options d'interface) et puis **P6** (Série) et sélectionnez **Non**. Quittez raspi-config et redémarrez votre Raspberry Pi. Vous devriez maintenant pouvoir accéder au port série (n'oubliez pas de- Vous pouvez régler la console une fois le processus terminé).

Dans les Raspberry Pi 3 et 4, le port série (/dev/ttyS0) est acheminé vers deux broches GPIO 14 (TXD) et GPIO 15 (RXD) sur la tête. Les modèles antérieurs au modèle 3 utilisent ce port pour le Bluetooth. Au lieu de cela, un port série est créé dans le logiciel (/dev/ttyS0).

Nous pouvons facilement rechercher les ports série disponibles en entrant la console suivante commande :

```
dmesg | grep tty
```

La dernière ligne dans la sortie ci-dessous indique que la console est activée sur le port série ttyS0. console [ttyS0] activé

Dans ce livre, nous utilisons Raspberry Pi 4. Le port série est : /dev/ttyS0. Si vous utilisez un modèle antérieur au 3, utilisez le port série nommé : /dev/ttyAMA0

Schéma de principe : La figure 11.2 montre le schéma du projet, où la température- Le capteur de température est connecté à l'Arduino Uno.

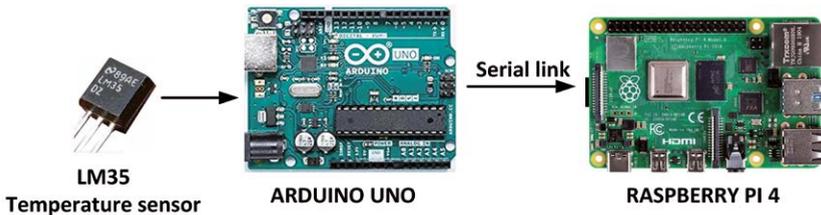


Figure 11.2 Schéma du projet

Schéma de circuit : Le schéma du projet est présenté à la figure 11.3. Dans ce projet, une puce de température analogique de type LM35DZ est utilisée et sa broche de sortie est connectée à l'entrée analogique A0 de l'Arduino Uno. Il s'agit d'un dispositif à 3 broches fonctionnant avec +5V, et sa broche- ration est représentée à la figure 11.3. La tension de sortie du LM35DZ est directement proportionnelle à la température et est donnée par la relation :

$$T = V_o / 10$$

Où, T est la température mesurée en degrés centigrades et Vo est la tension de sortie du capteur en millivolts. Par exemple, si la tension de sortie est de 200mV, alors la tension mesurée est de 20°C et ainsi de suite.

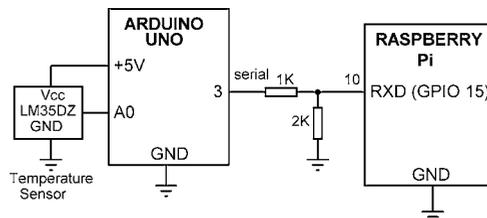


Figure 11.3 Schéma du projet


```

/*-----
 * Ce programme a reçu la température ambiante d'un
 * Capteur de température analogique LM35DZ tye connecté à la broche
 * A0 de l'Arduino Uno. La température est envoyée à
 * port 3 en série à 9600 Baud
 *
 * Fichier : ArduinoTemp
 *-----*/ #include <SoftwareSerial.h>
SoftwareSerial MySerial(2, 3); // RX, TX float mV, T;
char Temp[10];

void setup()
{
  MySerial.begin(9600);
}

boucle vide()
{
  int sensor = analogRead(A0); mV = // Lecture de la
  sensor * 5000.0 / 1024.0; T = mV / 10.0; // température // en mV
  dtostrf(T, 4,2,Temp); // dans Centigrade
  MySerial.print(&quot;T=&quot;); // Convertir en chaîne //
  MySerial.print(Temp); // Afficher T=
  MySerial.println(&quot;C&quot;); // Valeur d'affichage
  delay(5000); // Afficher C
  // Attendre 5 secondes
}

```

Figure 11.5 Programme Arduino Uno

Nous pouvons utiliser le moniteur série IDE Arduino pour confirmer que le programme fonctionne comme- La température est convertie en degrés centigrades correctement.

Programme de flux Node-RED : Supprimez le paramètre de la console série comme décrit plus haut dans ce chapitre. La figure 11.6 montre le programme flow qui se compose de seulement 2 nœuds **en série dans** nœud qui reçoit des données en série et un **déboguer le nœud** qui est utilisé pour afficher les données reçues.

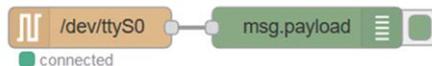


Figure 11.6 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **en série dans** Node et configurer comme indiqué dans la figure 11.7 (cliquez sur le stylo après le port série), cliquez **Mise à jour** et **Terminé**. Les **Port série** est réglé sur /dev/ttyS0, **Débit en bauds** est réglé à 9600, 8 bits de données, pas de parité et 1 bit d'arrêt (ces paramètres doivent être identiques aux paramètres du côté émetteur)

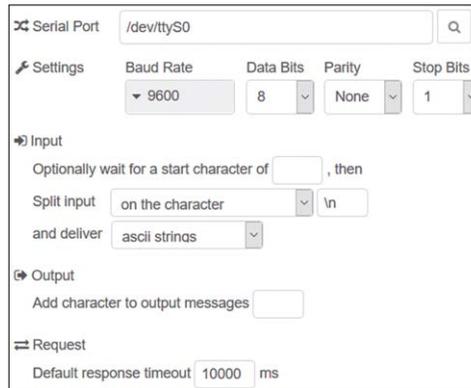


Figure 11.7 Configuration du nœud serial dans

- Créer un **déboguer** nœud, rejoindre les nœuds et cliquer **Déployer**. Vous devriez voir le tem- perature reçue de l'Arduino Uno affichée dans la fenêtre de débogage comme indiqué sur la figure 11.8



Figure 11.8 Température affichée dans la fenêtre de débogage

Programme de flux modifié 1

Nous pouvons modifier le programme de débit donné dans la figure 11.6 afin que la température reçue soit affichée à l'aide d'un tableau de bord **texte** nœud. Le nouveau programme de flux est représenté à la figure 11.9



Figure 11.9 Programme de débit modifié

Définir le **Étiquette** champ de votre **texte** nœud à chaîne **Température** : Vous devriez commencer votre Dash- carte. La figure 11.10 montre la température affichée avec le **texte** nœud.

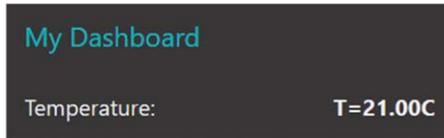


Figure 11.10 Affichage de la température à l'aide d'un nœud texte

Programme de débit modifié 2

Nous pouvons modifier le programme de flux à nouveau afin que la température reçue soit parlée sur les haut-parleurs du PC en utilisant un nœud audio Dashboard. Le nouveau programme de débit est représenté dans la figure- ure 11.11.



Figure 11.11 Programme de débit modifié

Les étapes sont les suivantes :

- Créer un **en série dans** nœud comme précédemment
- Créer un **fonction** et entrez les instructions suivantes dans cette fonction. Si par exemple, la température est de 20,00oC, alors cette fonction enverra le texte **La température est maintenant de vingt degrés centigrades** pour reliaer l'audio de sorte qu'il puisse être entendu sur les haut-parleurs du PC

```
var T = msg.payload.substr(2, 5);
md = {charge utile : « La température est maintenant de " + T + « degrés centigrades » } ;
renvoie [md];
```

- Créer un **sortie audio** node et cocher la case : Lire l'audio lorsque la fenêtre n'est pas au centre
- Rejoindre les nœuds, cliquer **Déployer** et démarrez votre tableau de bord. Vous devriez entendre le- rÃ©aliser la température parlée par les haut-parleurs de votre PC

11 . 3 Projet 54 – Réception de données en série – Réception de données GPS

Description : Dans ce projet, un module récepteur GPS est connecté à l'entrée série du Raspberry Pi. Le **en série dans** node est utilisé pour recevoir et afficher les phrases GPS dans la fenêtre Debug.

Viser : Le but de ce projet est de montrer comment les données en série peuvent être reçues à l'aide de nœud **en série dans**.

Renseignements généraux : Les récepteurs GPS reçoivent des données géographiques du satellite GPS- lites et fournissent des informations précises sur la position de l'utilisateur sur Terre. Ces satellites tournent autour de la Terre à une altitude d'environ 20000 km et effectuent deux orbites complètes chaque jour. Pour qu'un récepteur puisse déterminer sa position, il est nécessaire que le récepteur communique avec au moins 3 satellites. Par conséquent, si le récepteur n'a pas une vue dégagée du ciel, il ne sera peut-être pas possible de déterminer sa position sur la Terre. Dans certaines applications, des antennes externes sont utilisées pour recevoir même les signaux faibles provenant des satellites GPS.

Les données envoyées par un récepteur GPS sont en format texte et sont connues sous le nom de NMEA Sentences. Chaque phrase NMEA commence par un caractère \$ et les valeurs dans une phrase sont séparées par des virgules. Voici quelques-unes des phrases NMEA renvoyées par un récepteur GPS :

\$ GPGLL : Cette phrase renvoie la latitude et la longitude géographiques locales

\$ GPRMC : Cette phrase renvoie la latitude et la longitude, la vitesse, l'angle de la piste, la date, l'heure et la variation magnétique.

\$ GPVTG : Cette phrase correspond à la trajectoire, à la trajectoire magnétique et à la vitesse du sol.

\$ GGA : Cette phrase renvoie la latitude et la longitude géographiques locales, le temps, la qualité de la correction, le nombre de satellites suivis, la dilution horizontale de la position, l'altitude, la hauteur du géoïde et les données DGPS

\$ GPGSV : Il y a 4 phrases avec ce titre. Ces phrases renvoient le nombre de satellites en vue, le numéro du satellite, l'élévation, l'azimut et le SNR.

Dans ce projet, la carte GPS Click (www.mikroe.com) est utilisée. Il s'agit d'un petit récepteur GPS (voir la figure 11.12) qui est basé sur le GPS de type LEA-6S. Cette carte fonctionne avec +3.3V et fournit deux types de sorties : I²C ou série. Dans ce projet, la sortie série par défaut est utilisée et fonctionne à une vitesse de 9600 Baud. Une antenne dynamique externe peut être utilisée. Le conseil d'administration a été chargé d'améliorer la réception des signaux pour une utilisation à l'intérieur ou dans des endroits où il n'y a pas de vue dégagée sur le ciel.



Illustration 11.12 Carte GPS Click

La figure 11.13 montre la liste complète des phrases NMEA sorties de la carte GPS Click à chaque seconde.

```
$GPGLL,5127.3917,N,00003.13141,E,10534.00,A,A*67
$GPRMC,05305.00,A,5127.35909,.0003,13148,E,0.030,.270919,.,,A*7E
$GPVTG,.T,.M,0030,N,0.055,K,A*20
$GGGA,105305.00,5127.35909,N,00003.13148,E,1.09,1.18,46.5,M,45.4,M,.,*66
$GPRSA,A,3,01.32,08,28,18,03,22,14,11,.,.,2,12,1,18,1,76*06
$GPGSV,4,1,13,01,7,304,40,03,40,224,31,08,38,165,32,10,05,054,*,77
$GPGSV,4,2,13,11,83,217,3,14,39,094,24,17,17,314,22,18,73,091,41*76
$GPGSV,4,3,13,22,63,219,33,24,1,002,.,27,05,150,.,28,30,284,28*7F
$GPGSV,4,4,13,32,34,063,35*4E
```

Figure 11.13 Sortie des phrases NMEA à partir de la carte GPS Click

La carte GPS Click est un module biligne à 2 broches sur 8 et présente la configuration de broche suivante (la broche 1 est la broche supérieure gauche du module) :

- | | |
|----------------------|----------------------|
| 1 : Aucune connexion | 16 : Pas de |
| 2 : Réinitialisation | connexion 15 : Pas |
| 3 : Pas de connexion | de connexion 14 : TX |
| 4 : Pas de connexion | 13 : RX |
| 5 : Pas de connexion | 12 : SCL |
| 6 : Pas de connexion | 11 : SDA |
| 7 : +3.3V | 10 : Aucune |
| 8 : GND | connexion 9 : GND |

En mode série, seules les broches suivantes sont nécessaires : +3.3V, GND, TX. Dans ce projet, une antenne dynamique externe est attachée à la carte GPS Click comme elle était utilisée à l'intérieur.

Schéma de circuit : La figure 11.14 montre le schéma du projet. Le TX sort en série- La broche de mise en place de la carte GPS Click est connectée à l'entrée RXD du Raspberry Pi. Comme les deux broches TX et RXD fonctionnent avec +3.3V, il n'est pas nécessaire de baisser la tension pour l'entrée Raspberry Pi.

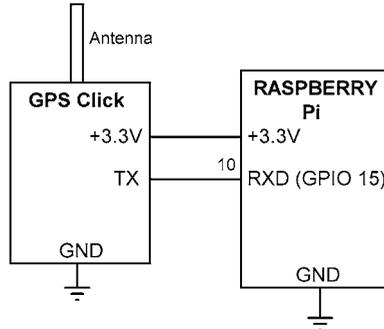


Figure 11.14 Schéma du projet

Programme de flux Node-RED : Supprimez le paramètre de la console série comme décrit plus haut dans ce chapitre. La figure 11.15 montre le programme flow qui se compose de seulement 2 nœuds **en série** dans un nœud qui reçoit les données GPS en série, et un déboguer le nœud **qui est utilisé pour afficher les données reçues**.

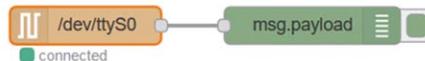


Figure 11.15 Programme de flux du projet

Configurer le nœud de série et le nœud de débogage exactement comme dans le projet précédent. Vous devriez voir les phrases NMEA affichées dans la fenêtre de débogage comme indiqué à la figure 11.16. Notez que seules certaines phrases NMEA sont présentées dans cette figure.

```

17/12/2019, 22:17:07 node: fa4d9c5d.7a2b28
msg.payload : string[52]
▶
"$GPGLL,5127.36351,N,00003.12720,E,22170.00,A,0.0,M,0.0,0000.0,000.0,M,0.0,0000.0,0000.0"

17/12/2019, 22:17:07 node: fa4d9c5d.7a2b28
msg.payload : string[68]
▶
"$GPRMC,221707.00,A,5127.36340,N,00003.12720,E,0.0,A,0.0,M,0.0,0000.0,000.0,M,0.0,0000.0,0000.0"

17/12/2019, 22:17:07 node: fa4d9c5d.7a2b28
msg.payload : string[35]
▶ "$GPVTG,,T,,M,0.098,N,0.181,K,A*2A"

17/12/2019, 22:17:07 node: fa4d9c5d.7a2b28
msg.payload : string[74]
▶
"$GPGGA,221707.00,5127.36340,N,00003.12720,E,1,00,0.0,00.0,M,0.0,0000.0"

17/12/2019, 22:17:07 node: fa4d9c5d.7a2b28
msg.payload : string[62]
▶
"$GPGSA,A,3,25,31,14,23,26,16,05,21,02,1"
    
```

Figure 11.16 Affichage des phrases NMEA dans la fenêtre de débogage

\$GPGLL est l'une des phrases NMEA couramment utilisées. Cette phrase est affichée à la figure 11.16 comme suit :

```
$GPGLL,5127.37032,N,00003.12782,E,221918.00,A,A*61
```

Les champs de cette phrase peuvent être décodés comme suit :

GLL	Position géographique, latitude et longitude
5127.37032	Latitude 51 deg, 27.3702 min. Nord Longitude 0
00003.12782	deg, 3.12782 min. Est
221918	Correction prise à 22L19L18 UTC
A	Données actives (ou V pour void)
*61	données de somme de contrôle

Notez que les champs sont séparés par des virgules. La validité des données est indiquée par les lettres A ou V dans les données, où A indique que les données sont valides et V indique que les données ne sont pas valides.

11 . 4 Projet 55 – Réception de données GPS – Extraction de la latitude et de la longitude

Description : Ce projet est similaire au précédent, mais ici la latitude, sa direction, la longitude et sa direction sont extraites et affichées dans la fenêtre Debug.

Viser : Le but de ce projet est de montrer comment les valeurs de latitude et de longitude peuvent être extraites d'une phrase NMEA. La phrase \$GPGLL est utilisée dans cet exemple qui inclut à la fois les informations de latitude et de longitude.

Schéma de circuit : Le schéma du projet est identique à celui de la figure 11.14.

Programme de flux Node-RED : Supprimez le paramètre de la console série comme décrit plus haut dans ce chapitre. La figure 11.17 montre le programme flow qui se compose de 4 nœuds **en série** dans nœud qui reçoit les données GPS en série, un **commutateur** Pour extraire uniquement la phrase \$GPGLL des phrases NMEA, un **fonction** nœud pour extraire la latitude et la longitude de \$GP- GLL, et une **déboguer le nœud** qui est utilisé pour afficher les données dans la fenêtre de débogage.

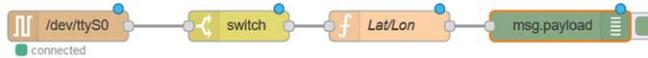


Figure 11.17 Programme de flux du projet

Les étapes sont les suivantes :

- Configurer le **en série dans** nœud comme précédemment
- Créer un **commutateur** nœud et le configurer comme indiqué dans la figure 11.18. Ce nœud ex- traite uniquement les phrases \$GPGLL



Figure 11.18 Configuration du nœud de commutation

- Créez un **fonction** nœud et nommez-le **Lat/Lon**. Entrez les instructions suivantes dans ce nœud. Ce nœud extrait les informations de latitude et de longitude ainsi que leurs directions :

```
var nmea=msg.payload.split(",");
var lat=nmea[1];
var latdir = nmea[2];
var lon = nmea[3];
var londir = nmea[4];

T = "Lat="+lat+latdir+" Lon="+lon+londir;
msg.payload=T;
retourner le message;
```

- Créer un **déboguer** nœud pour afficher le résultat dans la fenêtre Debug. Cliquez sur Deploy. Vous devriez voir la latitude, la longitude et les directions affichées comme indiqué dans la figure- ure 11.19.

```

18/12/2019, 18:28:21 node: 7885a9ad.70a3b8
msg.payload : string[32]
"Lat=5127.37053N Lon=00003.13165E"

18/12/2019, 18:28:22 node: 7885a9ad.70a3b8
msg.payload : string[32]
"Lat=5127.37051N Lon=00003.13163E"

18/12/2019, 18:28:23 node: 7885a9ad.70a3b8
msg.payload : string[32]
"Lat=5127.37047N Lon=00003.13160E"

```

Figure 11.19 Affichage des informations de latitude et de longitude

Programme modifié

Le programme de flux donné à la figure 11.17 indique la latitude et la longitude en degrés et minutes. Par exemple, Lat = 5127.3610N signifie que la latitude est de 51 degrés et 27,3610 minutes et qu'elle se trouve dans l'hémisphère nord. De même, Lon = 00003.12748E signifie que la longitude est de 0 degrés 03,12748 minutes à l'est de Greenwich (où la longitude est de zéro degrés). Dans la plupart des applications, la latitude et la longitude sont représentées en degrés décimaux et si le point se trouve dans l'hémisphère sud (c.-à-d. que la direction est S), un signe négatif est ajouté à la valeur. De même, si le point se trouve du côté ouest de Greenwich (c.-à-d. que la direction est W), un signe négatif est ajouté à la valeur.

Dans cette section, nous modifierons les valeurs affichées de sorte qu'elles soient affichées en degrés avec les signes corrects.

Le processus est le suivant. Considérons l'exemple où le Lat = 5127.3610S :

- Prendre la partie entière de la valeur et diviser par 100 pour obtenir 51
- Prendre 51x100 de la valeur originale pour trouver la valeur des minutes, 5127.3610 – 51x100 = 27,361
- Diviser la valeur des minutes par 60 pour convertir en degrés, 27.361/60 = 0.45601
- Ajouter à la valeur des degrés précédemment trouvée, 51 + 0,45601 = 51,45601 degrés
- Puisque la direction est W, insérer un signe négatif, -51.45601 degrés

Programme de flux Node-RED : Le programme de flux est le même que dans la figure 11.17, où seulement le contenu du **fonction** Les changements de nœud. Le programme d'écoulement est répété dans la figure 11.20 ci-dessous afin que le programme d'écoulement exporté puisse être chargé à partir de la figure 11.20.



Figure 11.20 Programme de flux du projet

Saisissez les déclarations suivantes dans la **fonction** noeud :

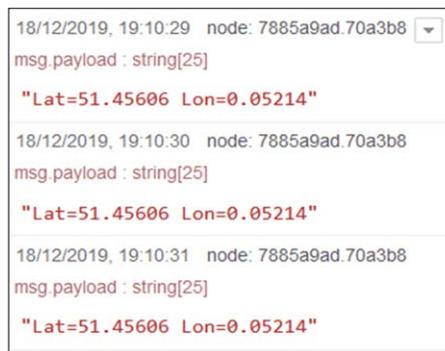
```

var nmea=msg.payload.split(",");
var lat=nmea[1];
var latdir = nmea[2];
var lon = nmea[3];
var londir = nmea[4];
//
// Latitude du processus
//
var Lat = nombre (lat);
var Latint = parseInt((parseInt(Lat)) / 100); var mins = Lat -
100*Latint;
divers degrés = minutes / 60
var LatValue = (Latint + degrees). toFixed(5, 2); if(latdir == "S")

    LatValue = -LatValue;
//
// Longitude du processus
//
var Lon = Nombre (lon);
var Lonint = parseInt((parseInt(Lon)) / 100); mins = Lon -
100*Lonint;
degrés = minutes / 60
var LonValue = (Lonint + degrees). toFixed(5, 3); if(londir ==
"W")
    LonValue = -LonValue;

msg.payload="Lat="+LatValue+" Lon="+LonValue; return
msg;
    
```

La figure 11.21 montre les latitudes et les longitudes en degrés.



```

18/12/2019, 19:10:29 node: 7885a9ad.70a3b8
msg.payload : string[25]
"Lat=51.45606 Lon=0.05214"

18/12/2019, 19:10:30 node: 7885a9ad.70a3b8
msg.payload : string[25]
"Lat=51.45606 Lon=0.05214"

18/12/2019, 19:10:31 node: 7885a9ad.70a3b8
msg.payload : string[25]
"Lat=51.45606 Lon=0.05214"
    
```

Figure 11.21 Affichage de la latitude et de la longitude en format degré

11 . 5 Project 56 – Affichage de notre emplacement sur une carte

Description : Dans ce projet, nous recevons et décodons les coordonnées GPS comme dans le projet 55 avec la latitude et la longitude converties en degrés. Nous traçons ensuite notre emplacement sur une carte en utilisant le nœud Dashboard appelé **carte du monde**.

Viser : Le but de ce projet est de montrer comment notre emplacement peut être tracé sur une carte en utilisant un nœud appelé **carte du monde**.

Schéma de circuit : Le schéma de circuit du projet est identique à la figure 11.14, où une carte GPC Click reçoit nos données de position et les transmet au Raspberry Pi.

Programme de flux Node-RED : Dans ce projet, nous utilisons un nœud de tableau de bord appelé **carte du monde**. Vous devez installer ce nœud dans votre nœud-RED avant de pouvoir l'utiliser. Les étapes sont :

- Cliquez **Menu -> Gérer la palette**, et cliquez sur **Installer**
- Entrer **node-red-contrib-web-worldmap** et cliquez **installer**
- Lorsque l'installation est terminée, vous devriez voir la carte du monde de nœud dans votre palette de nœuds.

Le programme de flux de ce projet est illustré à la figure 11.22. Il se compose de 4 nœuds : **en série** dans nœud pour recevoir les données GPS en temps réel, un commutateur **nœud pour extraire les phrases \$GPGLL**, un fonction **nœud pour extraire la latitude et la longitude en format degrés**, et un carte du monde **nœud pour afficher notre position sur une carte**.

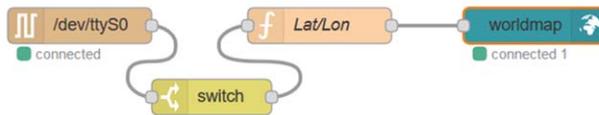


Figure 11.22 Programme de flux du projet

Les étapes pour élaborer le programme de flux sont :

- Créer un **en série dans** nœud comme précédemment
- Créer un **commutateur** nœud comme dans la figure 11.18
- Créer un **fonction** nœud, nommez-le comme **Lat/Lon** et entrez teh suivant les instructions à l'intérieur de ce nœud. Notez comment la latitude et la longitude doivent être spécifiées par variable **msg . charge utile** :

```
var nmea=msg.payload.split(","); var
lat=nmea[1];
var latdir = nmea[2];
```

```

var lon = nmea[3];
var londir = nmea[4];
//
// Latitude du processus
//
var Lat = nombre (lat);
var Latint = parseInt((parseInt(Lat) / 100); var mins = Lat -
100*Latint;
divers degrés = minutes / 60
var LatValue = (Latint + degrees). toFixed(5, 2); if(latdir == "S")

    LatValue = -LatValue;
//
// Longitude du processus
//
var Lon = Nombre (lon);
var Lonint = parseInt((parseInt(Lon)) / 100); mins = Lon -
100*Lonint;
degrés = minutes / 60
var LonValue = (Lonint + degrees). toFixed(5, 3); if(londir ==
"W")
    LonValue = -LonValue;

msg.payload = {"name":"Home", "lat":LatValue, "lon":LonValue}; return msg;

```

- Créer un **carte du monde** Node et le configurer comme indiqué dans la figure 11.23. Effectuer les réglages suivants (en supposant que vous êtes au Royaume-Uni) :

Carte de base :	UK OS Opendata
Zoom :	16 (peut être modifié) False
Carte de verrouillage :	Activer
Pan automatique :	Degrés
Coordonnées :	Faux
Verrouiller le zoom :	Désactiver
Clic droit :	Visible
Grilles :	

En définissant **Auto-pan** pour activer, nous activons le panoramique de la carte. En **Verrouiller le zoom** False nous pouvons changer le niveau de zoom de la carte en cliquant sur les signes + et – en haut à gauche de la carte. Vous devriez essayer de changer ces paramètres et voir leurs effets.

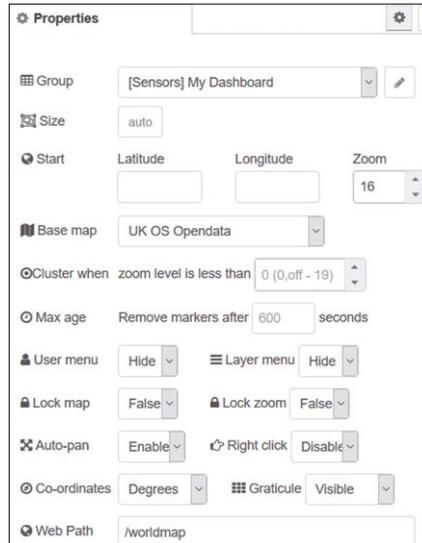


Figure 11.23 Configuration du noeud worldmap

- Rejoindre tous les nœuds comme dans la figure 11.22 et cliquer **Déployer**.
- Démarrer le tableau de bord en saisissant votre propre adresse IP :

<http://192.168.1.202:1880/ui/>

- Vous devriez voir votre position marquée sur la carte comme indiqué à la figure 11.24. Notez que la carte est ici agrandie pour montrer le niveau de la rue.

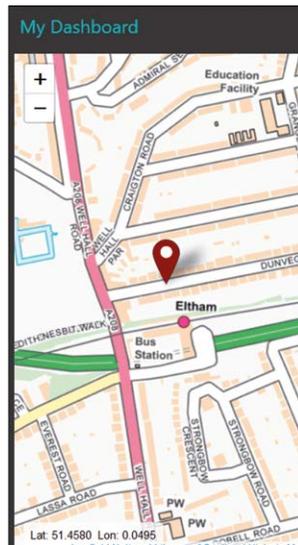


Figure 11.24 Votre position affichée sur la carte

Vous pouvez facilement modifier le niveau de zoom. La figure 11.25 montre la carte après avoir réduit le zoom.



Figure 11.25 Zoom sur la carte

Vous pouvez également modifier la carte de base en double-cliquant sur le **carte du monde** nœud. La figure 11.26 montre notre position avec la carte de base réglée sur le terrain.

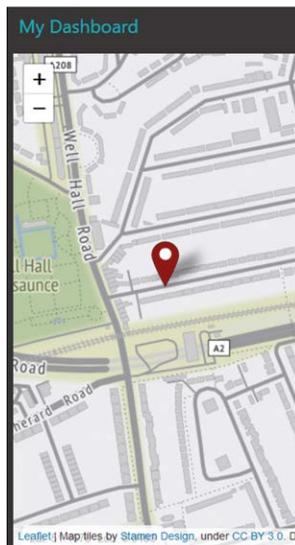


Figure 11.26 Carte de base réglée sur Terrain

Noeud **carte du monde** comprend de nombreuses fonctionnalités qui ne sont pas couvertes dans ce projet car ce sujet est au-delà du cadre de ce livre. Les lecteurs intéressés peuvent consulter le site web suivant pour obtenir des informations très détaillées :

<https://flows.nodered.org/node/node-red-contrib-web-worldmap>

11 . 6 Project 57 – Envoi de données série à arduino

Description : Dans ce projet, nous recevrons le bulletin météo en utilisant le nœud **openweathermap** et puis envoyer les lectures de température et d'humidité à l'Arduino Uno toutes les 5 secondes sur une ligne série. L'Arduino affichera la température dans la rangée supérieure et l'humidité dans la rangée inférieure d'un écran LCD.

Viser : Le but de ce projet est de montrer comment les données en série peuvent être produites à partir de Raspberry Pi en utilisant la **série** nœud.

Schéma fonctionnel : Le schéma du projet est présenté à la figure 11.27.

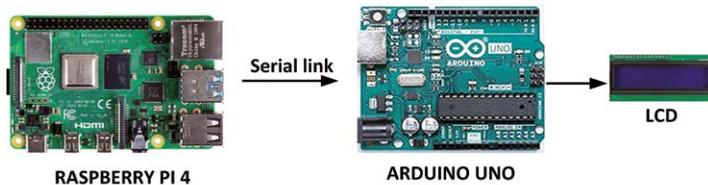


Figure 11.27 Schéma du projet

Schéma de circuit : La figure 11.28 montre le schéma du projet. La broche de sortie série du Raspberry Pi est connectée à la broche 3 de l'Arduino Uno, où cette broche est configurée comme entrée série. Un écran LCD parallèle compatible avec le HD44780 standard est connecté à l'Arduino Uno comme indiqué dans la figure. La sortie HAUTE tension d'une broche Raspberry Pi est maximum +3.3V et cela ne suffit pas pour piloter l'entrée de l'Arduino Uno. Un module de convertisseur de niveau logique (figure 11.29) est utilisé pour augmenter le niveau de tension de sortie du Raspberry Pi à +5V. Comme le montre la figure 11.30, celui utilisé dans ce projet est un module à 2 canaux. Un transistor MOSFET est utilisé pour convertir +3.3V en +5V, et un circuit de division de potentiel résistif est utilisé pour convertir de +5V en +3.3V.

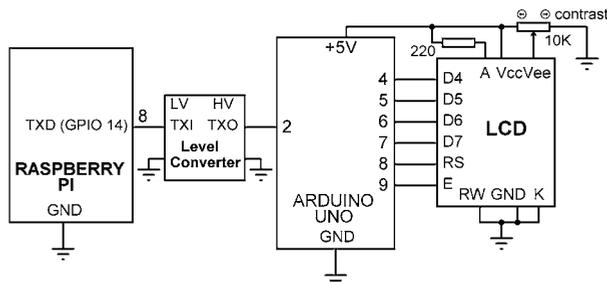


Figure 11.28 Schéma du projet

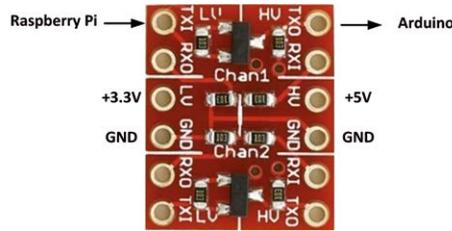


Figure 11.29 Module de convertisseur de niveau logique

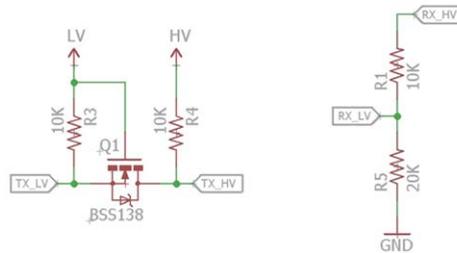


Figure 11.30 Schéma du convertisseur de niveau logique

Programme Arduino Uno : La figure 11.31 montre le programme Arduino Uno (programme : arduinolcd). Au début du programme, la bibliothèque de logiciels et les bibliothèques LCD sont incluses dans le programme. L'interface entre l'écran LCD et l'Arduino Uno est définie. Ici, les broches doivent être définies comme suit : RS, E, D4, D5, D6, D7. Les broches de la bibliothèque logicielle sont définies de telle sorte que la broche 2 est entrée et la broche 3 est sortie (non utilisée dans ce projet). Dans- Dans la routine de configuration, le débit en baud de la ligne série est réglé sur 9600, l'écran LCD est initialisé à 16 caractères par 2 lignes. Le texte Arduino est alors affiché sur l'écran LCD.

Le reste du programme s'exécute dans la boucle principale. Ici, le programme vérifie s'il y a des données sur le port série et si oui, la fonction readStringUntil() est utilisée pour lire une chaîne de données jusqu'à ce que le caractère de fin ''#'' soit détecté (ce caractère est envoyé par Raspberry Pi pour indiquer la fin de ses données) et les données reçues sont stockées dans la chaîne varia- Le formulaire suivant : nn.n,mm#. Ensuite, les fonctions indexOf() et substring() sont utilisées pour extraire la température et l'humidité de la chaîne s1. La température est affichée dans la rangée supérieure comme T=nn.nC et l'humidité est affichée dans la rangée inférieure comme H=mm%.

```

/*-----
 * Ce programme reçoit les données de température et d'humidité
 * du Raspberry Pi sur la ligne de série à 9600 Baud.
 * La température et l'humidité sont affichées sur un écran LCD
 *
 * Fichier : Arduinolcd *-----*/

```

```

#include <SoftwareSerial.h> #include // Bibliothèque série //
<LiquidCrystal.h> Lcd LiquidCrystal // Bibliothèque LCD
(8,9,4,5,6,7); SoftwareSerial MySerial (2, // Connexions LCD //
3); // RX, TX

```

```

void setup()
{
  MySerial.begin(9600);           // Démarrer série
  lcd.begin(16,2);               // Init LCD
  lcd.print("Arduino");         // Affichage Arduino //
  delay(1000);                  Attendre 1 seconde
}

boucle vide()
{
  si(MySerial.available() > 0) { // Données disponibles?

    lcd.clear();                // Clear LCD
    Chaîne s1 = MySerial.readStringUntil('#'); int virgule =
    s1.indexOf(',');             // Position de la virgule
    Température de chaîne = s1.substring(0, virgule); Humidité
    de chaîne = s1.substring(comma+1); lcd.setCursor(0,0);
    lcd.print("T=");            // Curseur à 0,0
    lcd.print (Température);    // Affichage T=
    lcd.print("C");             // Afficher la température
    lcd.setCursor(0,1);        // Afficher C
    lcd.print("H=");           // Curseur à 0,1
    lcd.print (humidité);      // Affichage H=
    lcd.print("%");            // Affichage de l'humidité //
                                Affichage %
  }
}

```

Figure 11.31 Programme Arduino Uno

Programme de flux Node-RED : La figure 11.32 montre le programme de flux de ce projet. Dans ce projet, il y a 4 nœuds : **injecter** nœud pour démarrer le flux, un **openweathermap** node pour obtenir le bulletin météo local, un **fonction** node pour extraire la température et l'humidité de ce rapport, et une **série** nœud pour envoyer des données à l'Arduino Uno sur la ligne série.

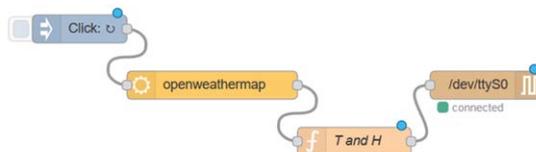


Figure 11.32 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injection** noeud à répéter toutes les 5 secondes
- Créer un noeud openweathermap comme avant et entrer la clé API
- Créer un **fonction** noeud nommé **T et H**, et entrez les déclarations suivantes dans- côté de ce noeud. Ce noeud extrait la température et l'humidité, les sépare avec une virgule et insère le caractère terminateur '#', cliquez sur **Terminé** :

```
var T=msg.payload.tempc;
var H = msg.payload.humidity; All = T
+ ","+H + "#"; msg.payload = All;
retourner le message;
```

- Créer un **série** noeud et le configurer comme indiqué sur la figure 11.33, cliquez **Mise à jour** puis **Terminé**

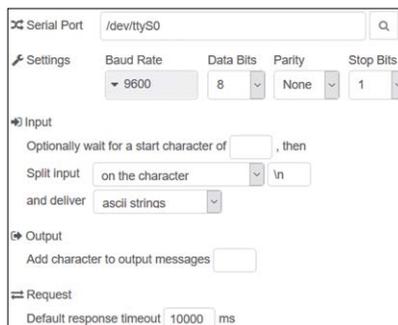


Figure 11.33 Configuration de la sortie série du noeud

- Rejoindre les noeuds et cliquer **Déployer**.
- Construisez votre circuit comme indiqué dans la figure 11.28, allumez l'Arduino Uno. Vous devriez voir la température et l'humidité affichées sur l'écran LCD comme indiqué dans la figure 11.34



Figure 11.34 Affichage de la température et de l'humidité

N'oubliez pas, après avoir fini de travailler avec votre port série, vous pouvez vouloir donner ce port à la console. Ceci est facilement fait en utilisant l'utilitaire raspi-config en sélectionnant option **5** (Options d'interface) et puis **P6** (Série) et sélectionnez **Oui**. Quittez raspi-config et redémarrez votre Raspberry Pi.

11 . 7 Projet 58 – Connexion de Raspberry Pi et d’Arduino Uno à l’aide de ports USB

Description : Dans le projet précédent, nous devions connecter le port de sortie série de notre Raspberry Pi à l’entrée série de l’Arduino Uno. Dans ce projet, nous connectons simplement le port USB de l’Arduino à un des ports USB du Raspberry Pi. Comme dans le projet précédent, nous obtenons et affichons la température locale et l’humidité sur l’Android Uno.

Viser : Le but de ce projet est de montrer comment la communication en série peut être établie à l’aide d’un des ports USB du Raspberry Pi.

Schéma fonctionnel : Le schéma du projet est illustré à la figure 11.35. L’Arduino est connecté au Raspberry Pi par un câble USB. Il n’y a pas d’autre connexion entre les deux ordinateurs.

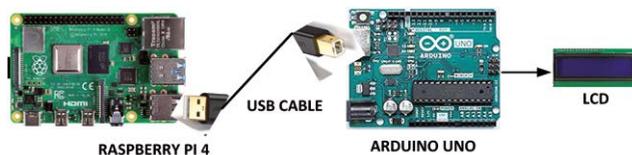


Figure 11.35 Schéma du projet

Raspberry Pi Nom du port USB : Avant d’utiliser les ports USB du Raspberry Pi, nous devons connaître le nom de leur périphérique. La façon la plus simple est peut-être d’afficher la liste des périphériques série avec- Sortir l’Arduino, puis connecter l’Arduino au Raspberry Pi et afficher à nouveau la liste des appareils. De cette façon, le nom du périphérique du Raspberry Pi peut être trouvé très facilement. La commande pour afficher les périphériques série est :

```
pi@framberrypi:~$ ls /dev/tty*
```

La figure 11.36 montre les périphériques série sur le Raspberry Pi de l’auteur avant que l’Arduino Uno ne soit connecté.

```
pi@raspberrypi:~$ ls /dev/tty*
/dev/tty /dev/tty0 /dev/tty1 /dev/tty10 /dev/tty11 /dev/tty12 /dev/tty13 /dev/tty14 /dev/tty15 /dev/tty16 /dev/tty17 /dev/tty18
/dev/tty19 /dev/tty2 /dev/tty20 /dev/tty21 /dev/tty22 /dev/tty23 /dev/tty24 /dev/tty25 /dev/tty26 /dev/tty27 /dev/tty28 /dev/tty29
/dev/tty3 /dev/tty30 /dev/tty31 /dev/tty32 /dev/tty33 /dev/tty34 /dev/tty35 /dev/tty36 /dev/tty37 /dev/tty38 /dev/tty39 /dev/tty4
/dev/tty40 /dev/tty41 /dev/tty42 /dev/tty43 /dev/tty44 /dev/tty45 /dev/tty46 /dev/tty47 /dev/tty48 /dev/tty49 /dev/tty5
/dev/tty51 /dev/tty52 /dev/tty53 /dev/tty54 /dev/tty55 /dev/tty56 /dev/tty57 /dev/tty58 /dev/tty59 /dev/tty60 /dev/tty61
/dev/tty62 /dev/tty63 /dev/tty7 /dev/tty8 /dev/tty9 /dev/ttyAMA0 /dev/ttyprintk
```

Illustration 11.36 Périphériques série Raspberry Pi

La liste des périphériques après avoir connecté l’Arduino Uno au Raspberry Pi avec un câble USB est illustrée à la figure 11.37.

```

pi@raspberrypi:~$ ls /dev/tty*
/dev/tty /dev/tty19 /dev/tty3 /dev/tty40 /dev/tty51 /dev/tty62
/dev/tty0 /dev/tty2 /dev/tty30 /dev/tty41 /dev/tty52 /dev/tty63
/dev/tty1 /dev/tty20 /dev/tty31 /dev/tty42 /dev/tty53 /dev/tty7
/dev/tty10 /dev/tty21 /dev/tty32 /dev/tty43 /dev/tty54 /dev/tty8
/dev/tty11 /dev/tty22 /dev/tty33 /dev/tty44 /dev/tty55 /dev/tty9
/dev/tty12 /dev/tty23 /dev/tty34 /dev/tty45 /dev/tty56 /dev/ttyACM0
/dev/tty13 /dev/tty24 /dev/tty35 /dev/tty46 /dev/tty57 /dev/ttyAMA0
/dev/tty14 /dev/tty25 /dev/tty36 /dev/tty47 /dev/tty58 /dev/ttyprintk
/dev/tty15 /dev/tty26 /dev/tty37 /dev/tty48 /dev/tty59 /dev/ttyS0
/dev/tty16 /dev/tty27 /dev/tty38 /dev/tty49 /dev/tty6
/dev/tty17 /dev/tty28 /dev/tty39 /dev/tty5
/dev/tty18 /dev/tty29 /dev/tty4 /dev/tty50 /dev/tty61
    
```

Figure 11.37 Périphériques série après la connexion d'Arduino Uno

En comparant les deux chiffres, il est évident que le nom du périphérique de port USB est : `/dev/ttyACM0` . Maintenant que nous connaissons le nom du périphérique, nous pouvons modifier notre flux pro Arduino et Node-RED- grammes en conséquence.

Programme Arduino Uno : Le port USB Arduino Uno est connecté aux broches de port série 0 (RX) et 1 (TX) du matériel et est nommé juste **Série**. Par conséquent, les seules modifications requises dans la figure 11.31 sont de supprimer les ports série du logiciel et de remplacer **MySerial** avec **Série**. Le programme modifié (programme : **arduinousb**) La liste est présentée à la figure 11.38.

```

/*-----
 * Ce programme reçoit les données de température et d'humidité
 * du Raspberry Pi sur une liaison USB à 9600 Baud.
 * La température et l'humidité sont affichées sur un écran LCD
 *
 * Dossier : Arduinousb *-----*/

#include <LiquidCrystal.h> Lcd // Bibliothèque LCD
LiquidCrystal (8,9,4,5,6,7); // Connexions LCD

void setup()
{
    Serial.begin(9600); // Démarrer série
    lcd.begin(16,2); // Init LCD
    lcd.print("&quot;Arduino&quot;"); // Affichage Arduino //
    delay(1000); // Attendre 1 seconde
}

boucle vide()
{
    si(Serial.available() &gt; 0) { // Données disponibles?

        lcd.clear(); // Clear LCD
        Chaîne s1 = Serial.readStringUntil(&apos;#&apos;);
        virgule = s1.indexOf(&apos;,&apos;); // Position de la virgule
        Température de chaîne = s1.substring(0, virgule); Humidité
        de chaîne = s1.substring(comma+1); lcd.setCursor(0,0);
        // Curseur à 0,0
    }
}
    
```

```

        lcd.print("T=");           // Affichage T=
        lcd.print(Temperature);    // Afficher la température
        lcd.print("C");           // Afficher C
        lcd.setCursor(0,1);       // Curseur à 0,1
        lcd.print("H=");         // Affichage H=
        lcd.print(Humidity);      // Affichage de l'humidité
        lcd.print("%");           // Affichage %
    }
}

```

Figure 11.38 Programme modifié d'Arduino Uno

Programme Node-RED Flow : La seule modification requise dans le programme de flux de la figure 11.32 est de changer le nom du port série à partir de `/dev/ttyS0` à `/dev/ttyACM0` dans le noeud **en série** tout le reste reste le même. C'est tout. Cliquez **Déployer** et vous devriez voir la colère-Température et humidité affichées sur l'écran LCD comme indiqué à la figure 11.34.

11 . 8 Project 59 – Envoi de données en série du Raspberry Pi à l'Arduino via RF Radio

Description : Dans les projets précédents, nous avons envoyé des données du Raspberry Pi en utilisant une connexion directe à l'Arduino, soit en connectant la broche de transmission du Raspberry Pi à la broche de réception d'Arduino, ou en connectant les deux appareils ensemble à l'aide d'un câble USB.

Dans ce projet, nous utilisons une paire de modules radio RF pour envoyer des données en série du Raspberry Pi à l'Arduino Uno. Comme dans le projet précédent, les données de température et d'humidité sont envoyées à l'Arduino Uno qui les affiche sur l'écran LCD.

Viser : Le but de ce projet est de montrer comment les modules radio RF peuvent être utilisés pour envoyer (ou recevoir) données de série d'un Raspberry Pi.

Renseignements généraux : Les modules de radiocommunication RF sont utilisés dans des projets où il n'existe aucun autre moyen d'établir la communication entre deux appareils. p. ex., il n'y a pas de connexion Wi-Fi, Internet ou Bluetooth. Dans ce projet, le module moyen tRF Click (voir la figure 11.39) est utilisé, un pour Raspberry Pi et un pour Arduino Uno. tRF Click est un module de communication RF à courte portée, fonctionnant sur la bande de fréquences sans licence 868MHz ISM. Ce module est basé sur le module LE70-868RF de Telit. Il offre une compatibilité complète avec les commandes AT. Un UART embarqué est fourni pour que l'hôte-puter peut envoyer et recevoir des messages via l'UART. Le module peut fonctionner avec +3.3V à +5V, et il fournit une puissance de transmission de 500mW, et -117dBm recevoir la sensibilité. La portée du tRF Click est estimée à environ 10 kilomètres. Une antenne externe doit être utilisée avec le module pour atteindre la portée maximale indiquée. Des détails complets sur le module tRF peuvent être obtenus à partir du lien : <https://www.mikroe.com/trf-click>.



Figure 11.39 tRF Module de clic

tRF est un module à broches 2x8 avec les noms de broche suivants (la broche 1 se trouve dans le coin supérieur gauche, NC = No Connection) :

1 : DGST	16 : Service de la sécurité
2 : RST	15 : ACQ
3 : NC	14 : TXD
4 : NC	13 : RXD
5 : NC	12 : NC
6 : NC	11 : NC
7 : +3,3V	10 : NC
8 : GND	9 : GND

Normalement, seules les broches suivantes sont utilisées dans la plupart des applications : +3.3V, GND, TXD, RXD

Schéma fonctionnel : Le schéma du projet est illustré à la figure 11.40. Deux modules tRF Click sont utilisés pour établir la communication radio RF entre les appareils.

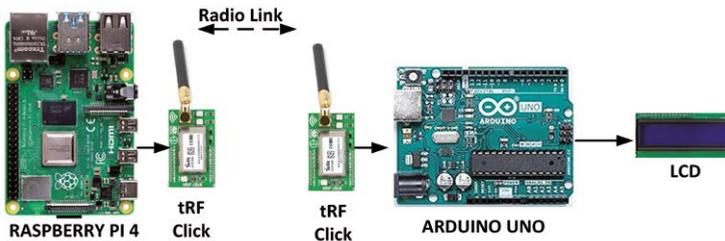


Figure 11.40 Schéma du projet

Schéma de circuit : Le schéma du projet est illustré à la figure 11.41. Sur le Rasp- côté Pi, la broche de sortie série TXD (GPIO 14) est connectée à l'entrée RXD du module tRF.

Du côté Arduino, la broche TXD du module tRF est connectée à l'entrée série 2 de l'Arduino par un convertisseur de niveau logique (voir la figure 11.29) puisque le niveau HIGH de sortie de tRF n'est que de +3,3V (vous pouvez constater qu'il n'y a pas besoin d'un convertisseur de niveau logique ici).

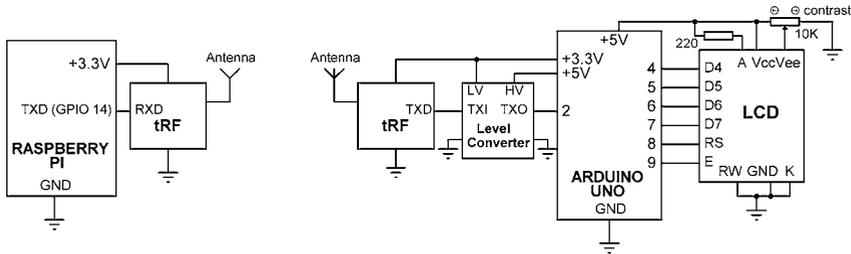


Figure 11.41 Schéma du projet

Programme Arduino Uno : Le programme Arduino Uno est exactement le même que celui donné dans la figure 11.31 (programme : **dur à cuire**), sauf que le débit en bauds doit être réglé à 19200 qui est le débit en bauds par défaut du module tRF.

Programme de flux Node-RED : Le programme de flux Node-RED pour Raspberry Pi est exactement le même que celui donné dans la figure 11.32, où /dev/ttyS0 est utilisé comme broche de port série du Raspberry Pi et le débit en bauds doit être réglé à 19200.

Construire les circuits Arduino et Raspberry Pi comme indiqué dans la figure 11.41. La température et l'humidité sont affichées sur l'écran LCD, comme indiqué à la figure 11.34.

11 . 9 Résumé

Dans ce chapitre, nous avons appris à utiliser Node-RED **en série dans et série** Les nœuds avec plusieurs projets.

Dans le prochain chapitre, nous allons apprendre à utiliser Raspberry Pi Sense HAT dans des projets avec Node-RED.

Chapitre 12 • Utilisation de la HAT sensée

12 . 1 Aperçu

Sense HAT est une carte complémentaire pour Raspberry Pi contenant un certain nombre de capteurs utiles et un réseau de LED. HAT est un acronyme pour **H**matériel de musique **A**fixé sur **T**op. Sense HAT was an im- Le programme Astro Pi, qui était un Raspberry Pi éducatif envoyé à la Station spatiale internationale avec l'astronaute britannique Tim Peake pour exécuter du code développé par des enfants. Le véritable Astro Pi a subi quelques modifications et avait un boîtier métallique pour le rendre approprié à une utilisation dans l'espace.

Sense HAT comprend des capteurs pour mesurer la température, l'humidité, la pression, l'accéléromètre, le gyroscope et un magnétomètre. De plus, une matrice LED 8 x 8 programmable indépendamment est incluse sur le tableau qui peut être programmée pour afficher du texte et des petites images.

Dans ce chapitre, nous allons concevoir divers projets en utilisant le nœud Sense HAT de Node-RED avec la carte Sense HAT. Avant cela, il est intéressant de se pencher sur les caractéristiques du tableau Sense HAT. Des informations détaillées sur Sense HAT peuvent être obtenues à partir du lien : [magpi. cc/AstroPiGuide](http://magpi.cc/AstroPiGuide). Aussi du **Essentials_SenseHAT_v1** (MagPi Essentials series), et de nombreuses autres sources Internet.

12 . 2 Le tableau HAT du Sense

La figure 12.1 montre la carte Sense HAT. On peut identifier les composants suivants sur la carte :

- Réseau de DEL 8 x 8, avec une résolution de 15 bits
- Accéléromètre, +/- 2/4/8/16 G
- Gyroscope, +/- 245/500/2000 DPS
- Magnétomètre, +/- 4/8/12/16 Gauss
- Baromètre, 260 – 1260 hPa pression absolue
- Capteur de température, précision +/-2oC, plage 0 - 65oC
- Capteur d'humidité, précision +/- 4,5 %, plage de 20 à 80 %
- Puce du contrôleur graphique
- Manette à cinq boutons avec mouvements gauche, droit, haut, bas et entrée

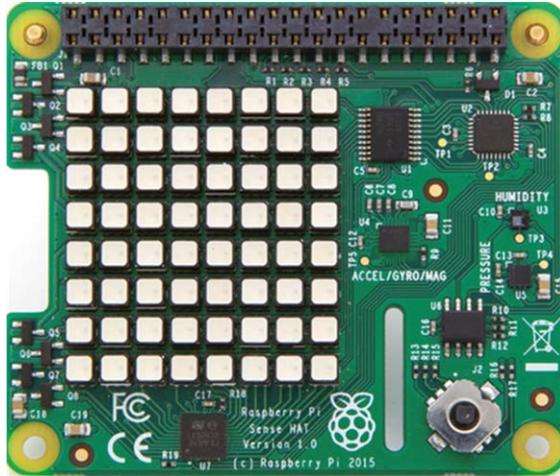


Figure 12.1 Carte HAT de détection

La matrice LED est très utile car elle vous permet d'afficher du texte et aussi des données de divers capteurs, pour jouer à des jeux, etc. Nous allons dans les sections ultérieures de ce chapitre comment utiliser l'affichage de divers éléments sur la matrice LED.

Un simulateur Node-RED est disponible et peut être téléchargé gratuitement sur Internet. Le simulateur vous permet de créer des flux et d'interagir avec un HAT virtuel, sans avoir le matériel réel. Le simulateur est utile pour essayer un projet sur le simulateur avant d'acheter le matériel.

12.3 nœuds RED Sense HAT

Deux nœuds HAT Sense sont disponibles en Node-RED sous la palette Raspberry Pi. Comme le montre la figure 12.2, il s'agit du nœud d'entrée Sense HAT et du nœud de sortie Sense HAT.



Figure 12.2 Nœuds HAT de détection

Nœud d'entrée de Sense Hat

Le nœud d'entrée du Sense HAT lit les valeurs de divers capteurs sur la carte Sense HAT et envoie ces lectures. Les capteurs sont regroupés en trois : **motion** événements, **environnement** événements, et joystick **événements**.

Les événements de la motion sont les lectures de l'accéléromètre, du gyroscope, du magnétomètre et du cap de la boussole, qui sont envoyées environ 10 par seconde. La charge utile est les valeurs des capteurs, et le sujet est le mouvement. Les valeurs des capteurs sont retournées avec les unités suivantes :

- Accéléromètre (x,y,x) en Gs
- Gyroscope (x.y.z) en radians/s
- Orientation (roulis, tangage, lacet) en degrés
- Boussole se dirigeant vers le nord en degrés

Les événements environnementaux Ce sont les valeurs des capteurs suivants : température, humidité et pression. La charge utile est la valeur renvoyée par un capteur et le sujet est l'environnement. Les valeurs sont retournées avec les unités suivantes :

- La température en degrés centigrades
- Humidité en pourcentage d'humidité relative
- Pression en millibars

Les événements de la manette sont envoyés lorsque la manette est déplacée. La charge utile est la valeur retournée et le sujet est la manette. Les valeurs suivantes sont retournées :

- La touche peut être : HAUT, BAS, GAUCHE, DROITE, ENTRÉE
- État de la touche peut être : 0 si la touche a été relâchée, 1 si la touche a été pressée et 2 si la touche est maintenue enfoncée

Par exemple, pour obtenir la température, l'humidité et la pression, nous pouvons créer une fonction avec 3 sorties et entrer la déclaration suivante dans la fonction :

```
var temperature = {payload : msg.payload.temperature}; var humidity =
{payload : msg.payload.humidity};
var pressure = {payload : msg.payload.pressure}; return [temperature,
humidity, pressure];
```

Node de sortie HAT Sense

Ce nœud envoie des commandes à la matrice LED 8x8 sur le matériel Sense HAT. msg.payload est chargé avec les commandes à envoyer. Nous verrons dans les sections suivantes comment utiliser le nœud de sortie dans les projets.

Des exemples de projets sont donnés dans les sections suivantes pour montrer comment les nœuds Sense HAT peuvent être utilisés dans les projets Raspberry Pi. Assurez-vous que la carte Sense HAT est branchée sur votre carte Raspberry Pi avant d'utiliser les projets.

12 . 4 Project 60 - Affichage de la température, de l'humidité, et pression (événements environnementaux)

Description : Dans ce projet, nous allons obtenir la température, l'humidité et la pression de la carte HAT Sense et les afficher sous forme d'indicateurs et de graphiques.

Viser : Le but de ce projet est de montrer comment les événements environnementaux peuvent être lus et affichés.

Programme de flux Node-RED : La figure 12.3 montre le programme de flux qui se compose de seulement 8 nœuds : un **Sense HAT** nœud pour obtenir les variables environnementales, une **fonction** nœud pour extraire la température, l'humidité et la pression, et 3 **jauge** des nœuds pour afficher la température, l'humidité et la pression, et 3 **graphique** nœuds pour afficher les mêmes variables.

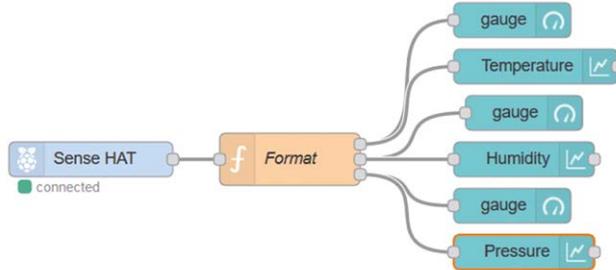


Figure 12.3 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **Sense HAT** nœud et ensemble **Sorties** aux événements environnementaux
- Créer un nœud de fonction, le nommer en tant que format et entrer les instructions suivantes dans ce nœud :

```
var temperature = {payload : msg.payload.temperature}; var humidity =
{payload : msg.payload.humidity};
var pressure = {payload : msg.payload.pressure}; return [temperature,
humidity, pressure];
```

- Créer 3 nœuds de jauge et 3 nœuds de graphique avec les configurations suivantes :

Gaine de nœud	Groupe	Étiquette	Unités de portée 0
Température	[Accueil]Température ambiante	jauge	à 35 C
Humidité	[Accueil]Humidité ambiante	jauge	0 à 100 % 900 à
Pression	[Accueil]Pression ambiante	jauge	1100 hPa

Nœud graphique	Groupe	Étiquette min-max
Température	[Accueil]Température ambiante	Température 0 à 35
Humidité	[Accueil]Humidité ambiante	Humidité 0 à 100
Pression	[Accueil]Pression ambiante	Pression 900 à 1100

Joindre tous les nœuds et cliquer sur Déployer. Vous devriez voir la température, l'humidité et la pression affichées dans le tableau de bord. Notez que par défaut les jauges et les graphiques sont affichés verticalement. Nous pouvons, cependant, regrouper les jauges et les graphiques de sorte qu'ils soient affichés horizontalement, avec les jauges et leurs diagrammes correspondants affichés verticalement ensemble comme indiqué à la figure 12.4. Le regroupement des jauges et des diagrammes est indiqué à la figure 12.5.

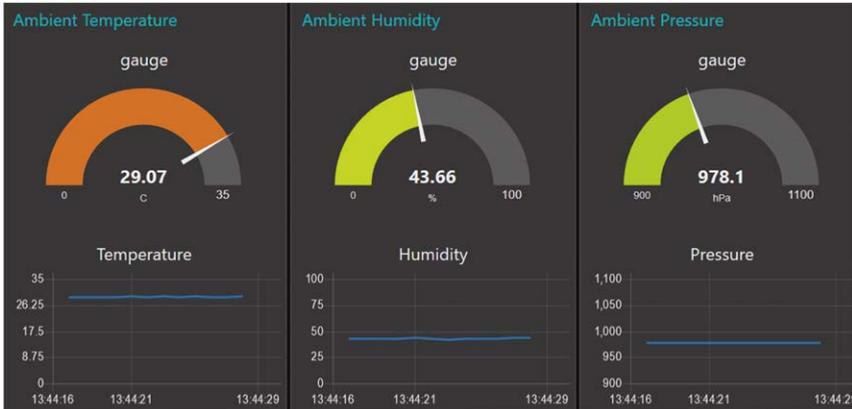


Figure 12.4 Affichage dans le tableau de bord

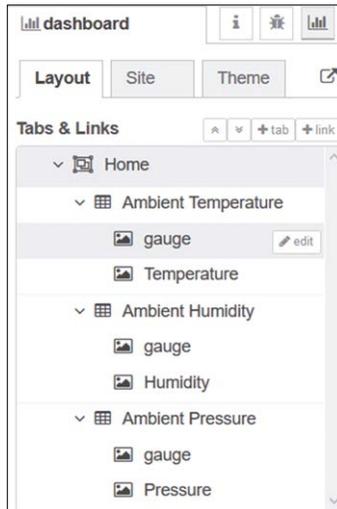


Figure 12.5 Regroupement des jauges et des diagrammes

Notez que la température n'est pas affichée correctement. Ceci est dû au fait que le capteur de température sur la carte Sense HAT est très proche du processeur Raspberry Pi et par conséquent, la lecture n'est pas la bonne. Une lecture correcte peut être obtenue si la carte Sense HAT est connectée au Raspberry Pi à l'aide d'un câble ruban de sorte que le capteur ne soit pas proche du processeur.

12 5Projet 61 - Affichage du cap de la boussole (événements de mouvement)

Description : Dans ce projet, nous allons afficher le cap de la boussole en continu.

Viser : Le but de ce projet est de montrer comment la direction du compas peut être affichée.

Programme de flux Node-RED : La figure 12.6 montre le programme de flux qui se compose de seulement 3 nœuds : a **Sense HAT** nœud pour obtenir les variables de mouvement, un **fonction** nœud pour extraire le cap de la boussole, et un tableau de bord **texte** nœud pour afficher l'en-tête de manière dynamique.

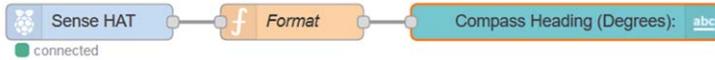


Figure 12.6 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **Sense HAT** nœud et ensemble **Sorties** aux événements de motion
- Créer un **fonction** nœud, nommez-le comme **Format** et entrer les instructions suivantes dans ce nœud :

```
var CompassHead = {payload : msg.payload.compass};
return [CompassHead];
```

- Créer un tableau de bord **texte** nœud, nommez-le comme **Cap de la boussole (degrés)** :, créer un nouveau groupe appelé **[Accueil]Sense HAT**
- Rejoindre les groupes, cliquez **Déployer**, et démarrez votre tableau de bord. Vous devriez voir le cap du compas du nord affiché dynamiquement comme indiqué dans la figure 12.7

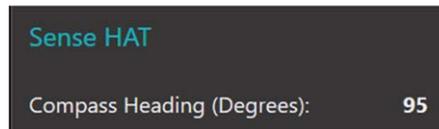


Figure 12.7 Affichage du cap de la boussole

12 . 6 Projet 62 - Affichage de l'accélération (événements de mouvement)

Description : L'accélération (quantité de force G) peut être obtenue en trois dimensions de x, y et z. Dans ce projet, nous allons afficher l'accélération en continu dans les trois di- mesures.

Viser : Le but de ce projet est de montrer comment l'accélération peut être affichée.

Programme de flux Node-RED : La figure 12.8 montre le programme de flux qui se compose de 5 nœuds : a **Sense HAT** nœud pour obtenir les variables de mouvement, un **fonction** nœud avec 3 sorties pour extraire l'accélération en 3 dimensions, et 3 tableau de bord **texte** nœuds pour afficher l'accélération dans chaque direction.

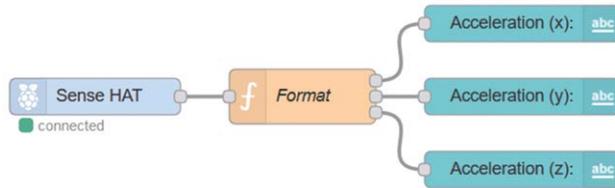


Figure 12.8 Programme de flux du projet

- Créer un **Sense HAT** noeud et ensemble **Sorties** aux événements de motion
- Créer un **fonction** noeud avec 3 sorties, nommez-le comme **Format** et entrez dans le fol- Les instructions à l'intérieur de ce nœud. Cette fonction renvoie l'accélération en 3 dimensions :

```
var accx = {payload : msg.payload.acceleration. x}; var
accy = {payload : msg.payload.acceleration. y}; var accz =
{payload : msg.payload.acceleration. z}; retour [accx, accy,
accz];
```

- Créer 3 tableaux de bord **texte** les nomment comme **Accélération (x) :**, **Accélération(s) :** et **Accélération (z) :**, utiliser le **groupe précédemment créé [Accueil]Sens CHAPEAU**
- Rejoindre les nœuds et cliquer **Déployer**. Vous devriez voir l'accélération affichée dans votre tableau de bord comme indiqué à la figure 12.9

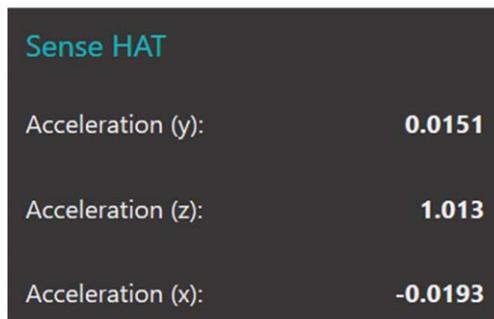


Figure 12.9 Affichage de l'accélération en 3 dimensions

12 7Projet 63 - Affichage de l'orientation (événements de mouvement)

Description : Dans ce projet, nous allons afficher le pitch, le roll et le yaw de manière dynamique.

Viser : Le but de ce projet est de montrer comment l'orientation peut être affichée dynamiquement.

Renseignements généraux : Il est important de comprendre la différence entre le pitch, le roll et le yaw. Peut-être que la façon la plus simple de comprendre ces termes est d'observer le mouvement- La figure 12.10 représente un avion avec les termes tangage, roulis et

Le mouvement de lacet est expliqué graphiquement.

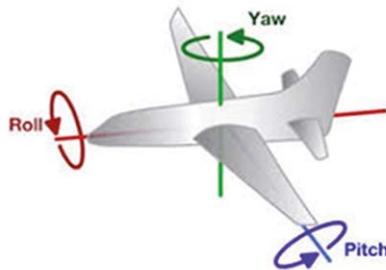


Figure 12.10 Tangage, roulis et lacet d'un avion

Dans la figure 12.11, les termes tangage, roulis et lacet sont appliqués à la fonction HAT du sens.

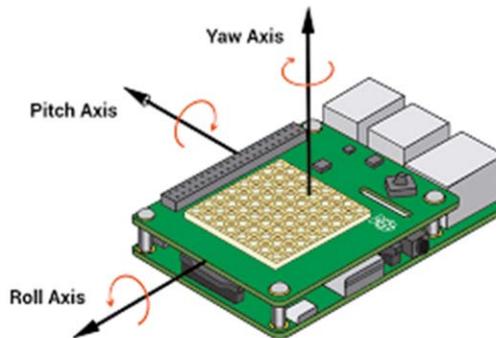


Figure 12.11 Tangage, roulis et lacet du HAT de sens

Programme de flux Node-RED : Le programme de flux est exactement le même que la figure 12.8, mais le contenu de **fonction** noe **texte** les nœuds sont modifiés. La figure 12.12 montre le programme de flux.

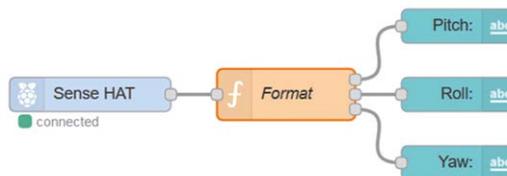


Figure 12.12 Programme de flux du projet

- Saisissez les déclarations suivantes dans la **fonction** noeud :

```
var accx = {payload : msg.payload.orientation.pitch}; var accy =  
{payload : msg.payload.orientation.roll}; var accz = {payload :  
msg.payload.orientation.yaw}; return [accx, accy, accz];
```

- Changer le **Étiquettes** de la **texte** Les nœuds comme indiqué dans la figure 12.12. La figure 12.13 montre le tangage, le roulis et le lacet affichés dans le tableau de bord.



Figure 12.13 Affichage du tangage, du roulement et de la lacet

12 . 8 Utilisation du joystick

La manette se trouve dans le groupe Joystick Events de Sense HAT. Comme décrit plus haut à la section 12.3, les événements de la manette sont envoyés lorsque la manette est déplacée. La charge utile est la valeur re- Le sujet est la manette. Les valeurs retournées sont UP, DOWN, RIGHT, LEFT, EN- TER. Les états des clés sont :

0 : la touche a été relâchée 1 :
la touche a été pressée 2 : la
touche est maintenue enfoncée

La façon la plus simple de comprendre le fonctionnement du joystick est peut-être de régler **Sorties**du nœud Sense HAT aux événements de Joystick et connecter un déboguer **La fenêtre de débogage affiche l'action effectuée en appuyant sur le bouton ou en le déplaçant (Figure 12.14).**

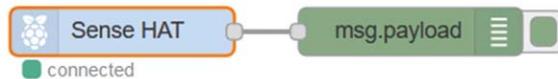


Figure 12.14 Programme de flux pour tester la manette

Par exemple, le fait de déplacer la manette vers la gauche et de maintenir la touche enfoncée génère le message indiqué à la figure 12.15.

```

    ▶ { key: "LEFT", state: 2 }
    20/12/2019, 15:08:50 node: f6527996.096db8
    joystick : msg.payload : Object
    ▶ { key: "LEFT", state: 2 }
    20/12/2019, 15:08:50 node: f6527996.096db8
    joystick : msg.payload : Object
    ▶ { key: "LEFT", state: 2 }
    20/12/2019, 15:08:50 node: f6527996.096db8
    joystick : msg.payload : Object
    ▶ { key: "LEFT", state: 2 }
  
```

Figure 12.15 Déplacement de la manette vers la gauche et maintien

12 . 9 Utilisation de la matrice LED

Le **Sense HAT output** nœud est utilisé pour contrôler la matrice DEL sur le matériel Sense HAT. Les commandes pour contrôler la matrice LED sont envoyées à msg.payload (plusieurs commandes peuvent être envoyées dans un seul message en les séparant avec un caractère de nouvelle ligne).

La plupart des instructions de contrôle de matrice LED sont basées sur les coordonnées LED (x, y) de la carte Sense HAT. La figure 12.16 montre les coordonnées des DEL par rapport à la carte.

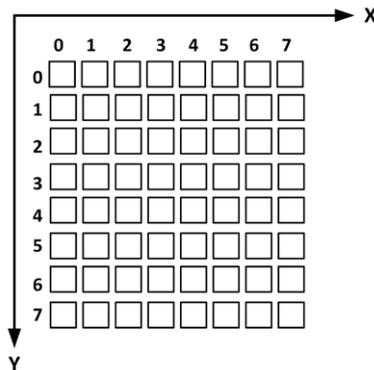


Figure 12.16 Coordonnées des DEL

La façon la plus simple de comprendre le fonctionnement de la matrice LED est peut-être de créer un programme de test matriciel LED en connectant une **injecter** nœud et un **fonction** nœud à **Sense HAT output** nœud comme indiqué dans la figure 12.17, et puis examiner la matrice de LED en écrivant des instructions à l'intérieur **fonction** nœud.



Figure 12.17 Programme de test de flux de matrice LED

Couleurs LED

La couleur d'une DEL peut être réglée à l'aide de l'instruction :

```
"x,y,colour"
```

Où x et y doivent être de 0 à 7 ou un * pour indiquer la ligne ou la colonne entière. La couleur peut être l'une des suivantes :

```
Nom de couleur HEX (p. ex.,
#aa991e) RVB triple (p. ex. 255, 180,
0) Nom de couleur HTML
off
```

Exemple 1

Régler la LED à 0,0 (c.-à-d. dans le coin supérieur gauche lorsque les prises USB de Raspber- Les PI sont orientés vers la droite) et allumez le voyant comme indiqué sur la figure 12.18.

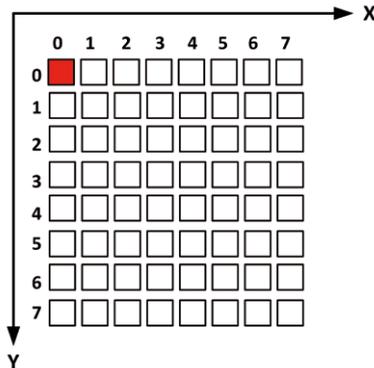


Figure 12.18 Motif LED pour l'exemple 1

Solution 1

Entrez l'énoncé suivant dans la fonction :

```
msg.payload = "0,0,red";
renvoie msg;
```

Exemple 2

Régler toutes les LED sur une couleur bleue.

Solution 2

Entrez l'énoncé suivant dans la fonction :

```
msg.payload = "*,*,blue";
renvoie msg;
```

Exemple 3

Régler les voyants LED aux quatre coins sur bleu, rouge, vert et jaune comme indiqué dans la figure 12.19

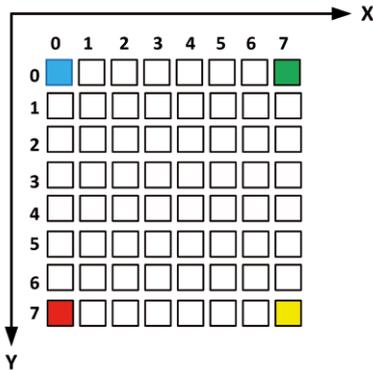


Figure 12.19 Motif LED pour l'exemple 3

Solution 3

Entrez l'énoncé suivant dans la fonction :

```
msg.payload = "0,0,blue,0,7,red,7,0,green,7,7,yellow" renvoie msg;
```

Exemple 4

Régler les LED dans une colonne de 4 pixels de large sur bleu, comme indiqué à la figure 12.20.

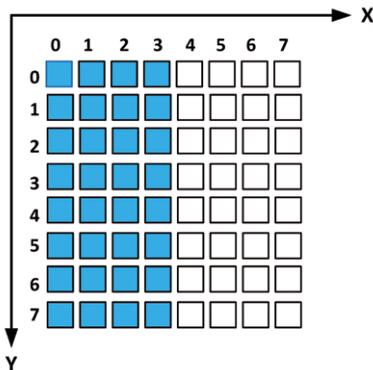


Figure 12.20 Motif LED pour l'exemple 4

```
msg.payload = "0-3,*,blue";  
renvoie msg;
```

Exemple 5

Régler toutes les LED de la deuxième rangée sur le bleu, comme indiqué à la figure 12.21.

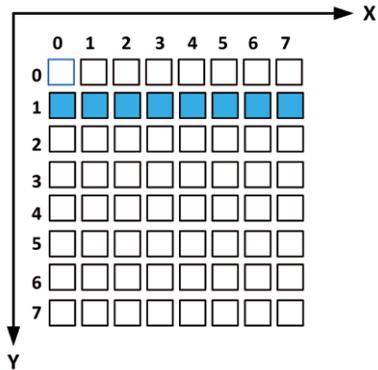


Figure 12.21 Motif DEL pour l'exemple 5

Solution 5

Entrez l'énoncé suivant dans la fonction :

```
msg.payload =    "*",1,blue";
renvoie msg;
```

Exemple 6

Régler les voyants diagonaux sur rouge, comme indiqué à la figure 12.22.

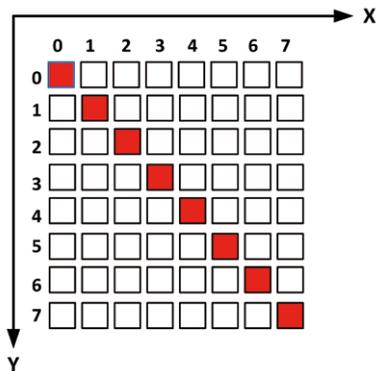


Figure 12.22 Motif LED pour l'exemple 6

Solution 6

Entrez l'énoncé suivant dans la fonction :

```
msg.payload = "0,0,red,1,red,2,2,red,3,red,4,4,red,5,red,6,6,red,7,7, red";
retour msg
```

Nous aurions aussi pu écrire la fonction suivante pour mettre les LED diagonales en rouge :

```

var L="";
pour(i=0; i <= 7; i++) {

    if(i != 7)
        L = L + String(i)+";"+String(i)+";",
        rouge,""; sinon
        L = L + String(i)+";";"+String(i)+";", rouge";
    }
    msg.payload = Chaîne(L);
    renvoie msg;

```

12 . 10 Projet 64 – Lumières LED clignotantes aléatoires de couleurs aléatoires

Description : Dans ce projet, nous allons faire clignoter des LED sur la matrice de LED au hasard. Les couleurs sont également aléatoires.

Viser : Le but de ce projet est de montrer comment des nombres aléatoires peuvent être utilisés pour clignoter des LED au hasard, ayant des couleurs aléatoires.

Programme de flux Node-RED : Le programme de flux est montré dans la figure 12.23, et il se compose de 11 noeuds : **injecter** noeud qui se répète toutes les secondes, 5 **aléatoire** nombre de noeuds. Deux de ces noeuds génèrent des nombres entre 0 et 7 qui correspondent aux coordonnées LED, les 3 autres noeuds génèrent des nombres de 0 à 255 pour correspondre aux couleurs RGB, un **rejoindre** noeud qui joint tous les nombres aléatoires et les passe à un **fonction** noeud. Le **fonction** les chaînes de sortie du noeud clignotent les LED ayant des couleurs aléatoires. A **retard** Le noeud est également utilisé pour éteindre toutes les LED chaque seconde afin qu'une seule LED soit allumée à tout moment.

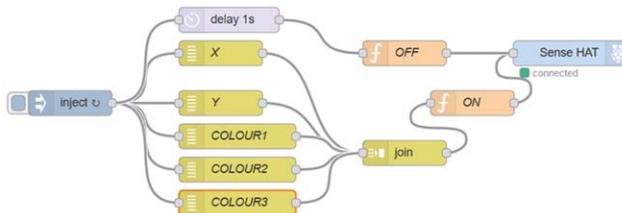


Figure 12.23 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** noeud à répéter chaque seconde
- Créer deux **aléatoire** noeuds nommés **X** et **Y** et les configurer pour générer ran- Nombre entre 0 et 7.
- Créer trois autres **aléatoire** numéros nommés **COLOUR1**, **COLOUR2** et **Col- Nos 3** et les configurer pour générer des nombres aléatoires entre 0 et 255

- Créer un **rejoindre** nœud et le configurer comme indiqué sur la figure 12.24

Figure 12.24 Configuration de la jointure des nœuds

- Créer un **fonction** nœud nommé **Sur** et entrer les instructions suivantes dans ce nœud :

```
var x = msg.payload[0];
var y = msg.payload[1];
var c1 = msg.payload[2];
var c2 = msg.payload[3];
var c3 = msg.payload[4];
msg.payload = x + "," + y + "," + c1 + "," + c2 + "," + c3; retour msg;
```

- Créer un autre **fonction** nœud nommé **off** et entrez les déclarations suivantes :

```
msg.payload = "*,*,off";
renvoie msg;
```

- Créer un **Sense HAT** Sortie du nœud et rejoindre tous les nœuds. Vous devriez voir des LED tourner **Sur** et **off** Au hasard dans différentes couleurs.

12 . 11 Project 65 – Affichage de la température par le compte des DEL

Description : Nous avons vu plus tôt dans ce chapitre comment contrôler les LED sur la matrice de LED. Dans ce projet, nous allons afficher la température en temps réel en allumant le nombre correct de LED.

Il y a 64 LED dans la matrice de LED. Pour plus de simplicité, nous allons assigner à chaque LED une température de 0,5°C afin que la température puisse être affichée de 0°C (correspondant à pas de LED ON) jusqu'à 32°C (correspondant à toutes les 64 LED ON). Par conséquent, en comptant le nombre de LED qui sont allumées et en divisant le nombre par 2, nous trouvons la température.

Viser : Le but de ce projet est de montrer comment la carte Sense HAT peut être utilisée pour lire la température ambiante et afficher la température en allumant le bon nombre de LED.

Programme de flux Node-RED : Le programme de flux est montré dans la figure 12.25, et il se compose de 4 nœuds **Saisie HAT de détection** nœud pour lire la température ambiante, une **fonction** nœud avec deux sorties pour formater les lectures et aussi pour ÉTEINDRE l'écran afin qu'il puisse être mis à jour, un nœud de retard pour limiter les messages, et un **Sense HAT output** Pour allumer le nombre approprié de LED.

```

▶ { key: "LEFT", state: 2 }
20/12/2019, 15:08:50 node: f6527996.096db8
joystick : msg.payload : Object
▶ { key: "LEFT", state: 2 }
20/12/2019, 15:08:50 node: f6527996.096db8
joystick : msg.payload : Object
▶ { key: "LEFT", state: 2 }
20/12/2019, 15:08:50 node: f6527996.096db8
joystick : msg.payload : Object
▶ { key: "LEFT", state: 2 }

```

Figure 12.25 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **Saisie de sens** nœud et ensemble **Sorties** aux événements environnementaux
- Créer un **fonction** nœud nommé **Format** avec deux sorties et entrez le fol- Les instructions à l'intérieur de ce nœud. Variable **T** extrait la température de Sense HAT. Variable **OnCount** calcule le nombre de LED qui doivent être allumées. Par exemple, si la température est de 22 oC, 44 LED doivent être allumées. Le reste du code est exécuté si la température est inférieure ou égale à 32oC. Variable **rangées** est le nombre de rangées de DEL qui doivent être allumées. Par exemple, si la température est de 22 oC, alors **rangées** est égal à 5 donc les lignes : 0, 1, 2, 3, 4 doivent être ON. **cols** est les DEL restantes qui doivent être allumés. Par exemple, si la température- ture est de 22 o C alors **cols** est égal à 4. Par exemple, 4 autres LED doivent être allumées. Trois **pour** boucles sont utilisées dans la fonction. La boucle externe itère de 0 à **rangées**, la boucle interne itère 8 fois de sorte que toutes les LED d'une rangée donnée sont allumées. La dernière boucle allume les LED restantes. Variable **L** stocke le motif qui sera produit par le nœud.

La deuxième sortie de la fonction envoie du texte **off** à l'écran pour effacer l'écran afin que celui-ci puisse être mis à jour. Lorsqu'une nouvelle DEL est allumée ou éteinte, celles qui sont déjà allumées restent allumées. Par exemple, si la température est de 22 oC, alors nous avons 44 LED ON. Si la température descend à 20 oC, le compte des ON ne descend pas à 40. Bien qu'une commande soit envoyée pour allumer seulement 40 LED, celles qui sont déjà allumées ne s'éteignent pas. Par conséquent, nous effaçons l'écran en désactivant toutes les LED et puis nous mettons à jour avec les nouvelles données.

Si la température est supérieure à 32 °C, alors la DEL du milieu (à la coordonnée 4, 4) est mis en rouge pour indiquer que la température est hors limite :

```

var L="";
var T = msg.payload.temperature; var
OnCount = parseInt(2*T); if(T <= 32)
{

    var rows = parseInt(OnCount / 8); var
    cols = OnCount-rows*8;

    pour(i = 0; i < rows; i++) {

        pour (j = 0 ; j <= 7; j++)
        {
            L = L + String(i)+"&quot;,&quot;+String(j)+"&quot;,&quot;,red,&quot;;
        }
    }
    pour (j=0; j < cols; j++) {

        L = L + String(rows)+"&quot;,&quot;+String(j)+"&quot;,&quot;,red,&quot;;
    }

    var l = L.length;
    msg.payload=L.substr(0,l-1);
}
autrement
msg.payload = "&quot;4,4,red&quot;;
var f = {payload : "&quot;*,*,off&quot;};
return [msg, f];

```

- Créer un **délai** node avec le **Action** définir la limite de débit et la configurer pour envoyer 1 message toutes les 30 secondes. Ce nœud est connecté à la deuxième sortie du nœud **fonction** pour que l'écran soit effacé en envoyant le texte.
- Créer un **Sense HAT output** node et joindre tous les nodes, cliquez **Déployer**. Vous devriez voir que la température affichée sur la matrice LED avec le nombre de LED allumées. Dans l'exemple illustré à la figure 12.26, 51 DEL sont allumées. Par conséquent, la température est de $51/2 = 25,5$ °C. Notez que, comme décrit précédemment, la lecture de température du Sense HAT n'est pas précise car le capteur est proche du processeur du Raspberry Pi.

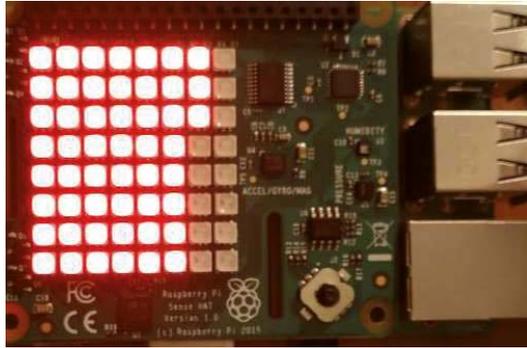


Figure 12.26 Affichage de la température 25,5 oC

12 . 12 Affichage et défilement des données sur la matrice LED

Les données peuvent être facilement affichées et défilées sur la matrice LED en les chargeant dans `msg.payload`. Par exemple, pour afficher et faire défiler du texte **Bonjour**, entrer la commande suivante dans une fonction :

```
msg.payload = "Hello";
renvoie msg;
```

Nous pouvons spécifier la couleur de fond, la couleur du texte et la vitesse de défilement en spécifiant : **retour- au sol**, **couleur**, et **vitesse** respectivement. La vitesse peut prendre les valeurs 1 à 5, où 1 est lent et 5 est rapide (par défaut est 3). Un exemple est montré ci-dessous qui spécifie la couleur de fond comme jaune, la couleur du texte comme rouge et la vitesse comme 2 :

```
var txt = {payload : "Hello"};
txt.background = "yellow"; txt.color =
"red";
txt.speed = 2;
retour txt;
```

Si le texte est un caractère unique, il s'affichera sans défilement. Pour faire défiler un seul caractère, ajouter un espace vide après lui, par ex. "X ".

12 . 13 Projet 66 – Défilement des valeurs de pression sur la matrice LED

Description : Dans ce projet, nous allons faire défiler les lectures de pression sur la matrice LED. La couleur de fond sera réglée sur le blanc, la couleur du texte sur le rouge et la vitesse sur 1.

Viser : Le but de ce projet est de montrer comment les données peuvent être défilées sur la matrice LED.

Programme de flux Node-RED : Le programme de flux est montré dans la figure 12.27, et il se compose de 3 noeuds **Saisie HAT de détection** noeud pour lire la pression ambiante, un **fonction** noeud pour formater les lectures, un noeud de retard pour limiter le taux des messages à 1 toutes les 5 secondes et **Sense HAT output** noeud pour afficher les données sur la matrice LED.

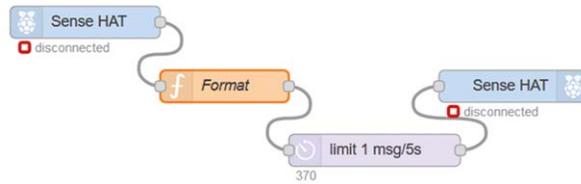


Figure 12.27 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **Saisie de sens** nœud comme dans le projet précédent
- Créer un **fonction** nœud et entrer les instructions suivantes :

```
var T = {payload : msg.payload.pressure}; T.color =
"red";
T.background = "blanc";
T.vitesse = 1;
retourner T;
```

- Créer un **retard** nœud et définir **Action** Limite de débit pour limiter 1 message toutes les 5 secondes. Ce nœud est nécessaire pour ralentir les messages provenant de Sense CHAPEAU.
- Créer un **Sense HAT output** node comme précédemment. Joignez tous les nœuds et cliquez **Déployer**. La pression ambiante doit s'afficher et défiler sur la matrice DEL.

Quelques autres commandes utiles de Sense HAT sont :

Faire pivoter l'écran

Pour faire pivoter l'écran, saisissez la commande R suivie de l'angle. L'angle doit être 0, 90, 80 ou 270. Dans l'exemple suivant, l'écran est tourné de 180 degrés :

```
var T = "Bonjour" ; msg.payload =
"R90 n" + T; return msg;
```

Retournez l'écran

Pour retourner l'écran, saisissez la commande F suivie de V (verticale) ou H (horizontale).

Régler la luminosité de l'écran

Pour régler la luminosité de l'écran, saisissez la commande D suivie du niveau. Le niveau doit être 0 (faible) ou 1 (élevé).

12 . 14 Résumé

Dans ce chapitre, nous avons appris comment utiliser les nœuds d'entrée et de sortie Sense HAT dans divers projets.

Dans le prochain chapitre, nous allons apprendre à utiliser Node-RED avec l'Arduino et développer plusieurs projets avec l'Arduino.

Chapitre 13 • Node-RED avec Arduino Uno

13 . 1 Aperçu

Arduino Uno est l'une des cartes de développement de microcontrôleurs les plus utilisées- rently. Il y a des centaines de projets disponibles sur Internet en utilisant l'Arduino Uno avec des capteurs, des actionneurs et des dispositifs d'affichage.

Ce chapitre traite de l'utilisation de Node-RED avec l'Arduino Uno. Les détails du matériel de l'Arduino Uno ne sont pas donnés dans ce chapitre car on suppose que les lecteurs connaissent bien l'Arduino Uno et ont développé des programmes en utilisant cette carte de développement. Les lecteurs intéressés peuvent trouver de nombreux tutoriels, notes d'application, fiches techniques et projets pour l'Arduino Uno sur Internet. Le chapitre décrit comment installer et utiliser Node-RED pour l'Arduino Uno. Plusieurs projets sont décrits dans le chapitre en utilisant Node-RED.

13 . 2 Installation de Node-RED pour Arduino Uno

Avant d'utiliser Node-RED sur notre Arduino Uno, nous devons l'installer sur notre ordinateur Windows. Les étapes sont les suivantes :

- Installer **noëud . js** (64 bits ou 32 bits selon votre ordinateur) depuis le site suivant :

<https://nodejs.org/fr/téléchargements>

- Ouvrez le **Commandement** Lancez l'application sur votre ordinateur en tant qu'administrateur et entrez la commande suivante (voir la figure 13.1) :

C : Windows system32> **npm install -g --unsafe-perm node-red**

```
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>npm install -g --unsafe-perm node-red
C:\Users\Dogan Ibrahim\AppData\Roaming\npm\node-red -> C:\Users\Dogan Ibrahim\AppData\Roaming\npm\node_modules\node-red\
red.js
C:\Users\Dogan Ibrahim\AppData\Roaming\npm\node-red-pi -> C:\Users\Dogan Ibrahim\AppData\Roaming\npm\node_modules\node-r
ed\bin\node-red-pi

> bcrypt@3.0.6 install C:\Users\Dogan Ibrahim\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

node-pre-gyp WARN Using request for node-pre-gyp https download
[bcrypt] Success: "C:\Users\Dogan Ibrahim\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt\lib\binding\bcry
pt_lib.node" is installed via remote
+ node-red@1.0.3
added 351 packages from 348 contributors in 18.211s

C:\WINDOWS\system32>
```

Illustration 13.1 Installation du noëud RED

- Pour tester l'installation, entrez la commande suivante :

C : Windows system32> **noeud-rouge**

- Vous devriez voir des messages similaires à ceux de la figure 13.2. Si vous pouvez voir ces messages, cela signifie que Node-RED est installé correctement sur votre PC

```
C:\WINDOWS\system32\node-red
22 Dec 18:09:45 - [info]

Welcome to Node-RED
=====

22 Dec 18:09:45 - [info] Node-RED version: v1.0.3
22 Dec 18:09:45 - [info] Node.js version: v12.14.0
22 Dec 18:09:45 - [info] Windows_NT 10.0.18362 x64 LE
22 Dec 18:09:46 - [info] Loading palette nodes
22 Dec 18:09:46 - [info] Settings file : C:\Users\Dogan Ibrahim\.node-red\settings.js
22 Dec 18:09:46 - [info] Context store : 'default' [module-memory]
22 Dec 18:09:46 - [info] User directory : C:\Users\Dogan Ibrahim\.node-red
22 Dec 18:09:46 - [warn] Projects disabled : editorTheme.projects.enabled=false
22 Dec 18:09:46 - [info] Flows file : C:\Users\Dogan Ibrahim\.node-red\flows_DESKTOP-U7VQMOI.json
22 Dec 18:09:46 - [info] Creating new flow file
22 Dec 18:09:46 - [warn]
```

Figure 13.2 Test de votre installation Node-RED

- Maintenant, démarrez votre navigateur web et entrez l'adresse suivante pour démarrer votre Node-RED dans votre navigateur (ne fermez pas votre fenêtre de commande) :

<http://127.0.0.1:1880>

- Vous devriez maintenant voir l'écran de démarrage du Node-RED comme indiqué dans la figure 13.3 (seule une partie de l'écran est affichée ici).



Figure 13.3 Écran de démarrage Node-RED

- Installer le nœud Arduino. Cliquez **Menu -> Gérer la palette** et cliquez **Installer**
- Entrer **noeud-rouge-noeud-arduino** et cliquez **installer**
- Deux nœuds doivent être ajoutés à la palette de nœuds : **arduino dans** et **arduino out**

L'Arduino doit être connecté à l'ordinateur hôte via une connexion série USB. Nous ne pouvons pas utiliser les deux nœuds Arduino IDE et Arduino en même temps car ils entreront en conflit. Nous devons sortir de l'IDE Arduino pour utiliser Node-RED. Sinon, nous devons arrêter Node-RED si nous voulons programmer l'Arduino en utilisant l'IDE.

Le **Firmata** protocole est utilisé pour la communication entre un Arduino et l'hôte computer, en fournissant un accès direct aux broches GPIO. Avant de développer nos programmes, nous devons charger le Firmata dans notre Arduino Uno. Les étapes sont les suivantes :

- Arrêter le noeud-ROUGE
- Connectez votre Arduino au PC via le câble USB
- Démarrer l'EDI Arduino
- Sélectionnez Arduino/Genuino Uno comme carte cible
- Prendre note du nom du port série (p. ex., COM5)
- Cliquez sur Fichiers -> Exemples -> Signatures -> Signatures standard
- Cliquez sur Sketch -> Verify/Compile puis cliquez sur Sketch -> Upload
- Sortie de l'EDI Arduino

Vous devriez maintenant démarrer Node-RED sur votre Windows, puis lancer le tableau de bord dans votre navigateur web :

C : Windows system32> **noeud-rouge**

<http://127.0.0.1:1880>

Nous sommes maintenant prêts à développer des programmes Arduino en utilisant Node-RED. Un exemple simple-programme est donné dans la section suivante pour montrer comment Node-RED peut être utilisé avec l'Arduino Uno.

13 3 Projet 67 – LED clignotante

Description : Il s'agit d'un programme très simple où la LED sur la carte Arduino Uno au port 13 est clignotée toutes les secondes.

Viser : Le but de ce projet est de montrer comment la LED embarquée sur le port broche 13 peut être clignotée toutes les secondes.

Programme de flux Node-RED : La figure 13.4 montre le programme flow qui se compose de seulement 3 nœuds : un nœud inject qui répète chaque seconde, un nœud de fonction qui bascule sa sortie et un nœud Arduino out.



Figure 13.4 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **injecter** noeud, nommez-le comme **1 seconde** et le configurer pour répéter chaque seconde
- Créer un **noeud de fonction**, nommez-le comme **Toggle LED** et insérez l'état suivant- Les données contenues dans ce noeud. Dans ce noeud, une variable de contexte est basculée de sorte que sa valeur change entre 0 et 1. Rappelez-vous que les variables de contexte gardent leurs valeurs à l'intérieur du noeud :

```
context.led = ! context.led || 0; msg.payload  
= context.led; return msg;
```

- Créer un **arduino out** et le configurer comme indiqué dans la Figure 13.5. Notez que le numéro de port série trouvé plus tôt (COM5) est utilisé dans ce noeud.



Figure 13.5 Configuration du noeud arduino

- Rejoindre tous les noeuds comme indiqué dans la figure 13.4 et cliquer sur **Déployer**.
- Vous devriez voir le voyant LED à bord clignoter toutes les secondes

13 . 4 Project 68 – Affichage de la température ambiante dans la fenêtre de débogage

Description : Dans ce projet, la température ambiante est mesurée à l'aide d'un- perature capteur puce.

Viser : Le but de ce projet est de montrer comment **arduino dans** Le noeud peut être utilisé pour lire les données d'entrée analogiques.

Schéma de circuit : Une puce de capteur de température analogique du type LM35DZ est utilisée dans ce projet. Le capteur est connecté à +5V et sa sortie est directement connectée à l'entrée analogique A0 de l'Arduino Uno. La figure 13.6 montre le schéma du projet.

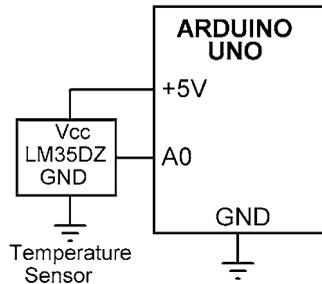


Figure 13.6 Schéma du projet

Programme de flux Node-RED : La figure 13.7 montre le programme de flux qui se compose de seulement 3 nœuds : **arduino dans** nœud pour recevoir les relevés de température, un **fonction** nœud pour- Les données et une **déboguer** node pour afficher les données dans la fenêtre Debug.

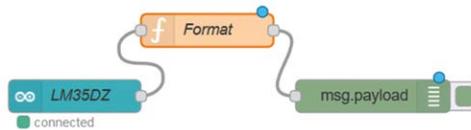


Figure 13.7 Programme de flux du projet

Les étapes sont :

- Créer un **arduino dans** nœud et configurer comme indiqué dans la figure 13.8

Arduino	COM5
Type	Analogue pin
Pin	0
Name	LM35DZ

Figure 13.8 Configuration du nœud arduino dans

- Créer un **fonction** nœud et entrer les instructions suivantes :

```
var T = msg.payload * 5000,0 / 1024,0;
var mv = T / 10,0 ;
msg.payload = "T = " + mv.toFixed(2) + "C"; renvoie msg;
```

- Créez un noeud de débogage, joignez les noeuds, cliquez sur Déployer.
- La température s'affiche dans la fenêtre de débogage comme indiqué sur la figure 13.9



Figure 13.9 Affichage de la température

13 . 5 Projet 69 – Affichage de la température ambiante dans le tableau de bord

Description : Dans ce projet, la température ambiante est mesurée comme dans le pro précédent- et s’affiche dans le texte du nœud du Tableau de bord.

Viser : Le but de ce projet est de montrer comment les nœuds du tableau de bord peuvent être utilisés dans Arduino pro- jects.

Schéma de circuit : Le schéma du projet est celui de la figure 13.6

Programme de flux Node-RED : La figure 13.10 montre le programme de flux qui se compose de seulement 3 nœuds **arduino dans** nœud pour recevoir les relevés de température, un **fonction** nœud pour- Les données et un nœud de texte du tableau de bord.

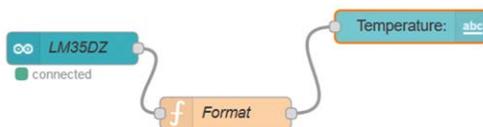


Figure 13.10 Programme de flux du projet

Avant d’utiliser un nœud de tableau de bord, nous devons installer les palettes de tableaux de bord. Les étapes sont :

- Cliquez **Menu -> Gérer la palette** et cliquez **Installer**
- Entrer **noeud-rouge-tableau de bord** et cliquez **installer**
- Vous devriez voir les nœuds du tableau de bord ajoutés à votre palette de nœuds

Les étapes suivantes sont nécessaires pour élaborer le programme de flux :

- Créer un **arduino dans** nœud comme dans le projet précédent
- Créer un **fonction** nœud comme dans le projet précédent

- Créer un tableau de bord **texte** nœud et le configurer comme dans la figure 13.11

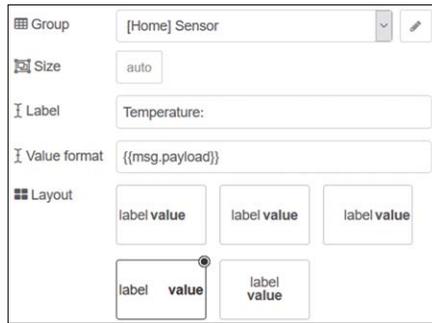


Figure 13.11 Configuration du nœud de texte

- Joindre tous les nœuds et cliquer **Déployer**. Démarrez le tableau de bord en saisissant les éléments suivants dans votre navigateur Web. La température ambiante doit s'afficher comme indiqué sur la figure 13.12 (notez que le style est réglé à Dark dans cet affichage) :

127.0.0.1:1880/ui/



Figure 13.12 Affichage de la température avec le texte du nœud

13 . 6 Projet 70 – Affichage de la température ambiante sous forme de jauge et de graphique

Description : Dans ce projet, la température ambiante est mesurée comme dans le pro précédent- Le projet est affiché sous la forme d'une jauge et d'un graphique.

Viser : Le but de ce projet est de montrer comment la jauge et le graphique du tableau de bord peuvent être utilisés dans les projets Arduino.

Schéma de circuit : Le schéma du projet est celui de la figure 13.6

Programme de flux Node-RED : La figure 13.13 montre le programme de flux qui se compose de 4 nœuds : un Arduino dans le nœud pour lire la température, un nœud de fonction pour formater la température- ature, un jauge et un graphique pour afficher la température.

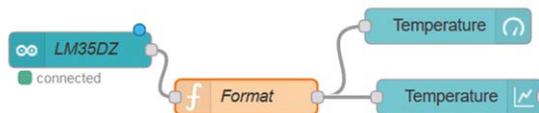


Figure 13.13 Programme de flux du projet

Les étapes sont :

- Créer un **arduino dans** nœud comme dans le projet précédent
- Créer un **fonction** nœud et entrer les instructions suivantes :

```
var T = msg.payload * 5000,0 / 1024,0; var mv = T
/ 10,0;
var Temp = mv.toFixed(2);
msg.payload = Temp;
retourner le message;
```

- Créer un **jauge** Node et configurer comme indiqué sur la figure 13.14

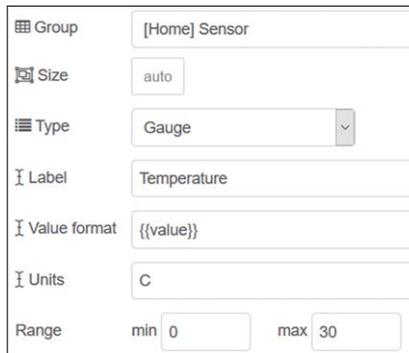


Figure 13.14 Configuration de la jauge de nœud

- Créer un **graphique** Node et configurer comme indiqué sur la figure 13.15

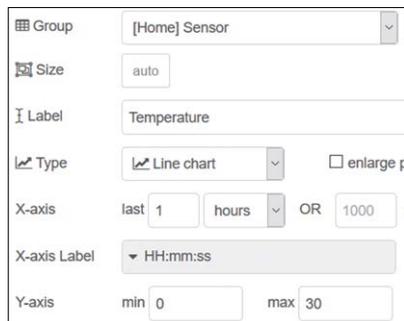


Figure 13.15 Configuration du diagramme de nœud

- Cliquez **Déployer** et vous devriez voir la température affichée dans le tableau de bord à l'aide d'une jauge et d'un graphique comme indiqué sur la figure 13.16

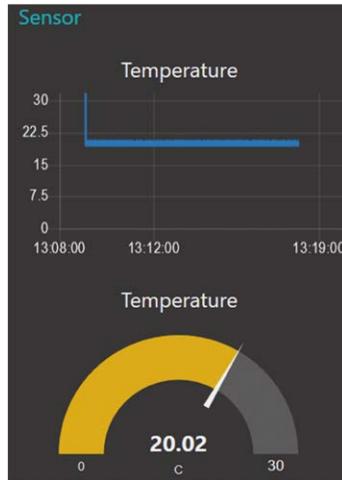


Figure 13.16 Affichage de la température à l'aide d'une jauge et d'un graphique

13 . 7 Utilisation du port série Arduino Uno

Les lecteurs constateront qu'il y a un nombre limité de nœuds lors de l'utilisation de l'Arduino avec Node-RED. Par exemple, au moment de la rédaction du présent ouvrage, il n'existait pas de nœud LCD pouvant être utilisé avec l'Arduino. La solution à ce problème est la suivante :

Nous pouvons connecter l'Arduino à un des ports USB du Raspberry Pi et ensuite écrire le code en utilisant l'IDE Arduino. La sortie peut ensuite être envoyée au Raspberry Pi via le port série. Le Raspberry Pi peut recevoir les données en utilisant le nœud série et les traiter comme nous l'avons vu dans les chapitres précédents de ce livre. Cette approche nécessite bien sûr du code à développer pour l'Arduino.

Un exemple de projet est donné ci-dessous en utilisant cette idée.

13 . 7 . 1 Projet 71 – Utilisation du DHT11 avec l'Arduino

Description : Dans ce projet, un capteur de température et d'humidité DHT11 est utilisé et les données de température et d'humidité ambiantes sont affichées sous la forme d'une jauge toutes les 5 secondes.

Viser : Le but de ce projet est de montrer comment le nœud série peut être utilisé pour recevoir des données de l'Arduino Uno.

Schéma de circuit : Le schéma de circuit du projet est illustré à la figure 13.17. Le DHT11 est connecté au port numérique pin 3 de l'Arduino Uno. L'Arduino Uno est connecté à un des ports USB du Raspberry Pi via un câble USB. Les données de température et d'humidité sont envoyées par ce câble.

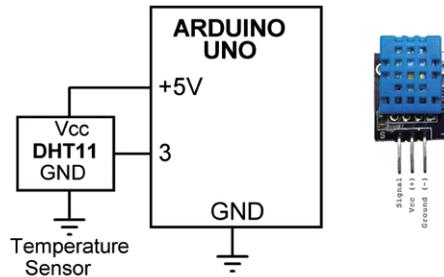


Figure 13.17 Schéma du projet

Programme Arduino Uno : Avant d'utiliser le DHT11 avec l'Arduino Uno, nous devons inclure la bibliothèque DHT11 dans notre bibliothèque Arduino. Les étapes sont les suivantes :

- Démarrer l'EDI Arduino
- Cliquez **Sketch** > **Inclure la bibliothèque** > **Gérer les bibliothèques**
- Enter DHT11
- Cliquez pour installer **Bibliothèque de capteurs DHT par Adafruit version 1 . 3 . 8** (ou une version ultérieure)
- Fermer la fenêtre
- Fermer et redémarrer l'EDI Arduino

Le programme Arduino (programme : **dht11arduino**) est représentée à la figure 13.18. Au début de l'étude, on peut voir- Lors de l'exécution du programme, la broche 3 est assignée à DHT11 et DHT11 est défini comme le type de capteur (pas DHT22). Dans la routine de configuration, le port série est initialisé à 9600 Baud et le DHT11 est initialisé. Le reste du programme est exécuté à l'intérieur de la boucle principale. Ici, l'humidité et la température sont lues par le capteur puis envoyées au port série. Ce processus se répète après 5 secondes.

```
#include <DHT.h>
#define DHTPin 3 // DHT11 pin
#define DHTType DHT11 // Type DHT11
DHT dht (DHTPin, DHTType); // Initialiser

Chaîne Humidité, température;
flotter Hum, temp;

void setup()
{
  Serial.begin(9600); // Démarrer série
  dht.begin(); // Initialie DHT11
```

```

}

boucle vide()
{
    Hum = dht.readHumidity(); Temp=           // Obtenir l'humidité
    dht.readTemperature(); Humidity =       // Obtenir la température
    String(Hum); Temperature =              // Pour la chaîne
    String(Temp); Serial.print(Humidity);   // Pour la chaîne
    Serial.print(",");                      // Send to serial port // Insérer
    Serial.println(Temperature);            une virgule
    delay(5000);                           // Envoi au port série //5
                                           secondes de retard
}

```

Figure 13.18 Programme Arduino Uno

Programme de flux Node-RED : Redémarrez votre Raspberry Pi, puis lancez Node-RED sur votre Raspberry Pi. Démarrer le tableau de bord en saisissant (saisir l'adresse IP de votre propre Raspberry Pi) sur votre PC :

192.168.1.202:1880

La figure 13.19 montre le programme de flux qui se compose de 4 nœuds : un nœud série pour lire les données de température et d'humidité de l'Arduino, un nœud de fonction pour formater les données reçues, deux jauges pour afficher la température et l'humidité.



Figure 13.19 Programme de flux du projet

Les étapes sont :

- Créer un **en série dans** avec la configuration illustrée dans la figure 13.20. Notez que le nom du port série est défini sur **/dev/ttyACM0**, qui est le nom du port USB Raspberry Pi.

Figure 13.20 Configuration du nœud serial

- Créer un **fonction** nœud avec deux sorties et entrez les instructions suivantes (ou, utilisez la fonction split(", ")) pour extraire l'humidité et la température :

```
var H = {payload : msg.payload.substr(0,5)}; var T  
= {payload : msg.payload.substr(6,5)}; return [H, T];
```

- Créer deux **jauge** nœuds. Pour l'humidité, le nœud définit **Étiquette** à l'humidité, **min** et **max valeurs à 0 et 100** et le **Unités en % respectivement. Pour le tempérément-ature**, le noeud définit la **Étiquette à la température**, **min et max valeurs 0 et 30**, et **Unités à C**
- Joindre tous les nœuds, cliquez **Déployer**, et lancer le tableau de bord en saisissant votre propre adresse IP :

192.168.1.202:1880/ui/

- Vous devriez voir l'humidité et la température affichées comme indiqué sur la figure 13.21

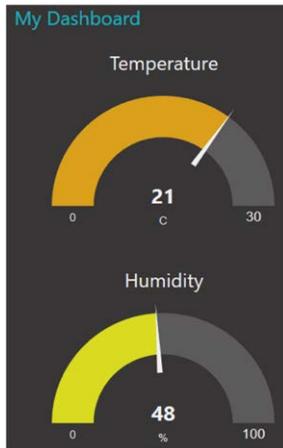


Figure 13.21 Affichage de l'humidité et de la température

13 . 8 Résumé

Dans ce chapitre, nous avons appris à utiliser Node-RED avec Arduino Uno, et plusieurs pro- Les actions sont données sur ce thème.

Dans le prochain chapitre, nous verrons comment utiliser Node-RED avec les processeurs ESP32.

Chapitre 14 • Utilisation du DevKitC ESP32 avec Node-RED

14 . 1 Aperçu

ESP32 est l'un des microcontrôleurs populaires utilisés actuellement par les étudiants, pratiquant elec- Les ingénieurs en électronique, les amateurs et bien d'autres personnes intéressées par la conception de systèmes embarqués basés sur des microcontrôleurs. Dans ce chapitre, nous allons apprendre à utiliser Node-RED avec des microcontrôleurs ESP32. Dans ce chapitre, nous utiliserons le tableau de développement DevKitC ESP32. On suppose que les lecteurs connaissent le processeur ESP32 et la carte de développement DevKitC ESP32 et l'ont utilisé dans au moins un projet. Dans le projet de ce chapitre, le processeur ESP32 est programmé en utilisant l'IDE Arduino. On suppose que l'IDE Arduino est chargé avec la **Module de développement ESP32** logiciel. Si ce n'est pas le cas, vous devez charger ce logiciel sur votre IDE Arduino. Les étapes d'installation du support ESP32 pour l'IDE Arduino sur un PC Windows sont données dans cette section. Si vous avez déjà installé ESP32 sur l'IDE Arduino, vous devez supprimer la **Espressif** dossier de votre répertoire Arduino avant de le réinstaller.

Les étapes suivantes sont nécessaires pour installer l'ESP32 sur l'IDE Arduino :

- Télécharger et installer la dernière version de l'EDI Arduino à partir du site web suivant :

<https://www.arduino.cc/en/Main/Software>

- Ouvrez votre IDE Arduino et cliquez **Fichier -> Préférences** pour ouvrir la fenêtre Préférences. Localiser la zone de texte **Autres URL du gestionnaire de la commission** au bas de la fenêtre et saisir le texte suivant comme indiqué à la figure 14.1. Si la zone de texte contient une autre URL, ajouter la nouvelle URL après l'avoir séparée par une virgule :

https://dl.espressif.com/dl/package_esp32_index.json

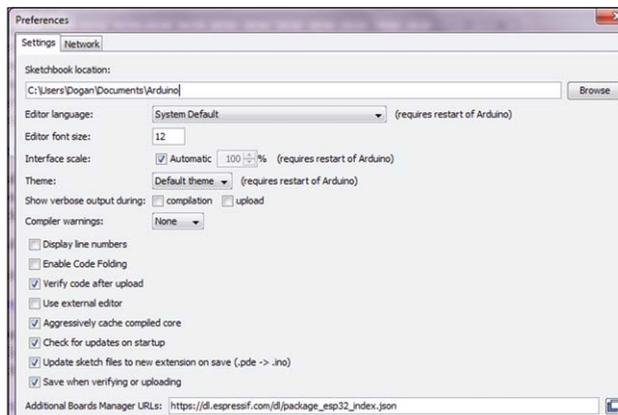


Figure 14.1 Fenêtre des préférences

- Cliquez D'accord

- Cliquez **Outils -> Conseil -> Directeurs de conseils** fenêtre et recherche ESP32 comme indiqué sur la figure 14.2

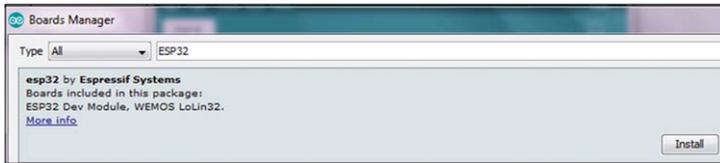


Figure 14.2 Recherche de ESP32

- Cliquez sur le **Installer** . Le message Installé s'affiche comme indiqué dans la figure 14.3. **Fermer** la fenêtre.

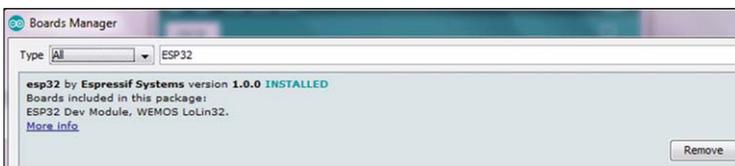


Figure 14.3 ESP32 installé

- Si le logiciel est installé correctement, alors vous devriez voir le module de développement ESP32 qui est la carte que nous allons utiliser avec notre DevKitC ESP32. Sélectionnez **Outils -> Carte ->ESP32 Dev Module** comme indiqué à la figure 14.4.

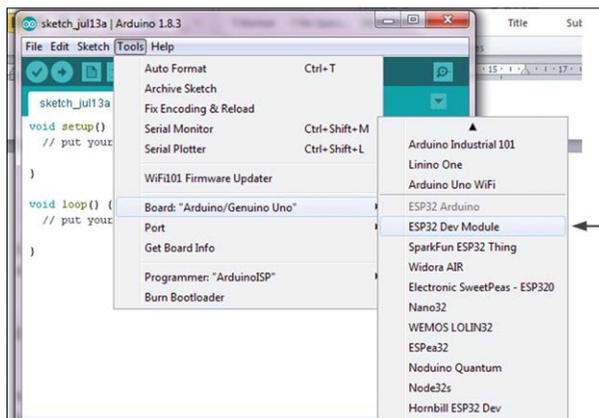


Illustration 14.4 Sélection du module de développement ESP32

Pour rappel, la figure 14.5 montre la configuration des broches de la carte de développement ESP32 DevKitC. Les lecteurs intéressés peuvent obtenir une grande quantité de données sur le processeur ESP32 à partir d'Internet :

Schéma de circuit : La figure 14.8 montre le schéma du projet. La LED est- raccordé à GPIO 21 de la DevKitC ESP32. La sortie de port série TXD (GPIO 14, broche physique 8) du Raspberry Pi est connectée au GPIO 23 de l'ESP32 DevKitC. Le DevKitC ESP32 possède 3 broches UART comme suit :

RXD (U0)	GPIO 3
TXD (U0)	GPIO 1
RXD (U1)	GPIO 9
TXD (U1)	GPIO 10
RXD (U2)	GPIO 16
TXD (U2)	GPIO 17

U0 est utilisé par le port de communication série USB vers PC. Dans certains appareils, U2 est réservé au flash embarqué et peut ne pas être disponible pour la communication série. En général, la plupart des autres broches peuvent être configurées pour une communication série. Dans ce projet GPIO, 23 est utilisé.

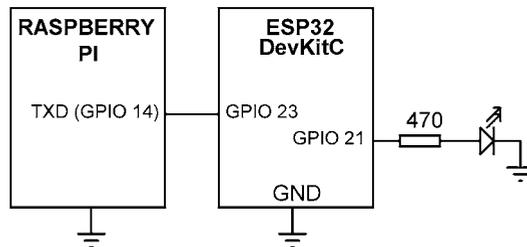


Figure 14.8 Schéma du projet

Programme ESP32 DevKitc : La liste des programmes (programme : `esp32nodered`) est représenté dans la Figure 14.9, qui est écrit en utilisant l'IDE Arduino. Au début du programme, **HardwareSerial** la bibliothèque est incluse dans le programme, les broches de port RXD et TXD (non utilisées dans ce projet) sont définies, et la LED est assignée au port 21 du GPIO. Dans la routine de configuration, le port série est configuré à 9600 bauds, 8 bits, pas de parité et 1 bit d'arrêt. La DEL est également configurée comme sortie. Le reste du programme est exécuté en boucle. À l'intérieur de cette boucle, le programme attend de recevoir des données du Raspberry Pi. Les données reçues sont stockées dans une variable de chaîne appelée **tampon**. Si les données reçues sont **On#** (où # est le caractère de terminateur) alors la LED est allumée. Si, d'autre part, les données reçues sont **Off#** puis la LED est éteinte. Le programme utilise la fonction de réception des données du port série `readStringUntil()` qui lit une chaîne à partir du port série jusqu'à ce que le caractère spécifié soit trouvé.

```
#include "HardwareSerial.h"
#define RXD 23
#define TXD 22

Serveur HardwareSerial(1); Chaîne
de mémoire tampon;
#define LED 21
```

```

void setup()
{
  pinMode (LED, OUTPUT);
  Ser.begin(9600, SERIAL_8N1, RXD, TXD);
}

boucle vide()
{
  while(Ser.available() > 0) {

    buffer = Ser.readStringUntil('#'); if(buffer[0] == 'O' &&
    buffer[1] == 'N')
      digitalWrite (LED, HIGH);
    sinon if(buffer[0] == 'O' && buffer[1] == 'F' && buffer[2] == 'F')
      digitalWrite (LED, LOW);
  }
}

```

Figure 14.9 Liste du programme ESP32 DevKitC

Cliquez **Sketch** -> **Compiler/Vérifier** pour compiler le programme. Après une compilation réussie, vous devez appuyer sur la touche **Boot** bouton sur ESP32 DevKitC (le bouton en bas à droite) et maintenez-le enfoncé et cliquez **Sketch** -> **Télécharger**. Lorsque le téléchargement est terminé, relâchez la **Boot** Appuyez sur le bouton et **FR** pour réinitialiser et démarrer le programme en cours d'exécution sur votre ESP32 DevKitC.

Programme de flux Node-RED : La figure 14.10 montre le programme de flux qui se compose de seulement 3 nœuds : deux nœuds d'injection pour envoyer les commandes ON# ou OFF# au port série, et un port de sortie série pour envoyer les commandes au DevKitC ESP32. Vous devez désactiver le port série de la console (/dev/ttyS0) comme décrit au chapitre 11 afin que vous puissiez utiliser le port série du Raspberry Pi.

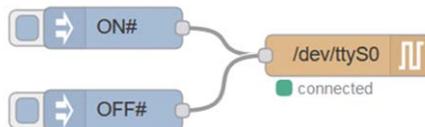


Figure 14.10 Programme de flux du projet

Les étapes sont les suivantes :

- Créer deux **injecter** nœuds, définissez la charge utile sur ON# dans l'un d'eux et sur OFF# dans l'autre.
- Créer un **série** nœud et le configurer comme indiqué sur la figure 14.11

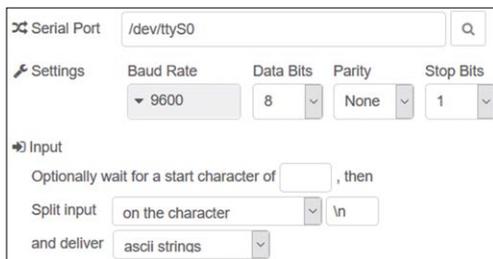


Figure 14.11 Configuration de la sortie série du nœud

- Rejoindre tous les nœuds comme dans la figure 14.10 et cliquer **Déployer**.
- Construire le circuit comme indiqué sur la figure 14.8. Cliquer sur le bouton de **injecter** Le nœud ON , la DEL doit s'allumer. Cliquez sur le bouton **injecter** nœud OFF#, la LED doit être éteinte.

14 . 4 Résumé

Dans ce chapitre, nous avons appris comment connecter le DevKitC ESP32 au Raspberry Pi en utilisant la liaison série à 9600 Baud. Node-RED a été lancé sur le Raspberry Pi. En envoyant des commandes du Raspberry Pi au DevKitC ESP32, nous pouvons facilement contrôler la LED. Les lecteurs intéressés peuvent développer des projets plus complexes basés sur le processeur ESP32 en utilisant la technique décrite dans ce chapitre.

Dans le prochain chapitre, nous allons apprendre à utiliser Amazon Alexa dans Node-RED basé projects.

Chapitre 15 • Utilisation d'Amazon Alexa dans les projets Node-RED

15 . 1 Aperçu

Alexa est un assistant virtuel basé sur le son développé par Amazon en 2014 et utilisé pour la première fois dans Amazon Echo et Amazon Echo Dot. Alexa interagit avec l'utilisateur par le son et peut obéir aux commandes données par un haut-parleur. Par exemple, Alexa peut jouer un morceau de musique nécessaire, il peut fournir des bulletins météorologiques, poser des questions, donner des rapports sur la circulation, donner les nouvelles actuelles, donner des recettes, traduire des mots dans d'autres langues et bien plus encore. Alexa peut également être utilisé pour contrôler un appareil par des commandes sonores et donc il peut être utilisé dans la domotique. L'appareil est activé en donnant le mot de réveil Alexa. L'utilisateur peut alors communiquer avec Alexa. Alexa est connectée au routeur Wi-Fi local de l'utilisateur et obtient les réponses aux questions de l'utilisateur à partir du cloud Amazon. Le dernier modèle d'Alexa au moment de la rédaction de ce livre était le 3rd génération. La figure 15.1 montre les 3rd Appareil Gen Alexa Dot.



Figure 15.1 3rd Gen Alexa Dot

Dans ce chapitre, nous allons utiliser Node-RED avec Alexa et Raspberry Pi et apprendre comment nous pouvons contrôler les ports GPIO de Raspberry Pi en donnant des commandes parlées à Alexa.

15 . 2 Projet 73 – Contrôle d'une DEL à l'aide d'Alexa

Description : Dans ce projet, une LED est connectée à l'un des ports GPIO du Raspberry Pi. La LED est allumée et éteinte en donnant des commandes vocales à Alexa.

Viser : Le but de ce projet est de montrer comment Alexa peut être utilisé avec Node-RED dans un Rasp- Projet berry Pi pour contrôler un appareil à distance.

Schéma de principe : Le schéma du projet est présenté à la figure 15.2.

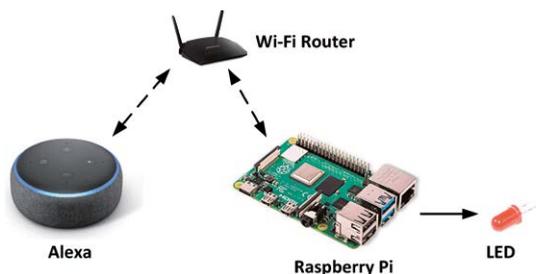


Figure 15.2 Schéma du projet

Schéma de circuit : Dans ce projet, une petite LED est connectée à la broche de port GPIO 2 du Raspberry Pi via une résistance de limitation de courant de 470 ohms.

Programme de flux Node-RED : Avant de développer notre programme flow, nous devons installer le nœud Alexa à notre palette de nœuds. Parce que les nœuds Alexa écoutent sur le port 80, il est important de démarrer votre Node-RED sur le Raspberry Pi avec une commande super user. i.e,

```
pi@framberrypi :~ $ sudo node-red-start
```

Ensuite, les étapes pour installer Alexa sont les suivantes :

- Cliquez **Menu -> Gérer la palette**, cliquez **Installer**
- Entrer **node-red-contrib-amazon.echo** et cliquez **installer**
- Vous devriez voir deux nouveaux nœuds appelés **hub d'écho amazon** et **dispositif d'écho Amazon** ajouté à votre palette de noeuds

Si vous constatez que vos nœuds GPIO Raspberry Pi ne sont pas répertoriés dans la palette de nœuds, vous pouvez les réinstaller en suivant les étapes suivantes :

- Cliquez **Menu -> Gérer la palette**, cliquez **Installer**
- Entrer **node-red-node-pi-gpio** et cliquez **installer**
- Nous sommes maintenant prêts à élaborer notre programme de flux.

La figure 15.3 montre le programme de flux qui se compose de seulement 4 nœuds : un **hub d'écho amazon**nœud, un dispositif d'écho Amazon **noeud**, un nœud de fonction, **et un rpi gpio out noeud**.

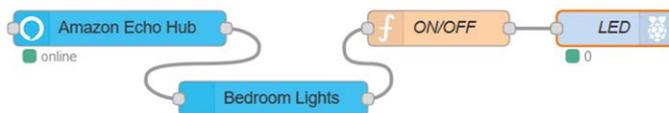


Figure 15.3 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **hub d'écho amazon** Le CEDEFOP a pour objectif de **Port** est réglé à 80 et **Données de processus** est réglé sur No.
- Créer un **dispositif d'écho Amazon** nœud et définir **Nom** Bedroom Lights (c'est le nom que Alexa connaîtra la LED. Vous devez choisir votre propre nom ici).

- Créer un **fonction** et le nommez ON/OFF. Entrez les instructions suivantes dans ce nœud. Ce nœud renvoie 1 si le message provenant d'Alexa est **sur**, ou il sort 0 est le message venant d'Alexa **off** :

```
si(msg.payload == "on")
  out = 1;
autrement
if(msg.payload == "off")
  out = 0;
msg.payload = out; return msg;
```

- Créer un **rpi gpio out** et le nommer comme LED. Définissez la **Pin** à GPIO 2, **Typesortie** numérique, et Initialiser l'état de la broche à la **logique 0**.
- Rejoindre tous les nœuds comme dans la figure 15.3 et cliquer **Déployer**.

Pour tester le projet, demandez à Alexa "**Alexa, découvrir les appareils**". Après une minute ou deux, Alexa confirmera qu'elle a détecté votre appareil (Bedroom Lights dans ce projet).

Pour allumer la DEL, demandez à Alexa "**Alexa, allumez les lumières de la chambre**"

Pour éteindre la LED, demandez à Alexa "**Alexa, éteins les lumières de la chambre**"

Les nœuds peuvent être utilisés pour allumer/éteindre un appareil ou pour réduire la luminosité et sont pris en charge par le 1st, 2ⁿ^d et 3rd génération de dispositifs Alexa. Notez que nous pouvons **dispositifs d'écho Amazon** comme nous le souhaitons à notre conception pour le contrôle de divers équipements. Un exemple de projet est donné ci-dessous où nous avons ajouté un buzzer à notre conception.

15 3 Projet 74 – Contrôle d'une LED et d'un buzzer à l'aide de l'Alexa

Description : Dans ce projet, une LED et un buzzer sont connectés aux ports GPIO du Raspberry Pi. La LED et le buzzer sont allumés et éteints en donnant des commandes vocales à Alexa.

Viser : Le but de ce projet est de montrer comment plusieurs appareils peuvent être contrôlés par Alexa.

Schéma de principe : Le schéma du projet est présenté à la figure 15.4.

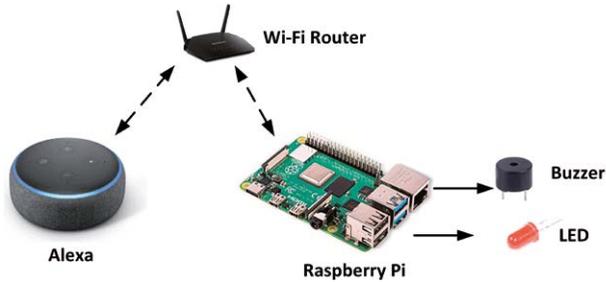


Figure 15.4 Schéma du projet

Schéma de circuit : Le schéma de circuit du projet est montré dans la figure 15.5, où la LED est connectée à GPIO 2 et le buzzer est connecté à GPIO 3.

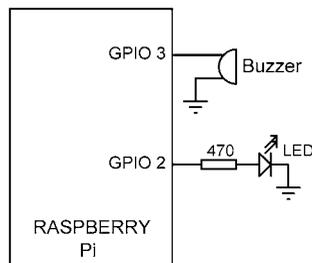


Figure 15.5 Schéma du projet

Programme de flux Node-RED : La figure 15.6 montre le programme de flux. Notez que nous avons ajouté un autre **dispositif d'écho Amazon** nœud de notre programme et le nommez comme **Buzzer**. Ce deuxième nœud entraîne un autre nœud de fonction qui a exactement le même contenu que celui dans la figure 15.3. Le second **rpi gpio out** nœud est utilisé pour conduire le buzzer où son **Pin** est réglé sur GPIO 3 et son **Type** est la sortie numérique comme avant. Parce que le nouveau dispositif d'écho amazon n'est pas connu d'Alexa, nous devons demander à Alexa de découvrir les nœuds à nouveau : "**Alexa, découvrez les appareils**". Après cela, nous pouvons contrôler la LED et le buzzer en demandant à l'Alexa :

Pour allumer la DEL, demandez à Alexa "**Alexa, allumez les lumières de la chambre**"

Pour éteindre la LED, demandez à Alexa "**Alexa, éteins les lumières de la chambre**"

Pour activer le buzzer, demandez à Alexa "**Alexa, allumez le buzzer**"

Pour désactiver le buzzer, demandez à Alexa "**Alexa, éteins le buzzer**"

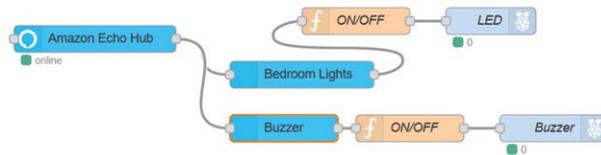


Figure 15.6 Programme de flux du projet

Notez que nous aurions pu utiliser un autre nœud Alexa appelé **node-red-contrib-alexa-home-skill** pour des tâches de contrôle plus sophistiquées basées sur Alexa. Ce nœud, appelé le nœud Alexa Home, nécessite que l'utilisateur soit enregistré et ait un compte avant de l'utiliser. Le nœud d'Alexa Home supporte les commandes suivantes (Les lecteurs intéressés peuvent obtenir beaucoup plus d'informations à partir du lien web : <https://alexa-node-red.bm.hardill.me.uk/docs>) :

- TurnOnRequest
- TurnOffRequest
- SetPercentageRequest
- Demande de majorationPercentageRequest
- DecrementPercentageRequest
- SetTargetTemperatureRequest
- IncrementTargetTemperatureRequest
- DecrementTargetTemperatureRequest
- GetTemperatureReadingRequest
- GetTargetTemperatureRequest
- SetLockState
- GetLockState
- SetColorRequest
- SetColorTemperatureRequest

Programme modifié

Dans le projet précédent, la LED est allumée et éteinte en envoyant des commandes vocales à Alexa. Dans certaines applications, nous pouvons vouloir allumer la LED en détectant une commande, puis nous pouvons vouloir que la LED s'éteigne automatiquement après un certain temps. Par exemple, dans une pièce, nous pouvons demander à Alexa d'allumer la lumière et après 20 secondes, nous pouvons vouloir que la lumière s'éteigne automatiquement. Cela peut être fait facilement en ajoutant un **déclencheur** nœud de la Figure 15.3. Dans l'exemple de programme d'écoulement illustré à la Figure 15.7, le voyant DEL est éteint après 15 secondes. Le nœud de déclenchement est configuré pour envoyer 1 pendant 15 secondes et ensuite il envoie 0. La figure 15.8 montre le **déclencheur** contenu du nœud. Notez que la sortie de l' **déclencheur** nœud est réinitialisé si msg.payload est égal à 0 (c.-à-d. si Alexa est invité à éteindre les lumières de la chambre).

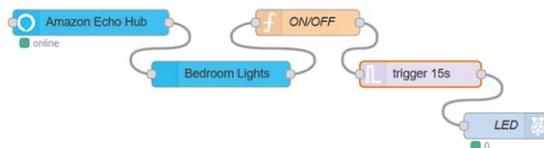


Figure 15.7 Programme de débit modifié

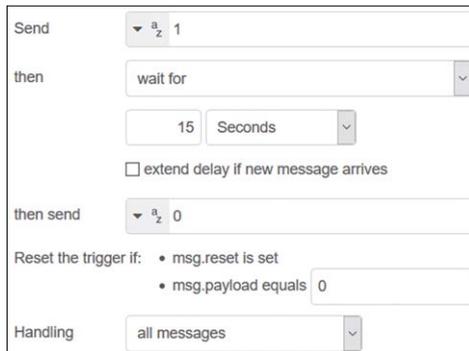


Figure 15.8 Configuration du nœud de déclenchement

15 . 4 Résumé

Dans ce chapitre, nous avons appris comment utiliser Amazon Alexa avec Node-RED pour contrôler l'équipement connecté à un Raspberry Pi. Bien que les projets de ce chapitre utilisent le Rasp- Pi, il n'y a aucune raison pour laquelle Arduino ou le DevKitC ESP32 ne peuvent pas être utilisés dans les projets basés sur Alexa. Comme décrit dans les chapitres précédents, nous pouvons connecter l'Arduino ou le ESP32 DevKitC au Raspberry Pi et ensuite contrôler leurs ports GPIO via le Raspberry Pi en envoyant des commandes via Alexa.

Dans le prochain chapitre, nous allons apprendre comment accéder à Node-RED sur Raspberry Pi de n'importe où sur terre.

Chapitre 16 • Accès au Raspberry Pi Node-RED depuis n'importe où

16 . 1 Aperçu

Dans tous les projets précédents, nous avons accédé au Raspberry Pi Node-RED via une liaison Wi-Fi à travers notre routeur Wi-Fi. Il existe de nombreux cas où nous pouvons vouloir accéder à Node-RED depuis n'importe où sur terre, afin que par exemple les appareils dans notre maison puissent être contrôlés et surveillés à distance depuis n'importe où sur terre. Dans ce chapitre, nous allons apprendre comment cela peut être fait.

16 . 2 Le ngrok

Nous utiliserons un service gratuit appelé **ngrok** afin de créer un tunnel vers notre Raspberry Pi pour que nous puissions y accéder depuis n'importe où dans le monde. Tout d'abord, nous devons changer notre numéro de port par défaut de 80 à quelque chose d'autre, p.ex. 1880 car pour une raison quelconque ngrok ne fonctionne pas avec le numéro de port 80.

Les étapes à suivre pour installer et utiliser **ngrok** sont les suivants :

- Aller au site web : <https://ngrok.com>
- Entrez vos coordonnées comme indiqué à la figure 16.1 pour créer un compte.

The image shows a registration form titled "Join ngrok". It contains the following elements from top to bottom:

- A heading "Join ngrok" in orange.
- A label "Your Name" above a text input field containing "Full Name".
- A label "Your Email" above a text input field containing "Email address".
- A label "Confirm Email" above a text input field containing "Email address".
- A label "Password" above a text input field containing "Password".
- A checkbox labeled "I'm not a robot" next to a reCAPTCHA logo and the text "reCAPTCHA Privacy - Terms".
- An orange button labeled "Create an account".

Figure 16.1 Entrez vos coordonnées

- Cliquez **Auth** lien à gauche pour obtenir votre jeton unique de tunnel comme indiqué dans la Figure 16.2. Dans cet exemple, le jeton de tunnel de l'auteur est (ceci est modifié pour des raisons de sécurité) : **gqudoMhDuJeXnLqVGHqv7jyUTtbYpnQzTk3Eb**

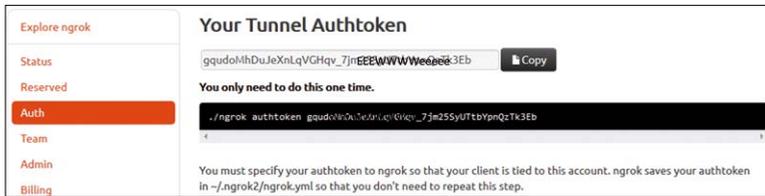


Figure 16.2 Jeton de tunnel

- Sélectionnez le **Linux (ARM)** version du logiciel
- Copiez le logiciel dans un dossier de votre Raspberry Pi (par ex. **/home/pi**). Vous pouvez, si vous le souhaitez, télécharger le logiciel sur votre PC puis utiliser l'utilitaire WinSCP pour le copier sur votre Raspberry Pi. Au moment de la rédaction du présent ouvrage, le logiciel portait le nom suivant : **ngrok-stable-linux-arm. zip**
- Décompressez le logiciel sur votre Raspberry Pi et enregistrez le jeton dans un fichier **ngrok . yml** par entrer la commande indiquée à la figure 16.3

```
pi@raspberrypi:~$ unzip ngrok-stable-linux-arm.zip
Archive:  ngrok-stable-linux-arm.zip
  inflating: ngrok
pi@raspberrypi:~$ ./ngrok authtoken gqudMhDuJeXnLqVGHqv_7JmEEeWWWWWee6e3Eb
Authtoken saved to configuration file: /home/pi/.ngrok2/ngrok.yml
pi@raspberrypi:~$
```

Figure 16.3 Décompresser le logiciel

- Créer un tunnel sécurisé avec le numéro de port 1880 :

```
pi@raspberrypi:~$ ./ngrok http 1880
```

- Vous verrez un écran similaire à celui de la figure 16.4. Ici, nous voulons la partie qui commence par **Envoi** et **https**, car c'est l'adresse que nous allons entrer dans notre navigateur web, i.e.

<https://2353edf8.ngrok.io>

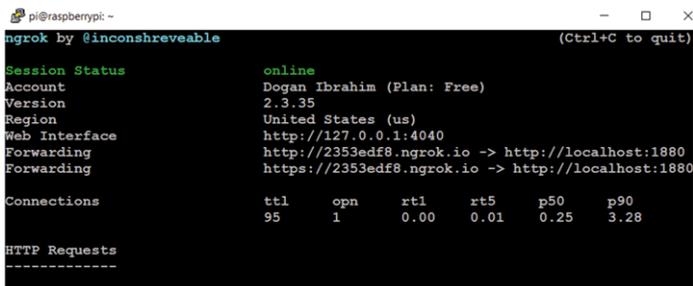


Illustration 16.4 Écran du Raspberry Pi

- Démarrez Node-RED sur votre Raspberry Pi manuellement (ou automatiquement pendant la re- Temps de démarrage. Ceci est indiqué dans la section suivante de ce chapitre)

- Vous pouvez maintenant accéder à Node-RED à distance en saisissant l'adresse suivante dans votre navigateur web. Notez que vous ne devez pas fermer l'écran ngrok (Figure 16.4) lorsque vous accédez à distance à Node-RED :

<https://2353edf8.ngrok.io>

16 . 3 Démarrage automatique du nœud-RED au redémarrage

La commande suivante peut être donnée pour démarrer automatiquement Node-RED après le démarrage de votre Raspberry Pi :

```
pi@framberrypi :~$ sudo systemctl enable nodered.service
```

De même, pour désactiver le démarrage automatique de Node-RED au redémarrage, saisir :

```
pi@framberrypi :~$ sudo systemctl disable nodered.service
```

Vous pouvez saisir la commande suivante pour vérifier si Node-RED est en cours d'exécution :

```
pi@framberrypi :~$ ps axg | grep noeud-rouge
```

16 . 4 Résumé

Dans ce chapitre, nous avons appris comment accéder à Node-RED depuis n'importe où sur Terre.

Dans le prochain chapitre, nous allons voir comment la communication Bluetooth peut être utilisée avec des projets basés sur Node-RED sur Raspberry Pi.

Chapitre 17 • Utilisation de la technologie Bluetooth avec Node-RED

17 . 1 Aperçu

Bluetooth est une technologie de communication sans fil utilisée sur de courtes distances. Bien qu'il ait été développé à l'origine comme une alternative sans fil à la communication série RS232, il est largement utilisé aujourd'hui pour transférer des données entre deux appareils compatibles Bluetooth. La plupart des téléphones mobiles intelligents sont dotés d'un matériel Bluetooth intégré et peuvent être utilisés pour transférer des données, p. ex., des images vers d'autres PC et vers d'autres téléphones portables compatibles. Bluetooth fonctionne dans la gamme de fréquences de 2,4 à 2,485 GHz et est géré par le groupe d'intérêt spécial Bluetooth.

Dans ce chapitre, nous allons apprendre à utiliser le Bluetooth avec Node-RED sur un Raspberry Pi. Avant d'utiliser le Bluetooth avec Node-RED, nous devons l'installer dans la palette de noeuds. Les étapes sont :

- Démarrer le noeud-RED
- Cliquez **Menu -> Gérer la palette** et cliquez **Installer**
- Entrer **node-red-contrib-bluetooth-serial-port** et cliquez **installer**
- Vous devriez voir deux nouveaux nœuds nommés **bt en série** et **bt série sortie** ajouté à votre palette de noeuds

Il faut maintenant installer le Bluetooth sur notre Raspberry Pi :

```
pi@framberrypi :~ $ sudo apt-get install bluez
```

Un exemple de projet est donné dans la section suivante pour montrer comment le noeud Bluetooth peut être utilisé.

17 . 2 Projet 75 – Contrôle d'une LED et d'un buzzer par Bluetooth

Description : Dans ce projet, une LED et un buzzer sont connectés aux ports GPIO du Raspberry Pi. La LED et le buzzer sont allumés et éteints en envoyant des commandes à partir d'un téléphone mobile intelligent via l'interface Bluetooth.

Viser : Le but de ce projet est de montrer comment Bluetooth peut être utilisé avec Node-RED sur un Raspberry Pi.

Schéma de principe : Le schéma du projet est présenté à la figure 17.1.



Figure 17.1 Schéma du projet

Schéma de circuit : Le schéma de circuit du projet est comme indiqué dans la figure 15.5, où la LED est connectée à GPIO 2 et le buzzer est connecté à GPIO 3 de Raspberry Pi.

Programme de flux Node-RED : La figure 17.2 montre le programme de flux du projet qui se compose- Système de 4 nœuds : a **bt en série** pour recevoir des données du périphérique Bluetooth, un **fonction** nœud pour formater les données des ports GPIO, et deux **rpi gpio out** nœuds pour contrôler la LED et le buzzer.

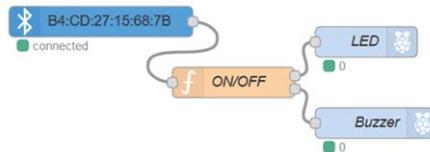


Figure 17.2 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **bt en série** node. Dans ce projet, nous utilisons un téléphone mobile Android. Nous devons trouver l'adresse MAC Bluetooth de notre téléphone mobile et l'entrer dans le **bt en série** nœud. L'adresse MAC du téléphone mobile Android peut être trouvée comme suit :
- Activer le Bluetooth sur votre téléphone mobile Android
- Cliquez **Paramètres**
- Cliquez **Système**
- Cliquez **À propos du téléphone**
- Cliquez **Statut**
- Faites défiler la liste jusqu'à ce que vous voyiez l'adresse Bluetooth de votre téléphone mobile Android (voir la figure 17.4). L'adresse MAC du téléphone mobile dans ce projet était B4:CD:27:15:68:7B

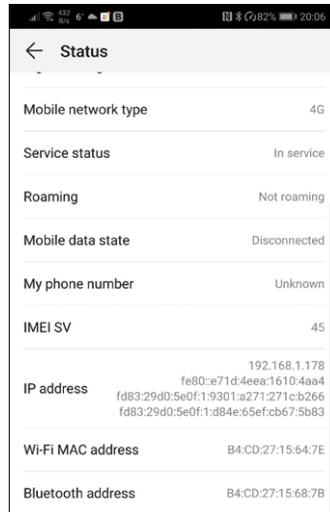


Figure 17.3 Adresse MAC du téléphone mobile Android

La figure 17.4 montre comment les **bt en série** node doit être configuré.



Illustration 17.4 Configurer le nœud bt serial dans

- Créer un **fonction** nœud nommé **ON/OFF** et entrez les données suivantes dans ce nœud. Les données retournées par le **bt en série** node est en binaire et il est converti en chaîne et stocké dans la variable **T**. Si la commande est LON, la sortie 1 est définie sur 1, si la commande est LOFF, la sortie 1 est définie sur 0. De même, la sortie 2 est définie sur 1 et 0 si les commandes sont respectivement BON et BOFF :

```

var out1 = null;
var out2 = null;
var T = msg.payload.toString(); if(T ==
"LON")
    out1 = {payload : 1};
sinon si(T == "LOFF")
    out1 = {payload : 0};

si(T == "BON")
    out2 = {payload : 1};
sinon si(T == "BOFF")

```

```
out2 = {payload : 0};
return[out1, out2];
```

- Créer deux **rpi gpio out** nœuds nommés **LED** et **Buzzer**. Régler le **Pin** à GPIO 2 pour la LED et à GPIO 3 pour le buzzer. Régler **Type** à la sortie numérique, et **Initialiser** la broche à 0 pour les deux nœuds.
- Rejoindre tous les nœuds comme dans la figure 17.3, et cliquer **Déployer**

Nous pouvons envoyer des données à notre Raspberry Pi en utilisant une application Bluetooth sur notre téléphone mobile Android. Il existe de nombreuses applications Bluetooth dans le Play Store. Celui utilisé dans ce projet s'appelle **Terminal Bluetooth par Qwerty** comme indiqué dans la figure 17.5. Installez cette application sur votre téléphone mobile.

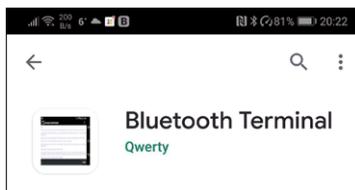


Illustration 17.5 Applications du terminal Bluetooth

Pour pouvoir accéder au Raspberry Pi à partir d'une application de téléphone mobile, effectuez les modifications suivantes sur votre Raspberry Pi depuis la ligne de commande :

- Démarrer nano pour modifier le fichier suivant :

```
pi@framberrypi :~ $ sudo nano /etc/systemd/system/dbus-org.bluez.service
```

- Ajouter **-C** à la fin de la ligne **ExecStart=**. Ajoutez également une autre ligne après l'**Exec-** Ligne de départ. Les deux dernières lignes devraient ressembler à ceci :

```
ExecStart=/usr/lib/bluetooth/bluetoothd -C
ExecStartPost=/usr/bin/sdptool add SP
```

- Quitter et enregistrer le fichier en appuyant sur **Ctrl+X** et **Y**
- Redémarrer le Raspberry Pi 3 :

```
pi@framberrypi :~ $ redémarrage sudo
```

Avant d'envoyer une commande à notre Raspberry Pi nous devons connaître l'annonce MAC Bluetooth- Description de notre Raspberry Pi. Vous pouvez le trouver en saisissant les éléments suivants dans le mode commande (votre nom d'appareil sera probablement différent) :

```
pi@framberrypi :~ $ bluetoothctl
[MyDevice]# montrer
```

Vous trouverez l'adresse MAC Bluetooth en haut de la liste. Saisissez les commandes suivantes pour permettre la détection du Bluetooth sur votre Raspberry Pi :

```
[MyDevice]# agent sur [MyDevice]#
découvrable sur [MyDevice]# sortie
```

Maintenant, assurez-vous que Node-RED est en cours d'exécution. Démarrez les applications Bluetooth sur votre téléphone mobile. Et cliquez sur les 3 points dans le coin supérieur droit. Cliquez **Rendre la découverte possible**, puis cliquez **Con- Besoin d'un appareil – Peu sûr**, sélectionnez le Raspberry Pi, et attendez que le message **connecter- e : framboises** est affiché. Vous verrez également que le message **connecté** est affiché sous le nœud **bt en série** votre programme de flux. Vous êtes maintenant prêt à envoyer des commandes depuis votre téléphone mobile vers le Raspberry Pi via le lien Bluetooth. Envoyez les com- Instructions (voir figure 17.6) :

LON	pour allumer la LED
LOFF	pour éteindre la LED
Bon	pour allumer le buzzer
BOFF	pour éteindre le buzzer



Figure 17.6 Envoyer des commandes pour contrôler la LED

Dans ce projet, nous avons reçu des commandes du téléphone mobile via une liaison Bluetooth. Vous pouvez essayer d'utiliser le **bt série sortie** Pour envoyer des commandes depuis le Raspberry Pi vers le téléphone mobile en utilisant la liaison Bluetooth.

17 . 3 Résumé

Dans ce chapitre, nous avons appris comment communiquer en utilisant des nœuds Bluetooth Node-RED. Nous avons développé un projet où une LED connectée à la Raspberry Pi peut être contrôlée en envoyant des commandes depuis un téléphone mobile via le lien Bluetooth. Il est possible d'utiliser une carte de développement Arduino ou un ESP32 DevKitC avec un Raspberry Pi pour

pour contrôler les appareils connectés à Arduino ou au DevKitC ESP32 en utilisant la connectivité Bluetooth avec Node-RED.

Dans le prochain chapitre, nous allons examiner l'important sujet du MQTT.

Chapitre 18 • Node-RED et MQTT

18 . 1 Aperçu

MQTT signifie **M**essage **Q**ueuing **T**elemétrie **T**ransport. C'est un protocole de message qui a été créé pour la communication machine-à-machine (M2M), et il est basé sur la publication et l'abonnement. MQTT est particulièrement utile pour l'envoi de données et le contrôle d'informations avec une faible bande passante et des temps de réponse élevés. Ce protocole est particulièrement intéressant lorsqu'il est nécessaire d'envoyer des données aux actionneurs et de récupérer des données à partir de capteurs. MQTT a été développé pour la première fois par IBM en 1999, principalement pour les communications par satellite, et est depuis devenu le protocole de communication standard pour les applications Internet des objets (IdO). MQTT utilise votre réseau domestique existant pour envoyer des messages à vos appareils de l'IdO et recevoir une réponse de leur part. Le site officiel de MQTT est : <http://mqtt.org>

MQTT fonctionne sur le protocole TCP/IP et est plus rapide que l'envoi de requêtes HTTP car un message peut être aussi petit que 2 octets et il n'y a pas d'en-tête, ce qui est le cas avec le HTTP. En outre, dans MQTT, les messages sont distribués automatiquement aux clients intéressés.

18 . 2 Fonctionnement du MQTT

Il est intéressant d'apprendre comment fonctionne MQTT avant de l'utiliser dans des projets. Dans le protocole MQTT, il y a un expéditeur appelé **Éditeur**, et le receveur appelé **Abonné**. Entre ces deux, un serveur appelé **Courtier** sert d'intermédiaire. La figure 18.1 montre la structure de base d'un système basé sur le MQTT.

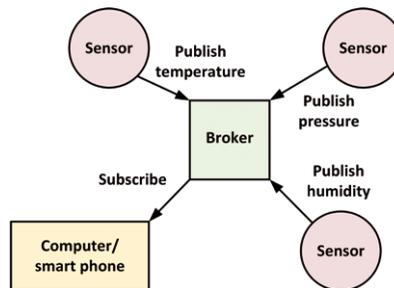


Figure 18.1 Structure de base du MQTT

Dans la figure 18.1, le capteur de température publie (ou envoie) sa valeur mesurée au bro- Qui accepte et enregistre les données. Le courtier envoie ensuite cette valeur mesurée à d'autres appareils ayant souscrit pour recevoir ces données.

Dans MQTT, il y a 5 concepts de base que vous devriez comprendre :

- Éditeur
- Abonné
- Messages
- Sujets
- Courtier

Éditeur : L'éditeur est le nœud qui envoie les données ou le message sur un sujet. Pour examen-simple, un éditeur peut être un nœud de capteur qui envoie des valeurs de température ambiante.

Abonné : L'abonné peut être un ordinateur, un smartphone, un microcontrôleur ou tout autre processeur qui s'inscrit pour recevoir les données ou le message. Par exemple, un appareil public sur un sujet, un autre appareil s'abonne au même sujet et reçoit le pub- message envoyé.

Messages : Les messages sont l'information échangée dans le réseau MQTT. Un message peut être des données ou une commande. Par exemple, un message peut être la valeur de température ou une commande pour activer un commutateur.

Sujets : Les sujets sont des concepts importants de MQTT. Voici comment nous enregistrons l'intérêt pour les messages entrants, ou comment nous spécifions où nous voulons publier le message. . l'échange de messages se fait par les rubriques. Les rubriques dans MQTT sont représentées par char- Chaînes de caractères, séparées par une barre oblique. Chaque barre oblique indique un niveau de sujet. Un exemple est montré ci-dessous qui crée un sujet pour une LED dans votre cuisine :

maison/cuisine/led

Dans l'exemple ci-dessus, les niveaux de sujet sont la maison, la cuisine et le led, séparés par des séparateurs de niveau de sujet. Les sujets sont sensibles à la casse et par exemple, Home/kitchen/led n'est pas le même que ci-dessus.

Wildcards à un seul niveau : Nous pouvons utiliser des caractères génériques dans les sujets de sorte que plusieurs capteurs peuvent être interrogés en même temps. Un "+" est utilisé pour identifier un caractère générique de niveau unique. Un exemple est donné ci-dessous :

home+/led	sujet avec wildcard de niveau singe
home/bedroom/led	remplacé par wildcard de chambre à
home/livingroom/led	coucher remplacé par wildcard de
home/garden/led	salon remplacé par garden

Caractères génériques à plusieurs niveaux : On peut aussi utiliser un caractère générique multi-niveaux, représenté par le signe « # ». Il doit toujours être utilisé à la fin d'une rubrique. Un exemple est donné ci-dessous :

maison/cuisine/#	thème avec wildcard multi-niveaux
maison/chambre à	cuisine remplacée par chambre à
coucher/maison/salon/lumière	coucher et lumière remplacée

Courtier : le courtier reçoit tous les messages, les filtre et décide qui est intéressé par eux, puis envoie le message à tous les clients abonnés

La figure 18.2 illustre le fonctionnement du MQTT, qui peut être résumé comme suit :

- Le processeur (et le capteur) publie les messages ON et OFF sur le sujet **maison / cuisine/led**
- Nous avons un Raspberry Pi qui contrôle une LED. Le Raspberry Pi est abonné au sujet : **/maison/cuisine/led**
- Lorsqu'un nouveau message est publié sur le sujet **maison/cuisine/led**, le Raspberry Pi reçoit les messages ON ou OFF et allume la LED ou OFF.

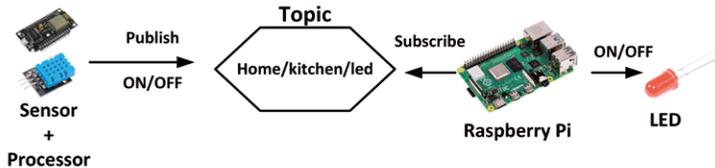


Figure 18.2 Fonctionnement du MQTT

18.3 Le courtier de Mosquitto

Le courtier est l'une des parties importantes de MQTT. Il y a plusieurs courtiers qu'on peut utiliser. Dans la plupart des projets de domotique, le **Mosquitto** Le broker est utilisé. Les étapes pour installer ce broker sur le Raspberry Pi sont les suivantes :

```

pi@framberrypi:~$ sudo apt-get update
pi@framberrypi:~$ sudo apt-get install mosquitto mosquitto-clients
  
```

Nous pouvons faire en sorte que Mosquitto démarre automatiquement au démarrage en saisissant la commande suivante :

```

pi@framboise:~$ sudo systemctl enable mosquitto.service
  
```

Pour tester l'installation de Mosquitto, entrez la commande **mosquitto -v** comme indiqué à la figure 18.3. Cette commande affiche la version du Mosquitto en cours d'exécution sur votre système (ne vous inquiétez pas de l'erreur : adresse déjà utilisée).

```

pi@raspberrypi:~$ mosquitto -v
1577732396: mosquitto version 1.5.7 starting
1577732396: Using default config.
1577732396: Opening ipv4 listen socket on port 1883.
1577732396: Error: Address already in use
pi@raspberrypi:~$ █
  
```

Figure 18.3 Essai de l'installation de Mosquitto

Nous pouvons vérifier si notre courtier fonctionne correctement comme suit :

Abonnez-vous au sujet « TestTopic » en saisissant la commande suivante. Mosquitto_sub indique que nous voulons souscrire à un sujet, et le nom suivant **-t** est le nom du sujet. Maintenant, chaque fois que nous publions (c.-à-d. envoyons un message) à TestTopic, le message apparaît dans la fenêtre :

```
pi@framberrypi: ~ $ mosquitto_sub -t "TestTopic"
```

Maintenant, parce que le terminal est à l'écoute des messages de notre courtier, nous devons ouvrir une autre fenêtre pour que nous puissions publier des messages. Après avoir ouvert une autre fenêtre, entrez la commande suivante pour publier sur TestTopic :

```
pi@framberrypi: ~ $ mosquitto_pub -t "TestTopic" -m "Hello There!"
```

Ici, le nom du sujet est après `-t` et le message est après `-m`. Après avoir appuyé sur la touche Entrée, nous verrons le message apparaître sur notre terminal d'abonné comme indiqué dans la figure 18.4.

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ mosquitto_sub -t "TestTopic"
Hello There!

pi@raspberrypi:~ $ mosquitto_pub -t "TestTopic" -m "Hello There!"
pi@raspberrypi:~ $
```

Figure 18.4 Exemple de courtier

18.4 Utilisation de MQTT dans la domotique et les projets IoT

Pour utiliser le MQTT dans la domotique et les projets IoT, nous avons besoin des éléments suivants :

- Un Raspberry Pi
- Node-RED et MQTT
- Un Arduino, ESP32, ESP8266, ou tout autre microcontrôleur compatible

La figure 18.5 montre la configuration de base du système MQTT. Les capteurs, actionneurs, relais, interrupteurs, lampes, LED, etc sont connectés au système via un Arduino, ESP32, ESP8266 ou tout autre processeur compatible.

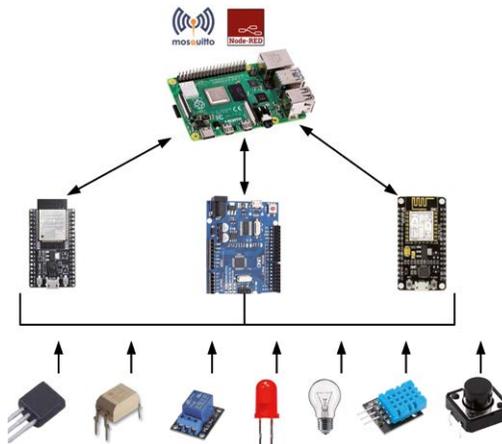


Figure 18.5 Configuration du système MQTT

Node-RED offre deux nœuds MQTT (**mqtt en** et **En dehors de la**) qui se trouve dans le filet- La palette de travail comme indiqué à la figure 18.6.

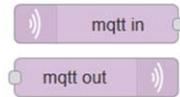


Figure 18.6 Nœuds MQTT

Nous pouvons envoyer un message au courtier MQTT en utilisant une **injecter** nœud comme indiqué à la figure 18.7. Notez que la petite boîte verte sous le **En dehors** nœud indique que la connexion depuis **En dehors** le nœud du courtier a été établie. Dans la figure 18.7, le sujet est défini sur **TestTopic**, et le **injecter** nœud **Charge utile** est défini sur message **Bonjour de ma part!**. Ce message s'affiche sur l'abonné MQTT comme indiqué dans la figure. Dans ce programme de flux, le **En dehors** nœud **Serveur** était réglé pour **localhost** et la **Port** numéro a été réglé sur 1883 (les ports réseau 1883 et 8883 sont réservés par défaut) :

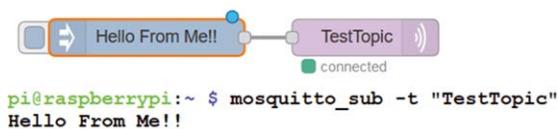


Figure 18.7 Envoi d'un message au courtier MQTT

Notez qu'en utilisant différents sujets, nous pouvons envoyer (publier) des messages différents et ces messages seront reçus par les abonnés ayant les mêmes sujets.

Un exemple très simple est donné ci-dessous qui illustre comment une LED peut être commandée en utilisant MQTT

18 . 5 Projet 76 – Contrôle d'une DEL au moyen de la technologie MQTT

Description : Dans ce projet, nous allons apprendre comment une LED peut être contrôlée en utilisant MQTT.

Viser : Le but de ce projet est de montrer comment MQTT peut être utilisé dans un projet très simple.

Schéma de circuit : Dans ce projet, une LED est connectée à la broche de port GPIO 2 du Raspberry Pi par l'intermédiaire d'une résistance de limitation de courant.

Programme de flux Node-RED : La figure 18.8 montre le programme de flux du projet. Fondamentalement, deux **injecter** les nœuds sont utilisés avec les Charges utiles réglées respectivement à 1 et 0. Lorsque par exemple le bouton de la **injecter** node est cliqué puis un 1 est envoyé au broker via le **En dehors** nœud. Le **mqtt en** le nœud dans le flux inférieur est donné le même nom de sujet que le **En dehors** nœud et ce nœud entraîne le **rpi gpio out** nœud qui contrôle la LED en conséquence. La console Raspberry Pi affiche les données reçues par le port GPIO puisqu'elle est abonnée au même sujet que celui indiqué dans la figure 18.9.

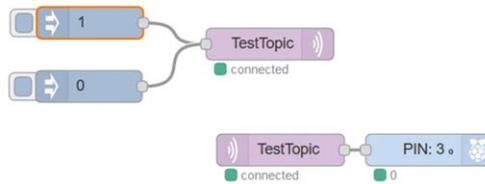


Figure 18.8 Programme de flux du projet

```
pi@raspberrypi:~ $ mosquitto_sub -t "TestTopic"
1
0
1
0
```

Figure 18.9 La console Raspberry Pi a souscrit au même sujet

Dans les projets plus complexes et réels, au lieu d'utiliser des nœuds injectés, nous avons généralement des processus- Les capteurs sont connectés à leurs entrées. Les données de ces capteurs sont envoyées au courtier via le **En dehors** nœud. Les nœuds mqtt reçoivent et traitent ensuite ces données (p. ex., afficher ou activer un actionneur).

Dans les sections suivantes, nous allons apprendre à utiliser le ESP8266 NodeMCU en tant que client dans une application MQTT.

18 . 6 Le processeur ESP8266

L'ESP8266 est une puce Wi-Fi à faible coût qui peut être utilisée avec ou sans microcontrôleur pour accéder à un réseau Wi-Fi. La puce a été développée par un chinois basé à Shanghai- fabriquer Espressif Systems et intègre la pile TCP/IP complète. Il peut être utilisé comme un petit microcontrôleur seul, ou il peut être interfacé avec un plus grand microcontrôleur développer- Système d'intégration tel que l'Arduino, qui peut être utilisé pour fournir une capacité Wi-Fi au système hôte.

Il existe de nombreuses versions de la puce ESP8266, disponibles sous forme de modules sur des petites cartes avec différentes fonctionnalités et différentes capacités d'entrée/sortie. Les spécifications de base de la puce ESP8266 sont résumées ci-dessous :

- CPU RISC 32 bits : Tensilica Xtensa LX106
- Vitesse de fonctionnement 80 à 160 MHz (par overclocking)
- Fonctionnement 3.3V (+2.5V min à +3.6V max)
- 64 Kbytes de mémoire de programme, 96 Kbytes de mémoire de données
- Mémoire flash externe QSPI (jusqu'à 16 Mo)
- Prise en charge du Wi-Fi IEEE 802.11 b/g/n
- Authentification WEP ou WPA/WPA2 (ou réseau ouvert)
- 1 convertisseur analogique-numérique (ADC) 10 bits
- Jusqu'à 16 GPIO (pas tolérant +5 V)
- Partagé SDIO, SPI, I²C, broches d'interface I2S
- Pile de protocoles TCP/IP intégrée
- UART sur les broches dédiées (TX supplémentaire uniquement UART sur la broche GPIO2)
- Puissance de secours < 10 mW

NodeMCU (LoLin) est une petite carte de développement de microcontrôleur basée sur le pro ESP8266-processor. Il a été fabriqué pour la première fois en 2014 et est actuellement l'un des kits de développement Wi-Fi les plus populaires basés sur la puce ESP8266 et il intègre un adaptateur USB-to-TTL embarqué, une prise micro USB, un grand nombre de GPIOs, PWM,²C, ADC, régu de tension +3,3 V- lator, UART et ainsi de suite. La figure 18.10 montre la configuration des broches du NodeMCU.

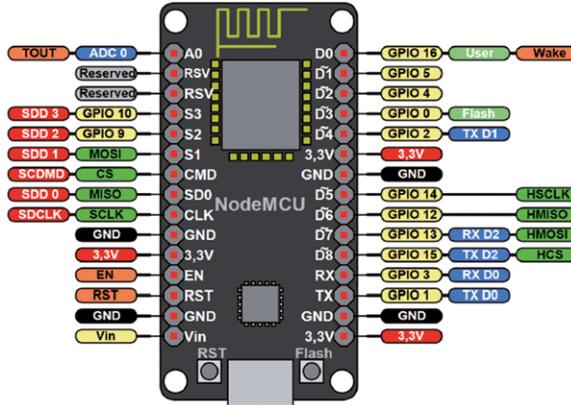


Figure 18.10 Configuration des broches du NodeMCU

NodeMCU peut être programmé à l'aide de l'IDE Arduino. Les étapes pour installer le logiciel nécessaire sur l'IDE Arduino sont indiquées ci-dessous :

- Démarrez votre IDE Arduino
- Cliquez **Fichier -> Préférences** et saisissez les informations suivantes au bas de l'écran, comme indiqué à la figure 18.11. Si vous avez déjà une autre URL, séparez-les avec une virgule et cliquez sur **D'accord**

http://arduino.esp8266.com/stable/package_esp8266com_index.json

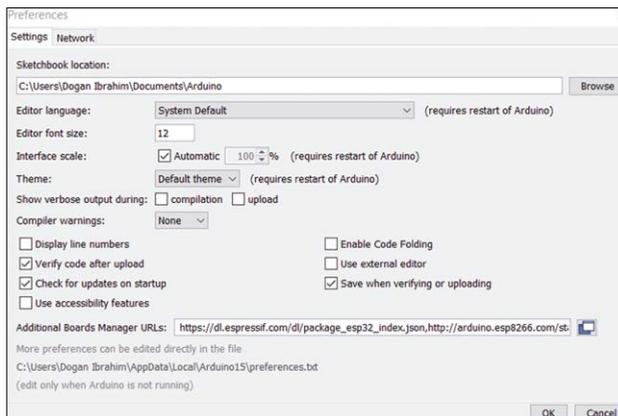


Figure 18.11 Installation de l'ESP8266 sur l'IDE Arduino

- Sélectionner **Outils -> Carte "Arduino/Genuino Uno" -> Gestionnaire de cartes**
- Entrer **esp8266** et appuyez sur le **installer** bouton pour **ESP8266 par ESP8266 Community** (voir figure 18.12)



Figure 18.12 Entrée esp8266

- Après une installation réussie, vous devriez voir le message **Installed** comme dans la figure 18.13. Fermer l'écran.

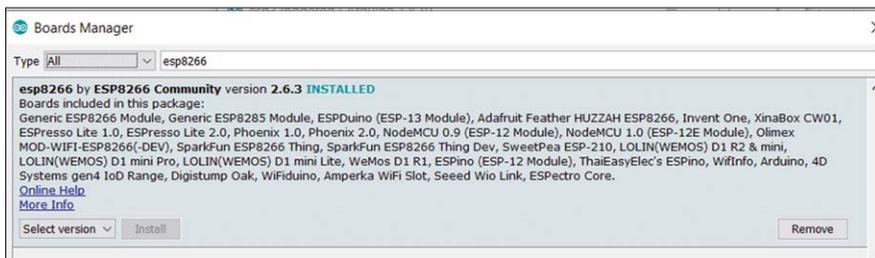


Figure 18.13 Installation réussie

La programmation détaillée du processeur ESP8266 dépasse le cadre de ce livre. Un projet simple est donné dans la section suivante juste pour rappeler aux utilisateurs comment l'ESP8266 peut être programmé en utilisant l'IDE d'Arduino. On suppose que les lecteurs sont familiarisés avec l'utilisation de l'IDE Arduino.

18 . 7 Projet 77 – Clignotement d'une LED à l'aide de la norme NodeMCU ESP8266

Description : Il s'agit d'un projet très simple où une LED est connectée à l'un des ports GPIO du NodeMCU et la LED clignote chaque seconde.

Viser : Le but de ce projet est de rappeler aux utilisateurs comment le NodeMCU peut être programmé en utilisant l'IDE Arduino.

Schéma de circuit : La figure 18.14 montre le schéma du projet. La LED est- raccordé au port 16 du GPIO de NodeMCU.

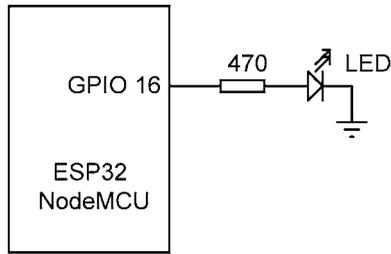


Figure 18.14 Schéma du projet

Liste des programmes : La liste des programmes (programme : **esp8266led**) est illustré à la figure 18.15. Les étapes pour commencer la programmation sont les suivantes :

Construire le circuit et connecter le NodeMCU à votre PC via un câble mini USB

- Démarrer l'EDI Arduino
- Cliquez **Outils -> Carte -> NodeMCU 1.0 (module ESP-12E)**
- Sélectionnez le port COM et cliquez sur **Outils -> Port n** où n est le numéro du port
- Entrez le programme et enregistrez-le avec le nom **esp8266led** :

```

DEL int = 16;
void setup() {

    pinMode (LED, OUTPUT);
}

boucle vide()
{
    digitalWrite(LED, HIGH);           // LED ON
    delay(1000); digitalWrite(LED,    // Attendre une seconde
    LOW); delay(1000);                // LED OFF
                                        // Attendez une seconde
}
    
```

Figure 18.15 programme esp8266led

- Cliquez **Sketch -> Vérifier/Compiler** et s'assurer qu'il n'y a pas d'erreurs de compilation
- Cliquez **Sketch -> Télécharger** et attendez que le message s'affiche **Téléversement terminé.**
- Vous devriez voir le LED clignoter à chaque seconde

La figure 18.16 montre le circuit construit sur une planche à pain.

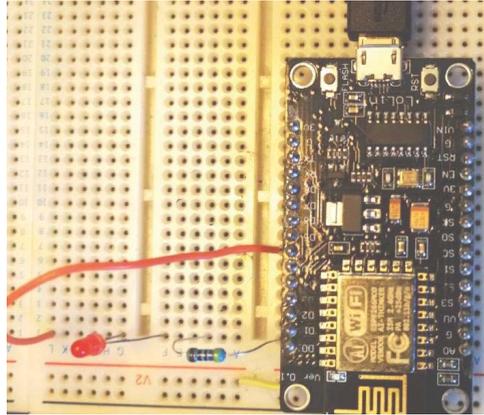


Figure 18.16 Le circuit construit sur une planche à pain

18 . 8 Utilisation de l'ESP8266 NodeMCU avec MQTT

Pour que MQTT fonctionne sur ESP8266, nous devons installer une bibliothèque appelée **PubSubClient**. Cette bibliothèque peut être téléchargée à partir du lien suivant :

<https://github.com/knolleary/pubsubclient/archive/master.zip>

Les étapes sont les suivantes :

- Télécharger le document ci-dessus **.zip** fichier (Cliquez **Cloner ou télécharger** et sélectionnez **Télécharger le fichier ZIP**). Le fichier s'appelle : **pubsubclient-maitre.zip**
- Démarrer l'EDI Arduino
- Cliquez **Sketch -> Inclure la bibliothèque -> Ajouter . ZIP Library**
- Sélectionnez le chemin d'accès à la **.zip** fichier
- Sélectionner **Fichier -> Exemples -> PubSubClient** et vous devriez voir un certain nombre de fichiers d'exemple basés sur mqtt listés

Vous devez faire en sorte que Mosquitto démarre automatiquement au démarrage en saisissant la commande suivante :

```
pi@framboise :~ $ sudo systemctl enable mosquitto.service
```

Un projet simple est donné ci-dessous pour illustrer comment MQTT peut être employé avec le ESP8266- deMCU.

18 . 9 Projet 78 – Contrôle d'une DEL à l'aide de la norme NodeMCU ESP8266 avec MQTT – LED connectée à Raspberry Pi

Description : Il s'agit d'un projet très simple où une LED est connectée à l'un des ports GPIO du Raspberry Pi. De plus, un interrupteur à bouton-poussoir est connecté au NodeMCU ESP8266. La LED est activée lorsque le bouton-poussoir est enfoncé.

Viser : Le but de ce projet est de montrer comment MQTT peut être utilisé avec l'ESP8266 NodeMCU. Ici, le NodeMCU est le client (éditeur) et le Raspberry Pi est l'abonné qui reçoit les messages.

Schéma fonctionnel : La figure 18.17 montre le schéma du projet.

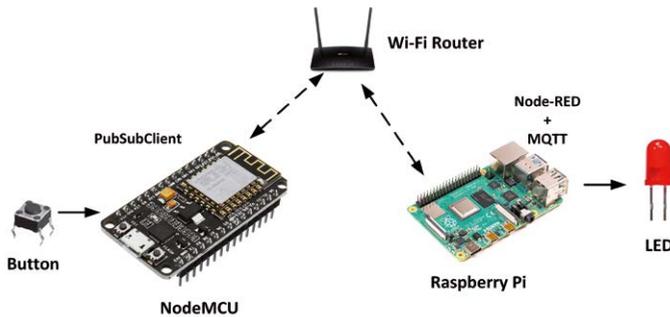


Figure 18.17 Schéma du projet

Schéma de circuit : Le schéma de circuit du projet est illustré à la figure 18.18. Le bouton est connecté au port GPIO 2 de l'ESP8266 NodeMCU. De même, la LED est connectée à la broche de port GPIO 2 du Raspberry Pi.

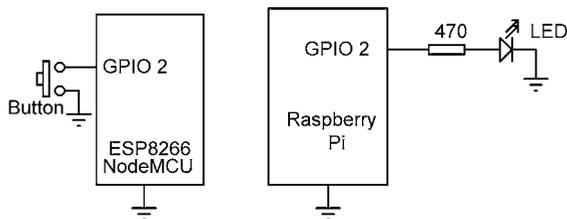


Figure 18.18 Schéma du projet

Liste du programme ESP8266 NodeMCU : La figure 18.19 montre la liste des programmes (programme : `mqttesp8266`). NodeMCU utilise la liaison Wi-Fi pour communiquer avec le Raspberry Pi dans les applications MQTT. Par conséquent, nous avons besoin de connaître le nom SSID et le mot de passe de notre routeur Wi-Fi, et l'adresse IP de notre Raspberry Pi. Dans cet exemple, nous avons les paramètres suivants (vous devrez entrer vos propres paramètres) :

SSID : BTHomeSpot-XNH
 Mot de passe : 49350baeb
 Adresse IP de la Raspberry Pi : 192.168.1.202

Nous devons saisir ces valeurs dans notre programme. Notez que le bouton est connecté à la broche de port GPIO 2 :

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = &quot;BTHomeSpot-XNH&quot;;
const char* mot de passe = &quot;49350baeb&quot;;
const char* mqtt_server = &quot;192.168.1.202&quot;;
const byte Button = 2;
const char *ID = &quot;Example_Button&quot;;
bool SwitchState = 0;
```

Dans cet exemple, nous avons choisi le sujet MQTT comme **cuisine/lumière**

```
const char *Sujet = &quot;kitchen/light&quot;;
```

Nous devons configurer un client Wi-Fi et un client PubSubClient :

```
Client WiFiClient ; Client PubSubClient
(wclient);
```

À l'intérieur de la routine de configuration, nous allons initialiser le moniteur série Arduino IDE à 115200 Baud afin que nous puissions voir facilement la trace de l'exécution du code. Nous devons également configurer la broche de port GPIO 2 à laquelle le commutateur est connecté en entrée et activer la résistance de traction, et activer le Wi-Fi sur notre NodeMCU ESP8266 :

```
//
// La routine de
// configuration //
void setup()
{
  Serial.begin(115200); pinMode(Button, INPUT);
  digitalWrite(Button, HIGH); delay(100);
  setup_wifi(); client.setServer(mqtt_server, 1883);
}
}
```

Le Wi-Fi est configuré dans la routine de configuration comme suit (les lecteurs qui ont utilisé l'ESP8266 sont- Vous devez donc connaître le code suivant :)

```

//
// Se connecter au routeur Wi-Fi local
//
void setup_wifi() {

    Serial.print(" nESP32 NodeMCU se connectant à : ");
    Serial.println(ssid);
    WiFi.begin (SSID, mot de passe);

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println(" nConnected to Wi-Fi"); }

```

La boucle principale du programme tente de se reconnecter au client en appelant la fonction **reconnecter** si la connexion est perdue, et lit l'état du bouton pour vérifier s'il est enfoncé.

Le code de reconnexion est :

```

//
// Reconnecter au client si la connexion est perdue //
void reconnect()
{
    while (!client.connected())
    {
        Serial.print("Tentative de connexion MQTT..."); si
        (client.connect(ID))
        {
            Serial.println("Connected");
            Serial.print("Publishing to : ");
            Serial.println(Topic); Serial.println(' n');
        }
        autrement
        {
            Serial.println(" Essayer de se reconnecter en 5 secondes");
            delay(5000);
        }
    }
}

```

Le code de boucle principal est :

```

boucle vide()
{
    si      (!client.connected())      reconnect();
    client.loop();

    if(digitalRead(Button) == 0) {      // Si le bouton est enfoncé

        ButtonState = ! ButtonState;      // Toggle ButtonState
        si(ButtonState == 1)      // ON
        {
            client.publish(Topic, "on"); Serial.println((String)Topic +
                " is ON");
        }
        autrement
        {
            client.publish(Topic, "off"); Serial.println((String)Topic + "
                est OFF");
        }

        while(digitalRead(Button) == 0) {      // Attendre le commutateur pour relâcher

            yield();
            délai(20);
        }
    }
}

```

Figure 18.19 *mqttesp8266* liste de programmes

Programme de flux Node-RED : La figure 18.20 montre le programme de flux pour le Raspberry Pi qui se compose de seulement 3 nœuds **mqtt en** noeud, un **fonction** nœud et un **rpi gpio out**noeud.

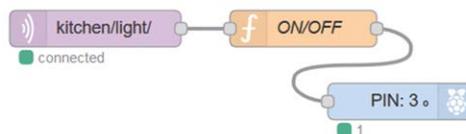


Figure 18.20 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **mqtt en** noeudet **définir son** Serveur à **l'hôte local**, Port à **1883** et le Sujet à **la cuisine/lumière/**

- Créer un **fonction** noms de nœud ON/OFF et entrez les instructions suivantes dans cette fonction :

```
si(msg.payload == "on")
  msg.payload = 1;
sinon si(msg.payload == "off")
  msg.payload = 0;
retourner le message;
```

- Créer un **rpi gpio out** nœud et définir **Pin** à GPIO 2, **Type** à la sortie numérique, et **Initialiser** à 0.
- Connecter tous les nœuds comme dans la figure 18.20 et cliquer **Déployer**.

Essais

Le projet peut être testé de la manière suivante :

- Mettre sous tension l'ESP8266 NodeMCU et ouvrir le moniteur série et régler sa vitesse de transmission sur 115200.
- Vous devriez voir des messages sur le moniteur série indiquant que le NodeMCU est connecté à votre Wi-Fi (voir la figure 18.21) et qu'il peut publier sur MQTT avec le thème cuisine/light/

```
ESP32 NodeMCU connecting to: BTHomeSpot-XNH
.....
Connected to Wi-Fi
Attempting MQTT connection...connected
Publishing to: kitchen/light/
```

Illustration 18.21 Messages de connexion sur le moniteur série

- Appuyer sur le bouton. Le message suivant doit s'afficher sur le moniteur série (voir la figure 18.22). En même temps, le voyant LED s'allume :

```
cuisine/lumière/ est allumée
```

- Relâchez le bouton et appuyez de nouveau. Vous devriez voir le message suivant- sur le moniteur série. En même temps, la LED s'éteint :

```
cuisine/lumière/ est OFF
```

```

ESP32 NodeMCU connecting to: BTHomeSpot-XNH
.....
Connected to Wi-Fi
Attempting MQTT connection...connected
Publishing to: kitchen/light/

kitchen/light/ is ON
kitchen/light/ is OFF
kitchen/light/ is ON
kitchen/light/ is OFF
kitchen/light/ is ON
kitchen/light/ is OFF
    
```

Illustration 18.22 Messages sur le moniteur série

- Si vous abonnez votre console Raspberry Pi à la rubrique cuisine/light/ alors vous devriez voir les messages affichés là aussi (voir Figure 18.23)

```

-
pi@raspberrypi:~ $ mosquitto_sub -t "kitchen/light/"
off
on
off
on
off
    
```

Illustration 18.23 Messages sur la console du Raspberry Pi

18 . 10 Project 79 – Contrôle d’une DEL à l’aide de NodeMCU ESP8266 avec MQTT – DEL connectée à NodeMCU ESP8266

Description : Ce projet est similaire au précédent mais ici la LED est connectée à la NodeMCU et le bouton est connecté à la Raspberry Pi. Comme précédemment, la LED est activée lorsque le bouton est enfoncé.

Viser : Le but de ce projet est de montrer comment le client (NodeMCU dans ce projet) peut être configuré comme un abonné afin qu’il puisse recevoir des messages en utilisant MQTT. Raspberry Pi est un éditeur dans ce projet.

Schéma fonctionnel : La figure 18.24 montre le schéma du projet.

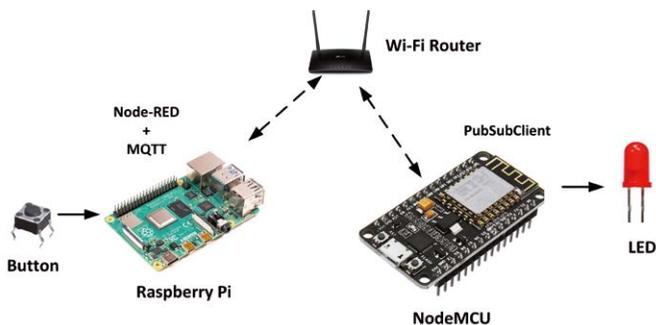


Figure 18.24 Schéma du projet

Schéma de circuit : Le schéma de circuit du projet est illustré à la figure 18.25. Le bouton est connecté au port pin GPIO 2 du Raspberry Pi. De même, la LED est connectée à la broche du port GPIO 2 du NodeMCU par l'intermédiaire d'une résistance de limitation de courant.

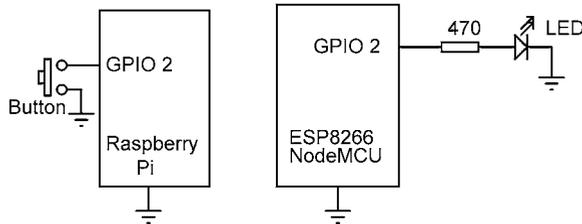


Figure 18.25 Schéma du projet

Liste du programme ESP8266 NodeMCU : La figure 18.26 montre la liste des programmes (programme : `esp8266mqtt`). Le programme est très semblable à celui présenté dans la figure 18.19, sauf les changements suivants :

L'instruction `client.subscribe(Topic)` est ajoutée à la routine de configuration afin que le NodeMCU soit configuré comme un abonné MQTT.

La boucle principale du programme a été modifiée de sorte que la boucle ne se reconnecte que si le- tion est perdue.

Une nouvelle fonction appelée callback est ajoutée au programme. Cette fonction est appelée chaque fois qu'un message est reçu par le NodeMCU. Au début de la fonction, le message Nouveau message arrivé s'affiche avec le nom du sujet MQTT. La fonction- lit le message reçu. Tableau de caractères `charge utile[0]` enregistre le premier caractère du message reçu qui est soit `'1'` ou `'0'` dans ce projet. Si `'1'` est reçu, la LED est allumée ; sinon, la LED est éteinte. Les contenus de cette fonction sont les suivants :

```
void callback (char *topic, byte* payload, longueur int non signée)
{
    Serial.print(« New message arrived - Topic:" »);
    Serial.println(Topic);
    pour(int j = 0 ; j < length; j++)
    {
        Serial.print((char)payload[j]);
    }
    Serial.println(); if((char)payload[0]
    == '1') {

        digitalWrite (LED, HIGH);
    }
    autrement
    {
        digitalWrite (LED, LOW);
    }
}
```

```

    }
}

/*****
 *          PROGRAMME MQTT NodeMCU ESP8266
 *          =====
 * Dans ce programme, un interrupteur à bouton-poussoir est connecté au port
 * GPIO 2 du Raspberry Pi. De même, une LED est connectée
 * vers le port GPIO 2 du NodeMCU. Appuyer sur le bouton permet de basculer
 * la LED.
 *
 * Auteur : Dogan Ibrahim
 * Fichier : esp8266mqtt
 * Date : décembre 2020
 *
 *****/ #include <ESP8266WiFi.h>

#include <PubSubClient.h>

const char* ssid = &quot;BTHomeSpot-XNH&quot;; // SSID Wi-Fi
const char* mot de passe = &quot;49345abaeb&quot;; // Mot de passe pour le réseau WiFi
const char* mqtt_server = &quot;192.168.1.202&quot;; // Raspberry Pi IP //
const byte LED = 2; // Broche LED
const char *ID = &quot;Example_LED&quot;;
const char *Sujet = &quot;kitchen/light&quot;; // SUJET DU PROGRAMME DE FINANCEMENT MULTISERVICES

Client WiFiClient
Client PubSubClient (wclient);

//
// Connectez-vous au routeur Wi-Fi local. Cette fonction
// NodeMCU au routeur Wi-Fi local. Comme le moniteur série // est activé, nous
pouvons voir la progression dans le moniteur
//
void setup_wifi()
{
    Serial.print(&quot; nESP32 NodeMCU se connectant à : &quot;);
    Serial.println(ssid);
    WiFi.begin (SSID, mot de passe);

    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(&quot;,&quot;);
    }
    Serial.println(&quot; nConnected to Wi-Fi&quot;);
}

```

```

}

//
// Voici la routine de configuration. Ici, le moniteur série est activé, le bouton // est
// configuré comme entrée numérique et le WiFi est configuré. // Remarque que le serveur
// MQTT (Raspberry Pi) fonctionne sur le port 1883 // La routine de rappel est appelée
// callback
//
void setup()
{
    Serial.begin(115200);                // Enable Serial Monitor // Le
    pinMode (LED, OUTPUT);              bouton est entré
    délai(100);                         // Petit délai
    setup_wifi(); client.setServer(mqtt_server, 1883); // Configuration du réseau Wi-Fi
    client.setCallback(callback);       // Définir le client
}

//
// Reconnecter au client si la connexion est perdue //
void reconnect()
{
    while (!client.connected()) {

        Serial.print("Tentative de connexion MQTT..."); si
        (client.connect(ID))
        {
            Serial.println(« connecté »);
            Serial.print (« Publication à : »);
            Serial.println(Topic);
            Serial.println(' n');
            client.subscribe (Sujet);
        }
        autrement
        {
            Serial.println(" Essayer de se reconnecter en 5 secondes");
            delay(5000);
        }
    }
}

//
// Voici la routine de rappel. Le programme saute ici lorsqu'un message

```

```

// est reçu par l'UCM NodeMCU (abonné). Activer/désactiver la LED en fonction du message
reçu (c.-à-d. la commande)
//
void callback (char *topic, byte* payload, longueur int non signée)
{
  Serial.print(« New message arrived - Topic:&quot;);
  Serial.println(Topic);
  pour(int j = 0 ; j < length; j++)
  {
    Serial.print((char)payload[j]);
  }
  Serial.println(); if((char)payload[0]
  == &apos;1&apos;){

    digitalWrite (LED, HIGH);
  }
  autrement
  {
    digitalWrite (LED, LOW);
  }
}

//
// Boucle du programme principal. Si le client est déconnecté, reconnectez-le //
boucle vide()
{

  si      (!client.connected())      reconnect();
  client.loop();
}

```

Figure 18.26 Liste du programme esp8266mqtt

Programme de flux Node-RED : La figure 18.27 montre le programme de flux pour le Raspberry Pi qui se compose de seulement 3 nœuds **rpi gpio dans** nœud pour lire l'état du bouton, un **fonction** nœud et un **En dehors de la**

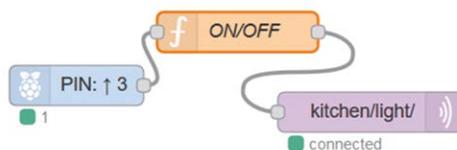


Figure 18.27 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **rpi gpio dans** nœud pour lire l'état du bouton et définir le **Pin** à GPIO 2 et **Résistance** à pullup
- Créer un **fonction** nœud nommé **ON/OFF** et entrez les déclarations suivantes dans-côté de cette fonction. La fonction renvoie "1" si le bouton est pressé (msg.payload = "0"), et "0" si le bouton n'est pas pressé :

```
if(msg.payload == "0")
  msg.payload = "1";
sinon
  msg.payload = "0" ;
renvoie msg;
```

- Créer un **En dehors** node et définir son **Serveur** à l'hôte local, **Port** à 1883 et le **Sujet** à la cuisine/lumière/
- Rejoindre les 3 nœuds et cliquer **Déployer**

Essais

Le projet peut être testé de la manière suivante :

- Mettre sous tension l'ESP8266 NodeMCU et ouvrir le moniteur série et régler sa vitesse de transmission sur 115200.
- Vous devriez voir des messages sur le moniteur série indiquant que le NodeMCU est-connecté à votre Wi-Fi et qu'il peut publier sur MQTT avec le thème cuisine/light/
- Appuyer sur le bouton. Le message suivant doit s'afficher sur le moniteur série (voir la figure 18.28). En même temps, le voyant LED s'allume :

```
Nouveau message arrivé - Sujet:cuisine/light/ 1
```

- Relâcher le bouton. Vous devriez maintenant voir le message suivant affiché sur le moniteur série. En même temps, la DEL s'éteint :

```
Nouveau message arrivé - Sujet:cuisine/light / 0
```

```
ESP32 NodeMCU connecting to: BTHomeSpot-XNH
.....
Connected to Wi-Fi
Attempting MQTT connection...connected
Publishing to: kitchen/light/

New message arrived - Topic:kitchen/light/
1
New message arrived - Topic:kitchen/light/
0
New message arrived - Topic:kitchen/light/
1
```

Illustration 18.28 Messages sur le moniteur série

Si vous publiez à partir de la console Raspberry Pi vers le thème kitchen/light/ alors vous devriez voir la LED allumée ou éteinte et les messages affichés sur le moniteur série (voir Figure 18.29)

```
pi@raspberrypi:~ $ mosquitto_pub -t "kitchen/light/" -m "1"
pi@raspberrypi:~ $ mosquitto_pub -t "kitchen/light/" -m "0"
pi@raspberrypi:~ $ █
```

Figure 18.29 Publication sur la console Raspberry Pi

18 . 11 Résumé

Dans ce chapitre, nous avons appris comment installer et utiliser MQTT avec Node-RED. Les projets présentés dans ce chapitre sont basés sur le conseil de développement NodeMCU ESP8266. Il n'y a aucune raison pour laquelle d'autres cartes de développement comme le ESP32 DevKitC ne peuvent pas être utilisées dans des applications basées sur MQTT. Ceci est laissé comme un exercice pour les lecteurs.

Dans le prochain chapitre, nous allons apprendre à utiliser les nœuds HTTP Node-RED dans des projets.

Chapitre 19 • Utilisation de HTTP dans les projets Node-RED

19 . 1 Aperçu

La palette node-red inclut les nœuds pour les requêtes HTTP, les réponses HTTP et les entrées HTTP. Le nœud de requête http est un nœud central qui peut être utilisé pour faire des requêtes http. En utilisant ce nœud, nous pouvons récupérer des pages web à partir d'un site web, envoyer et recevoir des données JSON vers un site web, faire des demandes d'API et ainsi de suite. Le nœud prend en charge la configuration suivante :

- Méthodes GET, POST, PUT et DELETE
- URL de la ressource
- Authentification
- SSL
- Réponse

HTTP est un protocole de demande et de réponse textuel qui utilise le modèle client-serveur de communication. Une requête se compose d'une commande, de certains en-têtes optionnels et d'un corps facultatif. De même, une réponse se compose d'un code d'état, d'en-têtes optionnels et d'op-Contenu du corps. Un navigateur web peut être le client qui soumet une requête HTTP au serveur.

Chaque fois que nous saisissons une URL dans la boîte d'adresse d'un navigateur, cette URL est traduite en- message de quête. Par exemple, l'URL <http://www.test.com/doc/index.html> peut être trans- lié dans une demande du format suivant :

```
GET /doc/index.html HTTP/1.1 Hôte :
www.test.com
Accept : image/gif, image/jpeg. */*
Accept-Language : en-us
Accept-Encoding : gzip, deflate
User-Agent : Mozilla/4.0
```

Ici, **Obtenir** est la méthode et **/doc/index . html** est le chemin et **HTTP/1** est la version du protocole. Le code d'état de la réponse peut avoir plusieurs valeurs. Le code 200 est le code de réussite.

Dans cette section, nous allons apprendre à utiliser le nœud de requête http dans les projets basés sur Node-RED.

19 . 2 Utilisation de HTTP GET

La requête GET est utilisée pour demander des données à partir d'une adresse web spécifiée. C'est l'une des méthodes HTTP les plus courantes. Node-RED node http request peut être utilisé pour faire une requête web. Un exemple de programme de flux est montré dans la figure 19.1 qui utilise la requête http du noeud pour faire une demande à l'URL suivante (il s'agit d'un domaine d'exemple) :

<http://example.com/hi/there?hand=wave>

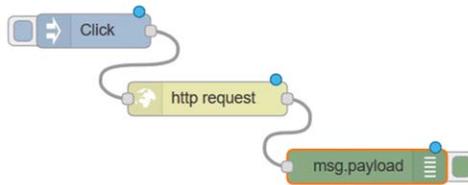


Figure 19.1 Programme de flux d'utilisation de la requête http

ce programme de flux, nous avons un **injecter** nœud, un **requête http** nœud avec le **Méthode** est réglé sur GET et le **URL** est défini comme indiqué ci-dessus. Un nœud de débogage est utilisé pour afficher la sortie dans la fenêtre Debug. La figure 19.2 montre la sortie affichée.

```
03/01/2020, 23:20:00 node: 40e29d34.413454
msg.payload : string[1256]
▼ string[1256]
<!doctype html>
<html>
<head>
  <title>Example Domain</title>
  <meta charset="utf-8" />
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <style type="text/css">
    body {
      background-color: #f0f0f2;
      margin: 0;
      padding: 0;
      font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-se
```

Figure 19.2 Sortie dans la fenêtre de débogage (seule une partie de la sortie est affichée)

Nous pouvons insérer une **html** nœud entre les **requête http** le nœud et la **déboguer** node et définir son **Sélecteur** pour div afin d'extraire les types de div, et son **Production** pour afficher uniquement le contenu textuel des éléments, sous forme d'un seul message contenant un tableau. La sortie est alors représentée dans la figure- ure 19.3 avec le programme de flux.



Figure 19.3 Programme de flux et sortie

19 3 Serveur Web

Un serveur web est un programme qui utilise HTTP (Hypertext Transfer Protocol) pour servir le web pages aux utilisateurs en réponse à leurs demandes. Ces demandes sont transmises par des clients HTTP.

En utilisant le serveur HTTP/client, nous pouvons contrôler n'importe quel appareil connecté à un processeur de serveur web sur le web.

La figure 19.4 montre la structure d'une configuration de serveur/client web. Dans cette figure, le Raspberry Pi est le processeur du serveur web et le PC (ou un ordinateur portable, une tablette ou un téléphone mobile intelligent) est le client web. Le dispositif à contrôler est connecté au processeur du serveur web. Le fonctionnement du système est le suivant :

- Le serveur web est en mode écoute, à l'écoute des requêtes du client web
- Le client web fait une demande au serveur web en envoyant une requête HTTP
- En réponse, le serveur web envoie un code HTTP au client web qui est activée par le navigateur web par l'utilisateur sur le client web et est affichée sous forme de formulaire sur l'écran du client web.
- L'utilisateur envoie des commandes (p. ex., des boutons de codage sur un formulaire client Web) et cela envoie du code au serveur Web afin que le serveur Web puisse effectuer les opérations requises.

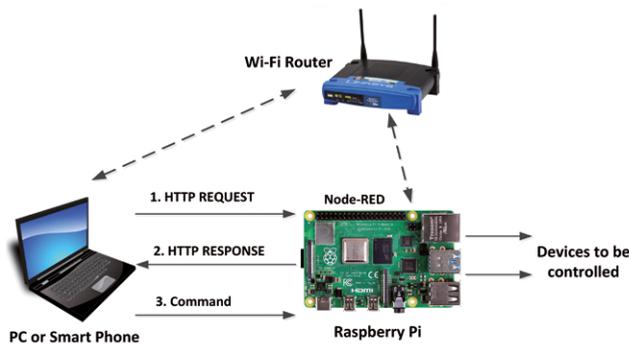


Figure 19.4 Structure du serveur Web/client

Un projet est présenté dans la section suivante pour illustrer comment un serveur web basé sur Node-RED peut être utilisé pour contrôler des relais.

19 . 4 Projet 80 – Contrôle de 4 relais à l'aide d'un serveur web

Description : Dans ce projet, un module relais composé de 4 relais est connecté au Raspberry Pi. Ces relais sont activés et désactivés à partir d'un PC (ou d'un téléphone mobile intelligent) en cliquant sur les boutons d'un formulaire de navigateur web.

Viser : Le but de ce projet est de montrer comment une application serveur web peut être conçue en utilisant Node-RED sur un Raspberry Pi.

Renseignements généraux : Dans ce projet, une carte de relais à 4 canaux (voir figure 19.5) d'Elegoo (www.elegoo.com) est utilisée. Il s'agit d'une carte relais couplée opto-optique qui possède 4 entrées, une pour chaque canal. Les entrées de relais se trouvent en bas à droite de la carte, tandis que les sorties de relais sont situées en haut. La position médiane de chaque relais est le point commun, la connexion à sa gauche est le contact normalement fermé (NC), tandis que la connexion à droite est le contact normalement ouvert (NO). Les contacts relais prennent en charge AC250V à 10A et DC30V 10A. IN1, IN2, IN3 et IN4 sont les **FAIBLE ACTIF** entrées, ce qui signifie qu'un relais est activé lorsqu'un signal LOW logique est appliqué à sa broche d'entrée. Les contacts de relais sont normalement fermés (NC). En activant le relais, les contacts actifs sont modifiés de telle sorte que la broche commune et la broche CN deviennent les deux contacts du relais et que la LED du circuit d'entrée de la carte relais correspondant au relais activé est allumée. Le VCC peut être raccordé à +3,3 V ou à +5 V. Le cavalier JD sert à sélectionner la tension du relais. **Comme le courant tiré par un relais peut dépasser 80mA, vous devez retirer ce cavalier et connecter une alimentation externe (e . g . +5V) pour fixer le JD-VCC.**

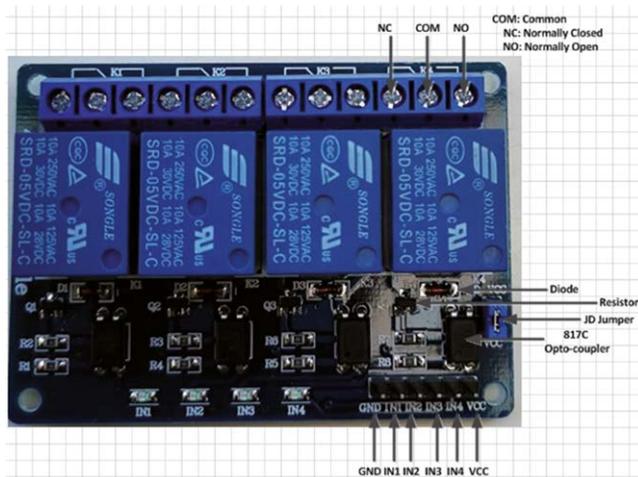


Figure 19.5 Carte de relais à 4 canaux

Schéma de principe : La figure 19.6 montre le schéma du projet.

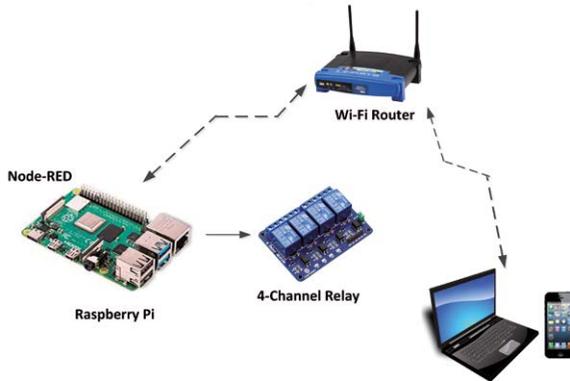


Figure 19.6 Schéma du projet

Schéma de circuit : Le schéma du projet est présenté à la figure 19.7. Les entrées IN1, IN2, IN3 et IN4 de la carte relais sont connectées aux broches GPIO Raspberry Pi 2, 3, 4 et 17 respectivement. De plus, les broches GND et +3.3V de la carte de développement sont connectées aux broches GND et VCC de la carte relais. Vous devez vous assurer que le cavalier JD est retiré de la carte. Brancher une alimentation externe +5 V sur la broche JD-VCC de la carte relais.

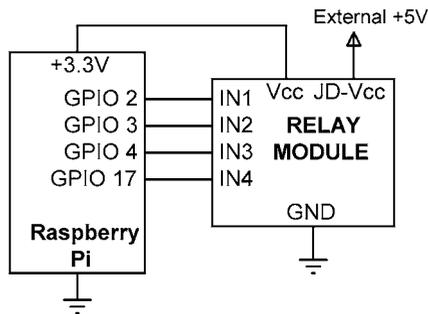


Figure 19.7 Schéma du projet

Programme de flux Node-RED : La figure 19.8 montre le programme de flux du projet qui-Système de 8 nœuds : a **http dans** nœud pour obtenir la requête HTTP, un **modèle** nœud pour stocker le code HTML, un **http out** nœud, un **fonction** nœud pour formater les données de sortie, et 4 **rpi gpio out** nœuds pour contrôler les 4 entrées de relais.

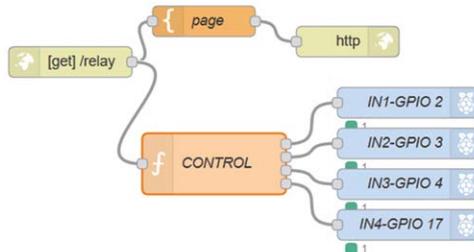


Figure 19.8 Programme de flux du projet

Les étapes sont les suivantes :

- Créer un **http dans** et configurer comme indiqué dans la Figure 19.9. Ici, l'URL est nommée **/relay** . Le fonctionnement du projet est le suivant : Lorsque l'utilisateur- Contient l'URL **192 . 168 . 1 . 202:1880/relais** depuis un navigateur web, une requête GET est envoyée au nœud Raspberry Pi **http dans**. Ce nœud active ensuite le nœud **modèle** , qui affiche un formulaire sur l'écran de l'utilisateur. Ce formulaire contient des boutons que l'utilisateur peut cliquer pour activer ou désactiver le relais choisi. Lorsqu'un bouton est cliqué, une commande est reçue par le nœud **http dans**. Cette commande est passée à node **fonction** qui extrait la partie appropriée de la commande et délivre le code pour activer la bonne **pio gpio out** , qui à son tour contrôle le relais connecté à ce nœud.



Figure 19.9 Configurer le http dans le nœud

- Créez un nœud de modèle et entrez le code HTML indiqué dans la figure 19.20 dans ce nœud. Au début, l'en-tête **Node-RED RELAY CONTROL** est affiché. La ligne suivante affiche le sous-titre **Exemple de serveur Web avec 4 relais**. Le reste du code HTML affiche un formulaire avec les noms des relais comme **RELAY1 SUR** et **RELAY1 OFF** à **RELAY4 SUR** et **RELAY4 OFF** où les états ON sont affichés en vert et les états OFF en rouge. Les éléments de formulaire sont des boutons sur lesquels vous pouvez cliquer. Une ligne du code est illustrée ci-dessous. Dans ce code, le bouton porte le nom RELAY1 ON et il est de couleur verte. Un code similaire est répété pour tous les relais.

```
<button name="RELAY1" button style="background:green", value="ON" type="submit"><b>RELAY1 SUR</b></button>
```

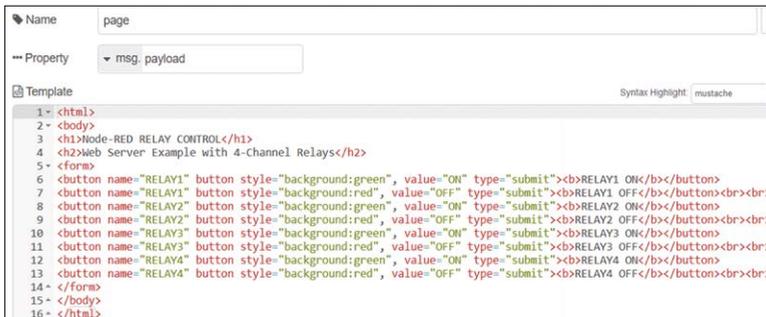


Figure 19.10 Code HTML pour le nœud de modèle

Lorsque la demande GET est reçue par le nœud http in (c.-à-d. lorsque l'utilisateur entre : 192.168.1.202:1880/relay), le formulaire illustré dans la figure 19.21 s'affiche sur l'écran de l'utilisateur.

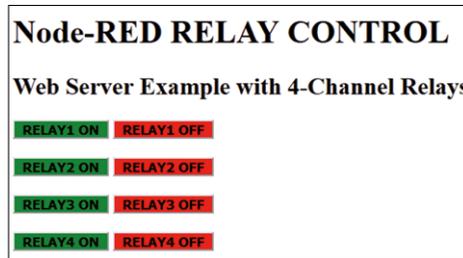


Figure 19.11 Formulaire affiché sur l'écran de l'utilisateur

- Créer un **fonction** noeud nommé **Contrôle** ayant 4 sorties, et entrez fol- Instructions suivantes à l'intérieur de ce noeud :

```

var var1 = null;
var var2 = null;
var var3 = null;
var var4 = nul;

var T = msg.req.url; switch
(T)
{
    case "/relay? RELAY1=ON" :
        var1 = {payload : 0};
        casser;
    case "/relay? RELAY1=OFF" :
        var1 = {payload : 1};
        break;
    cas "/relay? RELAY2=ON" :
        var2 = {charge utile : 0};
        casser;
    case "/relay? RELAY2=OFF" :
        var2 = {payload : 1};
        break;
    cas "/relay? RELAY3=ON" :
        var3 = {charge utile : 0};
        casser;
    case "/relay? RELAY3=OFF" :
        var3 = {payload : 1};
        break;
    cas "/relay? RELAY4=ON" :
        var4 = {payload : 0};
        casser;
    case "/relay? RELAY4=OFF" :
        var4 = {payload : 1};
        break;
}
return[var1, var2, var3, var4];

```

Lorsqu'un bouton est cliqué sur le formulaire, une commande est envoyée au serveur web en plus de certaines autres données. Ici, nous ne sommes intéressés que par la commande elle-même envoyée. La variable `msg.req.url` est chargée avec la commande reçue par le nœud `http` in lorsqu'un bouton est cliqué. Les commandes reçues sont les suivantes :

Bouton	Commande envoyée au serveur web
enfoncé RELAY1 SUR	<code>/relay? RELAY1=ON /relay?</code>
RELAY1OFF RELAY2	<code>RELAY1=OFF /relay? RELAY2=ON</code>
SUR RELAY2 SUR	<code>/relay? RELAY2=OFF /relay?</code>
RELAY3 SUR	<code>RELAY3=ON /relay? RELAY3=OFF</code>
RELAY3OFF RELAY4	<code>/relay? RELAY4=ON /relay?</code>
SUR RELAY4 SUR	<code>RELAY4=OFF</code>
RELAY4	

La fonction utilise une instruction `switch` pour vérifier le message reçu. Pour l'examen- simple, si le bouton **RELAIS 1 ALLUMÉ** est cliqué alors `msg . req. url` sera chargé avec les données `/relay? RELAY1=ON`. Variable `var1` est ensuite chargé avec 0 (rappelez-vous qu'un relais est activé si son entrée est à la logique 0) pour activer le Relais 1, connecté au GPIO 2. De même, si **RELAIS 1 OFF** est cliqué alors `msg . req. url` sera chargé avec le données `/relay? RELAY1=OFF`. Variable `var1` est ensuite chargé avec 1 pour désactiver le relais 1.

- Créer 4 `rpio gpio` les nœuds. Définir les broches de ces nœuds sur GPIO 2, GPIO 3, GPIO 4 et GPIO 17, correspondant aux entrées relais IN1, IN2, IN3 et IN4 respectivement. Régler les types sur la sortie numérique et les états de broche d'initialisation sur High (1) pour que toutes les sorties soient en logique 1 au démarrage du programme. Cela garantit que tous les relais sont désactivés au début du programme.
- Joindre tous les nœuds et cliquer **Déployer**.

Essais

- Connecter le module relais au Raspberry Pi comme indiqué sur la figure 19.7.
- Ouvrez un navigateur Web sur votre PC et entrez : **192 . 168 . 1 . 202:1880/relais**
- Le formulaire de la figure 19.21 devrait s'afficher sur l'écran de votre ordinateur
- Cliquez sur le bouton RELAY1 ON. Vous devriez entendre le relais 1 activé, et la LED- Ce relais permet de mettre le module relais sous ON. Essayez d'activer/désactiver les autres relais pour vous assurer qu'ils fonctionnent tous. La figure 19.22 montre l'URL lorsque le relais 1 est activé.



Figure 19.12 Le relais 1 est activé

- Vous pouvez également contrôler les relais depuis votre smartphone. Entrez l'URL : **192 . 168 . 1 . 202:1880/relais** comme indiqué dans la figure 19.23 et vous serez présent- Avec le formulaire pour contrôler les relais



Figure 19.13 Contrôle du relais à partir de votre téléphone intelligent

19 5 Résumé

Dans ce chapitre, nous avons appris comment utiliser le nœud de requête HTTP pour demander une page web. Un exemple de projet est également donné pour montrer comment un serveur web peut être conçu pour contrôler 4 relais à partir d'un navigateur web.

Annexe A • Le nœud de fonction

Aperçu du programme A . 1

Le nœud de fonction est l'un des nœuds les plus utiles dans Node-RED et il est basé sur JavaScript (JS). Dans cette annexe, nous allons voir comment écrire des instructions à l'intérieur de fonctions, et aussi apprendre les sujets très importants de manipulations de chaînes, chaîne intégrée, et les fonctions mathématiques, les instructions conditionnelles, la répétition, etc qui peuvent être utilisés dans les nœuds de fonction.

Variables A . 2

La bonne chose à propos de JS est qu'il ne se soucie pas des types de données. Les variables sont déclarées en utilisant le **var** mot-clé. Chaque déclaration doit être terminée par un point-virgule.

Certains ex- Les variables sont :

```
var name = "John Smith"; // string variable
var count = 10;           // variable entière //
var cnt = "20";          // variable de chaîne
```

Les variables dans JS peuvent être :

- Numérique . entier et flottant
- Chaîne . Par exemple, inclus dans " " ou dans ''
- Booléen . vrai ou faux
- Objecter e.g. objets JSON inclus dans { } e.g.
- Nul vide ou non attribué

Un nombre entier n'a pas de décimale, comme $x = 25$. Un nombre à virgule flottante a deci- Un point tel que $x = 25,322$. Les nombres peuvent être écrits en format scientifique comme indiqué ci-dessous :

```
x = 345e2 // x = 34500
```

ou

```
x = 345e-2 // x = 3,45
```

Toutes les variables sont détruites, sauf les trois types de variables : globale, contextuelle et de flux.

variables globales : ces variables conservent leurs valeurs dans tous les flux. Un exemple est donné ci-dessous où l'entier 10 est stocké dans la variable XYZ, et ensuite la variable abc est assignée à la valeur de XYZ :

```
global.set('XYZ', 10);
var abc = global.get('XYZ');
```

variables de contexte : ces variables conservent leurs valeurs dans les nœuds de fonction où elles sont déclarées seulement. Un exemple est donné ci-dessous où la variable cnt est initialisée à 0 si elle n'existe pas déjà. Ensuite, elle est incrémentée par une et stockée dans cnt. Par conséquent, chaque fois que la fonction est exécutée, la valeur de cnt sera augmentée d'un :

```
var cnt = context.get('cnt') || 0;
cnt++;
context.set('cnt', cnt);
```

variables de flux : ces variables conservent leurs valeurs dans le même flux. Un exemple est donné ci-dessous où encore entier 10 est stocké dans XYZ, et puis la variable abc est assignée à la valeur de XYZ.

```
flow.set('XYZ', 10);
var abc = flow.get('XYZ');
```

Les opérations mathématiques de base suivantes peuvent être effectuées dans JS :

Opérateurs arithmétiques valides : +, -, *, /, **, %, ++, --
Opérateurs logiques valides : <, >, ==, <=, !=
Opérateurs d'affectation valides : =, +=, *=, -=, /=, %=
Opérateurs de comparaison valides : ==, ===, !=, !==, >, <, >=, <=
Valid logical operators are: &&, ||, !
Les opérateurs bitwise valides sont : &, |, ~, <<, >>.

Notez la différence entre les opérateurs == et === de comparaison. == est utilisé pour comparer deux variables où le type de la variable (par exemple, chaîne ou nombre) n'est pas considéré. Par contre, === sert à comparer deux variables et leurs types. Par exemple, si a = 10, alors a == 10 est vrai et a = 5 est faux. Aussi, un === 10 est vrai, mais un === "10" est faux.

A . 3 Sorties multiples

Il est parfois nécessaire qu'une fonction ait plus d'une sortie. Cela peut se faire comme dans l'exemple suivant :

```
var msg1 = { payload : 'Message pour la sortie 1' };
var msg2 = { payload : 'Message pour la sortie 2' };
return [ msg1, msg2];
```

A . 4 Manipulation de la chaîne

La manipulation de chaînes est l'une des opérations les plus importantes dans JS. Des exemples d'opérations de manipulation de chaînes couramment utilisées sont donnés dans cette section.

Nombre

Convertit une chaîne en nombre.

```
var x = '10'; // chaîne '10'
var y = nombre(x); // y est numérique 10
```

Chaîne

Convertit un nombre en chaîne

```
var x = 10; // x est le nombre 10
var y = String(x); // y
est la chaîne "10"
```

Autres fonctions de chaîne

En supposant que :

```
var x = "mon nom est John, je suis âgé de 20 ans";
var y = "John Smith";
```

Ensuite :

```
var L = longueur y; // y est la longueur de la chaîne (L = 10 //
dans ce cas)

var pos = x.indexOf("name"); // première occurrence du nom, à partir de // 0 (pos = 3).
Si le texte n'est pas trouvé // alors -1 est retourné.

var pos = x.indexOf("name", 12); // recherche de nom, en commençant par position
// 12. pos = -1

var pos = x.search("name"); // recherche le nom et renvoie sa position // (pos
= 3)

var str = x.slice(3, 7); // extrait la chaîne à partir de la position 3 // et se
terminant par la position 7 (str =
// name). La position de fin n'est pas incluse
// (c.-à-d. 3 à 6)

var str = x.slice(-8, -6); // compter à partir de la fin. Extrait de
// position 6 à 8. str = ea

var str = y.slice(2); // extrait de 2 à la fin (str = hn
// Smith)

var str = x.substr(3, 4); // extrait à partir de la position 3 et // prenez 4
caractères str = nom

var str = y.substr(3); // extrait à partir de la position 3 jusqu'au // end
str = n Smith
```

```
var str = y.substr(-3);           // extrait le comptage à partir de la
                                  fin. // str = ith

var r = y.replace (« Smith », « Jones »); // replace string. r = John Jones. // Seule la
                                          première occurrence est
                                          // remplacé

var r = y.replace(/h/g, "s");      // remplacer toutes les occurrences de h par
                                  // s. r = Josn Smits

var u = y.toUpperCase();           // remplacer en majuscules. u = JOHN SMITH

var s = y.toLowerCase();           // change en minuscules. s = john smith

var c = x.concat(" ", y);         // joindre x et y.

                                  // c = mon nom, est John, je suis, 20 ans //
                                  vieux John Smith

var c = x + " " + y               // joindre x et y.
                                  // c = mon nom, est John, je suis, 20 ans // vieux
                                  John Smith

var t = y.trim();                  // supprimer les espaces avant et après. // t
                                  = John Smith

var r = y.charAt(0);               // renvoie le caractère à la position. r = J

var s = x.split(",");              // diviser la chaîne en virgules
                                  // s[0] = mon nom
                                  // s[1] = est John
                                  // s[2] = je suis
                                  // s[3] = 20 ans

var s = y.split("");               // fractionner des caractères individuels
                                  // s[0] = J
                                  // s[1] = o
                                  // s[2] = h
                                  // s[3] = n
                                  // s[4] =
                                  // s[5] = S
                                  // s[6] = m
                                  // s[7] = i
                                  // s[8] = t
                                  // s[9] = h
```

```

var s = y.repeat(2);           // répéter la chaîne y deux fois.
                               // s = John Smith John Smith

var n = y.endsWith("Smith");  // vérifier si la chaîne y se termine par Smith.
                               // n devient vrai

var n = y.startsWith (« Smith »); // vérifier si la chaîne y commence par Smith.
                               // n devient false

var n = y.includes("John");    // vérifier si la chaîne y inclut "John". // n
                               // devient vrai

```

A . 5 Fonctions mathématiques

Les fonctions mathématiques suivantes sont couramment utilisées. Toutes ces fonctions doivent être précédées du mot-clé **Math** . Par exemple, **Math.abs(x)** :

```

var x = sqrt(64);             // racine carrée. x = 8

var p = PI;                   // p = 3,14159

var s = sin(r);               // sine trigonométrique, r en radians

var c = cos(r);               // cosinus trigonométrique, r en radians

var t = tan(r);               // tangente trigonométrique, r en radians

```

Pour convertir les degrés en radians, multiplier par Pi et diviser par 180. Par exemple, pour calculer le péché à 30 degrés :

```

var s = Math.sin(30*Math.PI/180.0); // s = 0,5

var x = abs(y);                // valeur absolue de y

var x = ceil(y);               // y arrondi vers le haut au nombre entier le plus proche

var x = plancher (y);          // y arrondi vers le bas au nombre entier le plus proche

var y = exp(x);                // exponentiation de x, y=ex

var r = pow(x, y);             // r = xy

var y = round(x);              // arrondir x au nombre entier le plus proche

var y = trunc(x);              // renvoie une partie entière de x

var y = log(x);                // y = Lne(x)

```

```

var y = LN10(x);           // y = Ln10(x)

var y = asin(x);          // renvoie arcsine de x en radians

var y = acos(x);          // renvoie arccosine de x en radians

var y = atan(x);          // renvoie arctangente de x

var r = atan2(x, y);       // renvoie la tangente d'arc de y/x

```

Les fonctions hyperboliques telles que sinh, cosh, tanh, asinh, acosh et atanh sont également prises en charge.

A. 6 Conversions de nombres et vérification des nombres

Les manipulations numériques suivantes peuvent être effectuées :

En supposant que $x = 123,47$, $z = 25$ et $q = "45"$

```

Numéro:isInteger(x);      // si x est un nombre entier. Renvoie false

var y = x.toFixed(1);     // 1 chiffre après la virgule. y = 123,4

var y = x.toFixed(4);     // 4 chiffres après la virgule. y = 123,4700

var y = x.toPrecision(3); // Format à la longueur 2. y = 123

var y = x.toPrecision(4); // y = 123,5

var y = x.toPrecision(7); // y = 123,4700

var s = z.toString();     // convertir le nombre en chaîne. s = "25"

var s = z.toString(2);    // convertir avec base 2. s = 11001

var s = z.toString(16);   // convertir avec base 16. s = 10

var n = parseInt(q);      // analyse la chaîne et renvoie un entier. n = 45
                          // les espaces sont ignorés et seul le premier numéro // est
                          // retourné. Un rayon peut être spécifié

var n = parseFloat(q);    // analyse de la chaîne et retour float. n = 45
                          // les espaces sont ignorés et seul le premier numéro // est
                          // retourné, un rayon peut être spécifié

var n = eval("2 * z");    // evaluate expression. n = 50

```

A . 7 Date

Les fonctions de date sont utiles lorsque nous voulons inclure la date et/ou l'heure dans nos nœuds de fonction. Certaines des fonctions de date couramment utilisées sont données dans cette section. Les objets de données sont créés avec la nouvelle fonction Date(). Voici quelques-unes des fonctions couramment utilisées :

```

var dt = new Date();           // obtenir la date. e.g. Thu Dec 26 2019 10:52:02
                               // GMT+0000 (Greenwich Mean Time)

var d = dt.getDate();         // jour du mois, p. ex., d = 26

var d = dt.getDay();          // jour de la semaine (lundi = 1). p. ex., d = 4

var d = dt.getFullYear();     // année, p. ex., d = 2019

var d = dt.getHours();        // heures, p. ex., d = 22

var d = dt.getMinutes();      // minutes, p. ex., d = 40

var d = dt.getSeconds();      // secondes, p. ex., d = 20

var d = dt.toString();        // la date du jour en tant que chaîne, d = le
                               // jeudi 26 décembre // 2019

```

En outre, la date et l'heure peuvent être définies à l'aide des fonctions : setHours(), setMinutes(), setSeconds() etc.

Tableaux A . 8

Les tableaux sont très utiles dans tous les langages de programmation car ils permettent de accéder et accessible sous un nom. Dans les tableaux JS, ils sont indexés à partir de 0.

En supposant que nous avons la déclaration de tableau :

```

var cities = ["Londres", "Paris", "Ankara"]; var
counties = ["UK", "France", "Turquie"];

```

Nous pouvons utiliser les fonctions suivantes (seules les fonctions couramment utilisées sont données ici) :

```

var all = cities.concat(counties); // join arrays. all = ["London",
                               // "Paris", "Ankara", "UK", "France", //
                               // "Turquie"]

var all = Array.from("xyz");      // create array from string. all = // ["x",
                               // "y", "z"]

var n = villes.inclusive("Paris"); // vérifier si le tableau des villes inclut
                               // Paris. n devient vrai

```

```

var n = cities.indexOf(&quot;Paris&quot;);           // position de retour de Paris dans les villes.
                                           // n = 1. Si la chaîne n'est pas trouvée //
                                           // alors -1 est retourné

var s = cities.join();                     // joindre le tableau sous forme de chaîne séparée //
                                           // par des virgules. s =
                                           // &quot;London,Paris,Ankara&quot;;

var n = cities.length;                     // longueur des villes du tableau. n = 3

var s = cities.pop();                      // supprimer le dernier élément des villes de tableau
                                           // et puis le retourner. s = &quot;Ankara&quot;;

var s = cities.shift();                    // supprimer le premier élément des villes de tableau
                                           // et puis le retourner. s = &quot;London&quot;;

var s = cities.push (&quot;Cairo&quot;);            // ajouter un nouvel élément Cairo à la fin // des
                                           // villes de tableau et renvoie le nouveau // longueur
                                           // des villes

var s = cities.reverse();                  // inverser les éléments du tableau
                                           // cities. s = [&quot;Ankara&quot;, &quot;Paris&quot;,
                                           // « London »]

var s = cities.sort();                     // sort array cities. s = [&quot;Ankara&quot;,
                                           // &quot;London&quot;, &quot;Paris&quot;]

var s = cities.slice(1, 2);                // renvoie les éléments sélectionnés du tableau //
                                           // villes, en partant de la position 1 à // 2. s = [
                                           // &quot;Paris&quot;; ]

```

A . 9 Déclarations conditionnelles

Les énoncés conditionnels sont utilisés lorsqu'un choix doit être fait dans un programme. JS prend en charge les énoncés conditionnels suivants :

si . . else . . elseif

Il s'agit d'une des déclarations conditionnelles couramment utilisées. Un exemple est donné ci-dessous. Si plusieurs énoncés sont utilisés dans une condition, ils doivent être placés entre crochets :

```

si (x > 10)
{
  a = 0;
  b = 0;
}
sinon si (x < 10) {

  a = 1

```

```

    b = 1;
  }
  autrement
    c = 0;

```

commutateur

L'instruction switch sert à sélectionner une des nombreuses conditions. Un exemple est donné ci-dessous :

```

commutateur(s)
{
  cas 1 :
    texte = "vous avez sélectionné 1";
    casser;
  cas 2 :
    texte = "vous avez sélectionné 2";
    casser;
  cas 3 :
    texte = "vous avez sélectionné 3";
    casser;
  défaut :
    texte = "vous avez sélectionné autre chose";
}

```

A . 10 Répétition (boucles)

Les instructions de répétition sont utilisées lorsqu'il est nécessaire de créer des boucles dans les programmes. JS prend en charge les instructions de répétition suivantes :

do . . while

un exemple est donné ci-dessous où la boucle est répétée 5 fois. À la fin de la boucle, k = 5

```

var k = 0;
faire
{
  k++;
}while(k < 5);

```

pour

répéter la boucle un certain nombre de fois. Dans l'exemple suivant, la boucle est répétée 5 fois et la valeur de j = 5 à la fin de la boucle :

```

var k;
var j = 0;
pour(k = 0; k < 5; k++) {

  j++;
}

```

pendant

Boucle pendant qu'une condition est vraie. Dans l'exemple suivant, la boucle est répétée pendant que $k < 5$. At the end of the loop, $k = 5$

```
var k = 0;
et (k < 5)
{
  k++;
}
```

pour . . de

Faire une boucle à travers les valeurs d'un tableau. Un exemple est donné ci-dessous. A la fin de la boucle, variable $y = \text{"LondonParisAnkara"}$:

```
var cities = ["Londres", "Paris",
"Ankara"]; var k;
var y = "";

pour (k de villes)
{
  y = y + k;
}
```

casser

Utilisé pour quitter une boucle lorsqu'une condition devient vraie. Dans l'exemple suivant, la boucle se termine lorsque $k = 3$. À la fin de la boucle $j = 4$

```
var k;
var j = 0;
pour(k = 0; k < 10; k++) {

  j++;
  si (k == 3)
    casser;
}
```

continuer

Saute la boucle lorsqu'une condition devient vraie. Dans l'exemple suivant, la somme de num- Les valeurs de 0 à 10 sont calculées. Lorsque $k = 5$, la boucle continue mais ne correspond pas à cette valeur. A la fin de la boucle, $j = 50$ et non 55

```
var k;
var j = 0;
pour(k = 0; k <= 10; k++) {

  si (k == 5) continue;
}
```

```

    j = j + k;
  }

```

Fonctions de 11

Les fonctions sont utiles lorsqu'une certaine opération doit être répétée plusieurs fois dans un programme. Les fonctions sont également utilisées pour décomposer un programme complexe en sections plus petites et plus faciles à gérer. Dans JS, une fonction est déclarée avec le mot-clé `function`, suivi du nom de la fonction. Une fonction peut avoir des paramètres facultatifs entre parenthèses. Le corps d'une fonction doit être placé dans des crochets.

Dans l'exemple suivant, une fonction appelée `Add` est créée qui a 2 arguments. Le fonction ajoute ces arguments et renvoie le résultat au programme appelant :

```

function Add(a, b) {

  renvoie a + b;
}

```

Appeler la fonction comme suit :

```
y = Add(3, 4)
```

Où `y` prend la valeur de 7

A . 12 Exemples

Quelques exemples sont donnés ci-dessous pour montrer comment certaines des fonctions intégrées peuvent être utilisées dans les nœuds de fonction.

Exemple

La figure A.1 montre un diagramme de flux avec un nœud de fonction et un nœud de débogage. En supposant que l'entrée au nœud de fonction est :

```
T = 15, 500, 752
```

Écrire une fonction avec 3 sorties pour extraire 15, 50 et 75. Afficher les résultats dans la fenêtre de débogage.

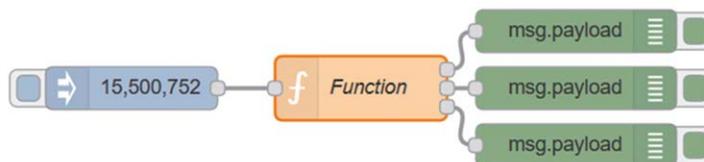


Figure A.1 Programme de flux de l'exemple 1

Solution

La fonction doit avoir les instructions suivantes. `split()` est utilisée pour extraire les éléments :

```
var F = msg.payload.split(","); var1 =  
{payload : F[0]};  
var2 = {charge utile : F[1]};  
var3 = {charge utile : F[2]};  
retour [var1, var2, var3];
```

La figure A.2 montre les données affichées dans la fenêtre de débogage.



Figure A.2 Données affichées dans la fenêtre de débogage

Annexe B • Programmes de flux utilisés dans le livre

Aperçu du programme A . 1

Le site web du livre comprend tous les programmes de flux utilisés dans ce livre. Les programmes sont enregistrés sous forme de documents Word et on leur donne le nom du projet auquel ils appartiennent, p. ex., Projet 29. Certains programmes de flux n'appartiennent pas à un projet et pour ceux-ci, le nom du chiffre auquel ils appartiennent est donné. p. ex.

A . 2 Utilisation des programmes de flux

Les étapes à suivre pour utiliser les programmes de flux du site Web du livre sont les suivantes :

- Sélectionnez le programme de flux requis sur le site Web
- Ouvrez le programme et copiez-le dans le presse-papiers
- Démarrer le tableau de bord Node-RED et cliquer sur **Menu -> Import**
- Coller le programme dans la fenêtre ouverte et cliquer **Importation**
- Vous devriez voir le programme de flux ajouté à l'espace de travail Node-RED
- Cliquez **Déployer**

Annexe C • Composants utilisés dans le livre

Transformateurs :

Raspberry Pi 4, ESP32 DevKitC, ESP8266 NodeMCU, Arduino Uno

Composants :

- 4 x LED rouge
- Résistance 4 x 470 ohms
- 1 x LED verte
- 1 x LED orange
- 1 pot de 10 000
- 1 x 220 ohm résistance
- 2 x résistances 1K
- 2 x résistances 2K
- 1 buzzer
- 1 x bouton
- 1 x LCD I2C
- 1 x LCD parallèle
- 1 x DHT11
- 2 x capteurs ultrasoniques
- 1 x MCP3002 ADC
- 1 x TMP36DZ capteur
- 1 x GPS Cliquez sur la carte
- 2 x tRF Click Board
- 2 x Antenne pour tRF
- 1 x Sense HAT
- 1 x 4 module de relais (Elegco)

Index**A**

Accélération,	230
CDA,	147
Noeuds avancés, Alexa,	58
Amazon Alexa,	263
Capteur analogique,	263
nœuds d'analyse,	146
Arduino,	58
Tableaux,	199, 245
Pression atmosphérique,	315
Auto-pan,	76
	213

B

Graphique à barres	157
Débit en bauds,	198
Bluetooth,	272
Rupture,	318
Courtier,	279
Bouton,	106
Nœud de bouton,	163
Buzzer,	92, 135

C

Cat,	34
Graphique,	156
Clavardage,	184
Nœud de commentaire,	106
variable de contexte,	87, 309
en-tête de boussole,	229
déclarations	316
conditionnelles, continuer,	318
Copier le nœud,	55
Noeuds centraux,	55
Port CSI,	16

D

Tableau de bord,	153
Nœuds du tableau de bord,	153
Date et heure,	47
Supprimer,	300
DevKitC,	257
DHT11,	124, 253
DHT22,	124

Utilisation du disque	39
Capteur de distance,	129
Port DSI,	16
Cycle de travail,	111

E

Courriel,	74, 93
Envoyer un courriel à.	93
Envoi d'un courriel;	97
ESP32,	257
ESP8266,	283
Etcher,	21
Compteur d'événements,	122
Exportation,	54

F

Opération de fichier,	33
Autorisation de fichier,	28
Firmata,	247
LED clignotante,	85
Variables de flux,	309
fonctions,	319
Nœuds de fonction,	57, 309

G

Jauge,	154, 251
Get,	300
Connecteur GPIO,	80
broches GPIO,	81
GPS,	204
GPS cliquez,	206
Gyroscope,	227

H

Halt,	40
HC-SR04,	129
HD44780,	137
HDMI,	16
Graphique à barres	158
horizontales, HTTP,	300
Demande HTTP,	57
humidité,	120

I

Ifconfig,	42
Importation,	54

Nœuds d'entrée,	55	P	
GSI,	222	LCD parallèle,	137
J		Ping,	187
Jason,	72	Hauteur,	232
Joystick,	233	port PoE,	16
L		Pression,	242
Latitude,	208	Ps,	39
CLD,	113	POSTE,	300
Jeu de caractères LCD,	148	Éditeur,	279
Couleurs des DEL,	235	Appuyer sur le bouton.	106
Matrice LED,	234	Mettre,	300
Graphique linéaire,	156	Mastic,	23
LM35DZ,	201	<small>Gestion privée de patrimoine</small>	110
Convertisseur de niveau	217	R	
logique, longitude,	208	Random node, Random	121
M		number, Raspberry Pi	60
Carte,	212	nodes, Raspbian Buster,	58
Mark,	110	Remote access,	18
Fonctions mathématiques,	313	Repetition,	22
MCP3002,	142	Radio RF,	317
Signifie	67	Roulez,	222
Carte Micro SD,	17	Faire pivoter l'écran.	232
MOSFET,	216		243
Mosquitto,	280	S	
Événements de mouvement,	229	SCL,	
MQTT,	278	Données de défilement,	113
MQTT en,	281	SDA,	242
MQTT out,	281	Envoyer UDP,	113
Plusieurs GPIO,	98	Chapeau de sens,	178
Calibre multiple,	160	Sens des nœuds,	225
Entrées multiples,	62	communication série,	226
Plusieurs LED,	173	arrêt,	198
Sorties multiples,	65	Nœud de curseur,	40
N		Nœud lisse,	161
Ngrok,	269	Les nœuds sociaux,	67
NodeMCU,	285	Espace,	58
Configuration du noeud-RED,	78	SPI,	110
O		SSH,	143
Openweathermap,	71, 91	Nœuds de stockage,	23
Orientation,	231	Manipulation de chaîne,	58, 190
Nœuds de sortie,	56	abonné,	310
		Commutateur de nœud,	279
		Super utilisateur,	109
		Changer,	37
			317

Commutateur de nœud, 165

T

Prévisions météo, TCP, 166

Conversion de température, 170

nœud texte, 52

TightVNC 165

Date et heure 26

TMP36, 190

Sujets, 146

Feux de circulation, 279

Trf clic, 102

Nœud de déclenchement, 223

Twitter, 86. 104

Twitter dans, 76

Twitter out, 76

76

U

UDP,

Réception UDP, 170

Ultrasons, 176

Conversion en unités, 129

URL, 62

Clavier USB, 301

Port USB, 17

Utf-8, 220

195

V

Variables, 309

VNC, 25

Serveur VNC, 25

Voltmètre, 144

W

Météo, 70

Serveur Web, 301

Websocket, 56

Pendant ce temps, 318

Whoami, 36

Wi-Fi, 170

Caractère générique, 33

Y

Yaw, 232

Programming with Node-RED

Dogan Ibrahim



Prof. Dr. Dogan Ibrahim is a Fellow of the Institution of Electrical Engineers. He is the author of over 60 technical books, published by publishers including Wiley, Butterworth, and Newnes. He is the author of over 250 technical papers, published in journals, and presented in seminars and conferences.

ISBN 978-1-907920-88-2



Elektor International Media BV
www.elektor.com

The Internet of Things (IoT) is becoming a major application area for embedded systems. As a result, more and more people are becoming interested in learning about embedded design and programming. Technical colleges and universities are moving away from legacy 8 and 16-bit microcontrollers and are introducing 32-bit embedded microcontrollers to their curriculums. Many IoT applications demand precision, high processing power, and low power consumption.

Produced by IBM, Node-RED is an open-source visual editor for wiring the Internet of Things. Node-RED comes with a large number of nodes to handle a multitude of tasks. The required nodes are selected and joined together to perform a particular task. Node-RED is based on flow type programming where nodes are configured and joined together to form an application program. There are nodes for performing complex tasks, including web access, Twitter, E-mail, HTTP, Bluetooth, MQTT, controlling GPIO ports, etc. One particularly nice aspect of NodeRED is that the programmer does not need to learn how to write complex programs. For example, an email can be sent by simply joining nodes together and writing only a few lines of code.

The aim of this book is to teach how Node-RED can be used in projects. The main hardware platform used with most of the projects in this book is Raspberry Pi 4. Chapters are included to show how Node-RED can be also be used with Arduino Uno, ESP32 DevKitC, and the ESP8266 NodeMCU microcontroller development boards.