Apprendre à fabriquer sa propre carte d'extension Pi HAT

Une carte d'extension compatible avec les spécifications officielles de la Fondation Raspberry Pi

Par Richard Hayler - f-leb (traducteur)

Date de publication : 11 septembre 2016

TOUT PUBLIC

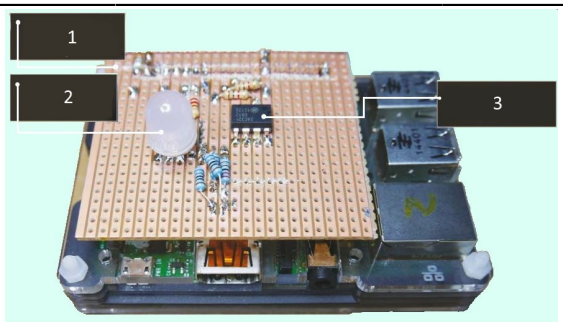
Il existe maintenant une quantité incroyable de cartes additionnelles HAT (Hardware Attached on Top) disponibles pour étendre les fonctionnalités matérielles de votre Raspberry Pi. Mais ne vous êtes-vous jamais demandé comment concevoir et réaliser une carte HAT en prototype ? Ce tutoriel va vous montrer comment débuter dans la réalisation de votre propre carte d'extension, compatible avec les spécifications HAT.

Commentez

En complément sur Developpez.com

- Raspberry Pi - Déballage et installation
- Raspberry Pi - Python et le port GPIO

I - Ressources nécessaires



1 - *Soudez tous vos composants sur la plaque* veroboard*, une fois que tout est en place.*	**2** - *Notre carte HAT est toute simple, avec une LED comme composant principal.*	**3** - *Pour répondre aux spécifications officielles, une carte HAT doit avoir une mémoire EEPROM.*

Ressources nécessaires
• Une mémoire EEPROM (1) (un CAT24C32 est recommandé). **Avertissement :** les mémoires EEPROM peuvent être facilement endommagées par l'électricité statique. Donc, prenez soin de toucher un conducteur électrique relié à la terre avant de les manipuler. Idéalement, portez un bracelet anti électricité statique, et au minimum, rangez ce pull-over en nylon dans votre placard.
• Une plaquette de câblage rapide et quelques fils de câblage.
• Quelques LED (ou une LED RVB), assortiment de résistances électriques.
• **Les spécifications du Pi HAT**

II - Les cartes d'extensions Pi HAT

Les premières cartes d'extension HAT sont arrivées avec le lancement en 2014 du modèle B+ du Raspberry Pi. Depuis, d'incroyables et fantastiques **cartes HAT** sont apparues, et la plupart ont été décrites dans **les différents numéros du MagPi** : l'*Unicorn HAT*, le *piano HAT*, le *Sense HAT* et l'*Explorer HAT*. Officiellement, HAT signifie *Hardware Attached on Top*, bien que votre expert soupçonne que l'acronyme soit venu après le nom. Mais voyez-vous de meilleures façons de nommer une carte additionnelle qui s'enfiche par-dessus le Pi en recouvrant sa surface ?

Bien que vous puissiez fabriquer et vendre des cartes d'extension sans vous conformer aux spécifications standards des cartes HAT, il y a de nombreux avantages à s'y conformer. Avoir des cartes qui présentent les mêmes dimensions permet de s'assurer qu'elles rentreront dans la plupart des boîtiers pour Raspberry Pi, et inclure une mémoire EEPROM permet au système d'exploitation d'identifier la carte HAT et de configurer les ressources matérielles nécessaires au démarrage du Pi.

Les instructions qui suivent fonctionneront aussi bien pour la dernière version de Raspbian (Jessie), que pour l'ancienne Wheezy.

III - Activation du support du bus I2C

Comme d'habitude, commencez par un `sudo apt-get update`, suivi d'un `sudo apt-get upgrade` pour **mettre à jour votre système Raspbian**.

Ensuite, il faut activer le support du bus I^2C dans le noyau Linux. Pour cela, tapez :

```
sudo raspi-config
```

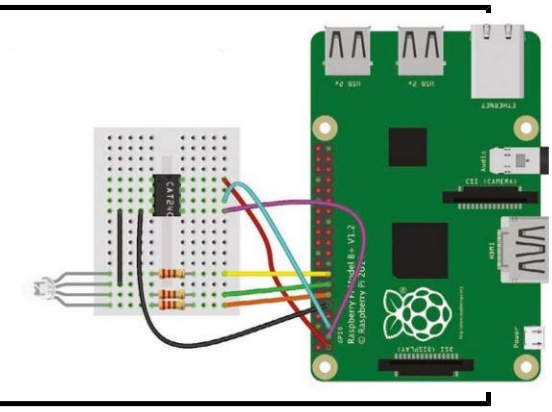
Sélectionnez **Advanced Options**, descendez avec la touche fléchée vers le bas puis sélectionnez **A7 I2C**. Répondez favorablement aux deux questions qui vous sont posées en vous assurant que **Yes** est bien mis en surbrillance. Si vous utilisez Jessie, vous pouvez également activer le support de l'I^2C via l'outil de configuration disponible dans l'interface graphique fenêtrée. Dans les deux cas, vous serez alors invité à redémarrer. Une fois que le Pi a redémarré, installez le paquet des utilitaires de gestion de l'I^2C avec la commande :

```
sudo apt-get install i2c-tools
```

IV - Prototypage du circuit

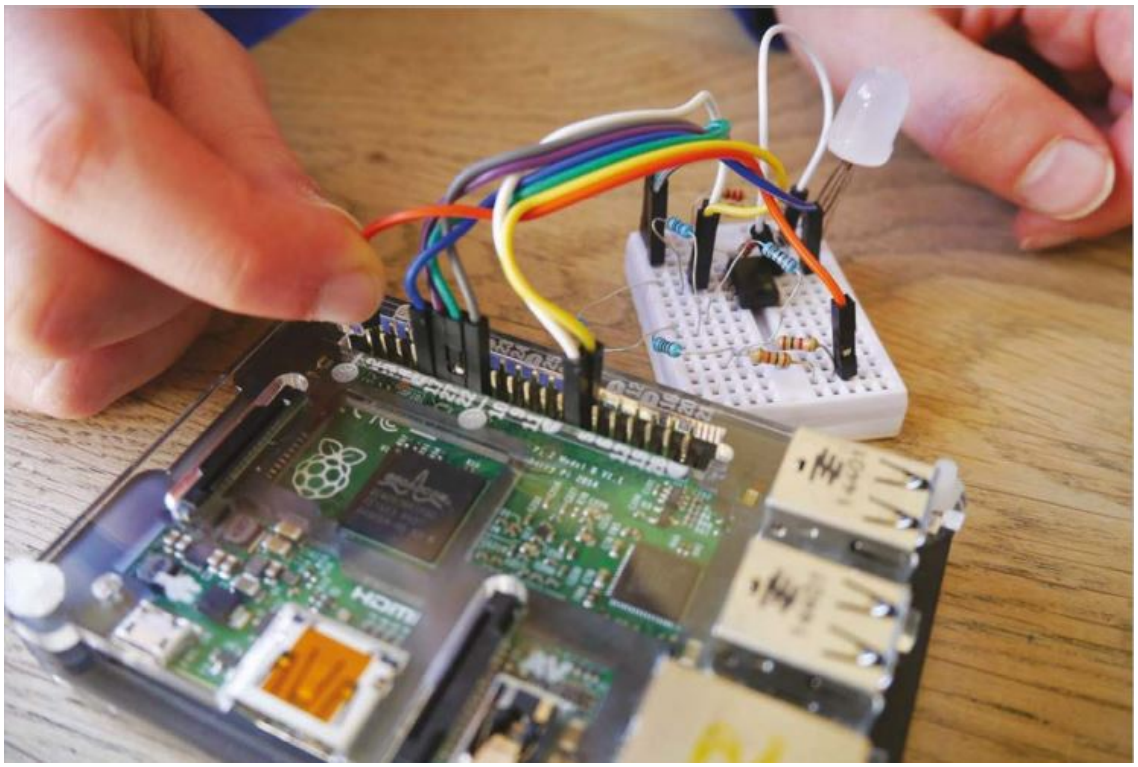
Il est maintenant temps de réaliser le prototype du circuit avec l'EEPROM sur une plaquette de câblage rapide. On commence par mettre en place le circuit de la **figure 1** ci-dessous :

Figure 1 : l'EEPROM devra être connectée aux broches SDA ou SCL pour la programmation. Si vous n'utilisez pas un 24C32, vous devrez revoir les connexions (consultez la documentation constructeur du composant).



Si vous n'utilisez pas une EEPROM 24C32, vous devrez peut-être modifier le câblage concernant les broches SDA et SCL. Consultez la documentation constructeur (*datasheet)* du composant au besoin.

Votre carte HAT doit évidemment proposer des fonctionnalités observables. Pour faire simple, ici le circuit ci-dessus comporte seulement une LED tricolore RVB, mais vous pouvez utiliser tout type de composant selon les nouvelles fonctionnalités que vous souhaitez offrir à votre Pi.



Faire d'abord un prototype de circuit sur une plaquette de câblage rapide ou un Explorer HAT permet de s'assurer que tout fonctionne et de faire des ajustements rapides.

Alimentez le Pi. Vous remarquerez peut-être que la LED est déjà allumée, même faiblement, cela parce que le Pi ne sait pas ce qui est connecté sur son port et que les sorties GPIO sont laissées « flottantes » au démarrage. Voici une chose que votre Pi devra prendre en compte en configurant au démarrage les broches utilisées par la carte HAT.

Le bout de code Python, **plus loin dans l'article** (myoh-rgp.py), permet d'interagir avec la LED. Vous aurez besoin auparavant d'installer la bibliothèque **gpiozero**.

Ensuite, vérifiez les périphériques I^2C détectés sur le second bus I^2C :

```
i2cdetect -y 1
```

Vous devriez voir s'afficher un tableau des périphériques détectés sur le bus I^2C. En supposant que vous n'ayez rien d'autre de connecté, toutes les entrées, sauf une, doivent être vides en laissant apparaître deux tirets « -- ». Normalement, votre EEPROM devrait apparaître dans la colonne la plus à gauche (0) sur la ligne repérée « 50 », à côté du nombre « 50 » sur l'axe horizontal.

Si votre EEPROM n'apparaît pas où qu'un message d'erreur s'affiche (*No such file or directory*), essayez avec l'autre bus :

```
i2cdetect -y 0
```

Si l'EEPROM n'apparaît toujours pas, vérifiez à nouveau votre câblage et assurez-vous d'avoir bien activé le support de l'I^2C.

V - Programmation de la mémoire EEPROM

Maintenant, vous avez besoin d'outils logiciels. Clonez la documentation de référence des cartes HAT et les outils nécessaires depuis la **plateforme GitHub**. La documentation vaut la peine d'être lue, car elle explique l'idée sous-jacente à la spécification de carte HAT.

Vous allez utiliser les outils fournis pour flasher l'EEPROM. Avant tout, lancez la compilation de l'utilitaire **eepmake** :

```
cd hats/eepromutils
make
```

Il faut ensuite modifier le script **eepflash**, qui par défaut suppose que l'EEPROM est sur le premier bus I^{2}C. Si c'est le cas, vous pouvez bien entendu passer cette étape. Copiez le fichier **eepflash.sh** et modifiez-le au besoin avec votre éditeur préféré :

```
cp eepflash.sh eepflash1.sh
nano eepflash1.sh
```

Modifiez toutes les mentions `i2c-0` en `i2c-1`, et `i2c-0/0-0050` en `i2c-1/1-0050`, puis sauvegardez les modifications.

Ensuite, vous devez modifier le fichier de configuration-modèle selon votre paramétrage. Ouvrez le fichier **eeprom_settings.txt** dans votre éditeur de texte préféré et modifiez les valeurs des différents champs.

La signification de la plupart des champs ne nécessite pas d'explications supplémentaires et vous trouverez davantage d'explications dans les spécifications. Cependant, le champ **UUID** est particulièrement important et doit être conforme à la RFC 4122 pour que chaque carte HAT puisse être identifiée de façon unique, comme s'il s'agissait du numéro de série de la carte HAT. Cela permet également d'utiliser plusieurs cartes HAT en les empilant.

Vous devez ensuite convertir les données textuelles du fichier en données binaires qui pourront être écrites dans l'EEPROM.

```
./eepmake eeprom_settings.txt eeptest.eep
```

Enfin, transférez les données dans la mémoire EEPROM - il s'agit de la partie flashage à proprement parler. Vous verrez un message d'avertissement vous demandant de confirmer avant de poursuivre l'opération de flashage :

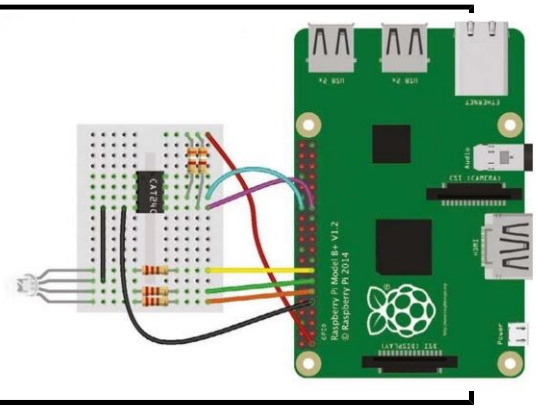
```
./.eepflash1.sh -w -f=eeptest.eep -t=24c32
```

Vérifiez que vous utilisez bien la version du script modifiée plus tôt. Si vous utilisez un autre type de mémoire EEPROM, changez le paramètre **-t** en conséquence.

VI - Premiers tests

Afin de tester votre nouvelle création, vous devez modifier le câblage de votre circuit de façon à ce que l'EEPROM soit détectée par le Pi au démarrage. Les broches GPIO 27 et 28 sont justement réservées à la détection de carte HAT. Coupez à nouveau l'alimentation du Pi, et modifiez le câblage conformément à la **figure 2**. Ajoutez les résistances de tirage 3,9 kΩ (*pull-up*) afin d'éviter les broches « flottantes ».

Figure 2 : une fois l'EEPROM programmée, vous pouvez passer au test de votre carte HAT. Il faut alors modifier le câblage et connecter l'EEPROM aux broches ID_SC et ID_SD (2) . Vous devrez également ajouter deux résistances de tirage (*pull-up*).



Maintenant, rallumez le Pi et laissez démarrer le système. Si tout se passe correctement, le répertoire **/proc/device-tree/hat** devrait être présent, et le contenu de ses fichiers correspond alors aux détails du fichier de configuration **eeprom_settings.txt**. Vous devriez également vous rendre compte que les broches GPIO utilisées par la LED RVB sont maintenant correctement configurées, et la LED RVB doit être complètement éteinte.

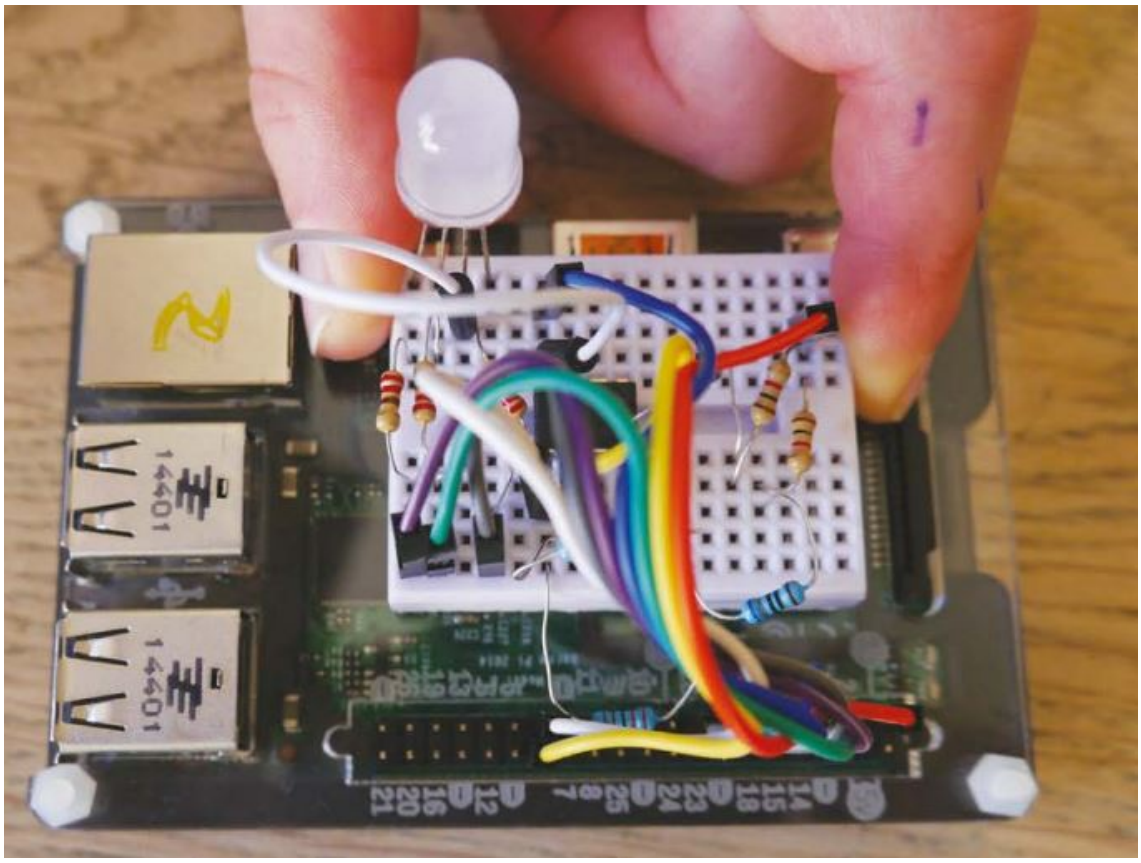
Félicitations ! Vous avez programmé une mémoire EEPROM que le Pi reconnaît conformément aux spécifications d'une carte HAT.

Comme test final, lancez le programme *myoh-rgb.py* ci-dessous et constatez la séquence d'apparition des couleurs de la LED : rouge, vert puis bleu.

myoh-rgb.py
```
from gpiozero import RGBLED
from time import sleep

led = RGBLED(22,27,17)

led.on()
sleep(0.5)
led.off()
led.red = 1
sleep(0.5)
led.red = 0
led.green = 1
sleep(0.5)
led.green = 0
led.blue = 1
sleep(0.5)
led.blue = 0
```



Testez le fonctionnement du prototype sur une plaquette de câblage rapide, en exécutant le code Python proposé.

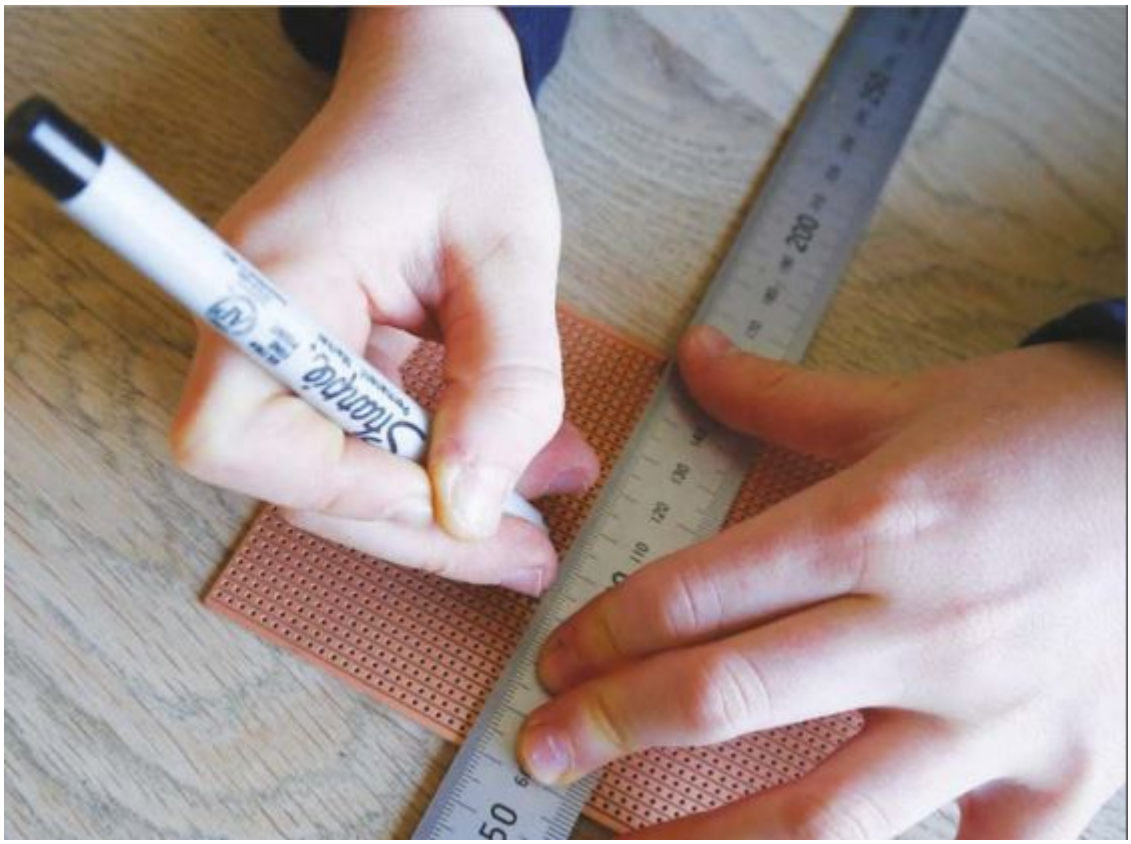
Bien sûr, cette première carte HAT est très basique, mais il y a encore beaucoup de possibilités à exploiter pour configurer les ressources au démarrage, et ainsi utiliser des matériels beaucoup plus sophistiqués qui seront facilement supportés par le Pi. La carte **Sense HAT** est un bel exemple d'utilisation de cette technique.

VII - Aller plus loin

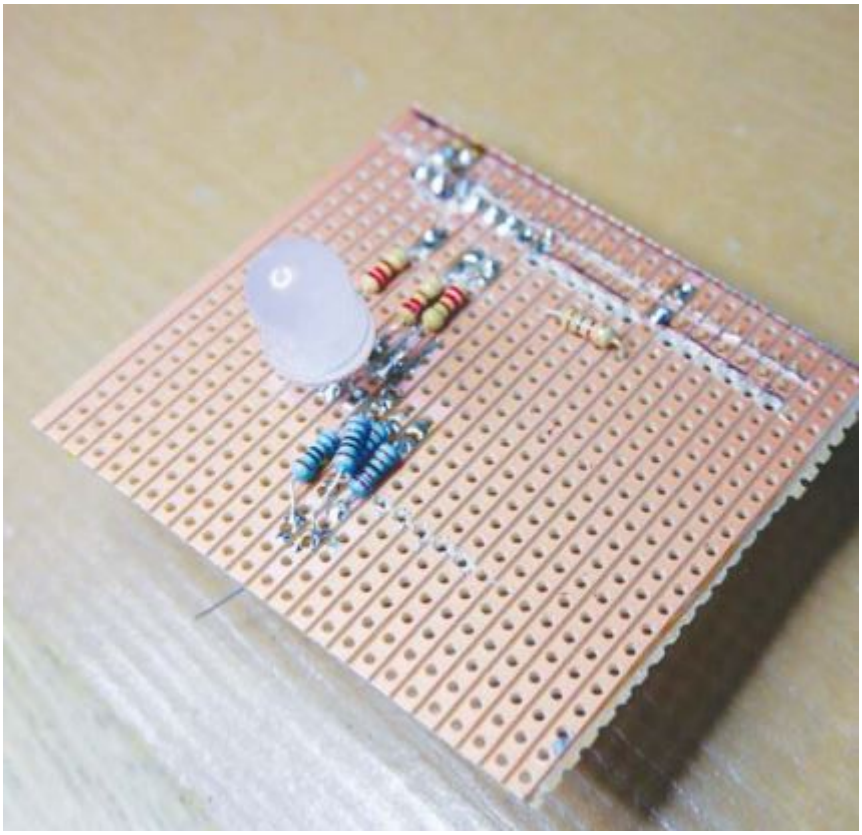
Maintenant que vous avez conçu une carte HAT toute simple, et que vous avez bien examiné le processus, songez à toutes ces cartes additionnelles que vous pouvez créer.

Mais si vous voulez poursuivre le processus de conception et de réalisation d'une carte HAT complète, vous pouvez maintenant passer de la plaquette de câblage rapide à un véritable circuit imprimé.

Les spécifications de la Fondation Raspberry Pi définissent également les propriétés physiques de la carte, jusqu'au rayon de l'arrondi dans les coins de la carte, l'intégration de ports pour les câbles d'une caméra ou d'un écran, etc. Pour du prototypage, vous n'aurez sans doute pas besoin de vous préoccuper de tout cela. Tous les détails de conception d'une carte sur *verobard* (voir **la première image en début de tutoriel**) sont disponibles dans **le dépôt GitHub de ce projet**. Si vous voulez aller encore plus loin, le très utile logiciel **Fritzing** dispose de modèles de circuits imprimés qui vous aideront à concevoir les plans appropriés au lancement de la fabrication.



Découpez un morceau de plaque veroboard à la bonne taille, de façon à recouvrir la surface du Pi avec la carte HAT.



Pour aller plus loin, soudez un connecteur femelle GPIO et le reste des composants en suivant les indications : github.com/topshed/MYOH

VIII - Notes de la Rédaction de Developpez.com

Cet article est une traduction de l'article écrit par Richard Hayler et paru dans le n° 42 du magazine **TheMagPi**, sous le titre **Make your own Pi HAT**.

Les fichiers utilisés pour ce tutoriel sur GitHub : **github.com/topshed/MYOH**.

Nous remercions les membres de la Rédaction de **developpez.com** pour le travail de traduction et de relecture qu'ils ont effectué, en particulier : **f-leb**, **scsab**, et **milkoseck**.

1 : EEPROM : *Electrically Erasable Programmable Read-Only Memory*, ou mémoire morte effaçable électriquement et programmable.
2 : C'est une des nouveautés du port GPIO du Pi avec ses 40 broches. Les deux broches P27 (ID_SD, *EEPROM Data*) et P28 (ID_SC, *EEPROM Clock*) sont réservées à **la communication I2C** avec une mémoire EEPROM. L'EEPROM contient les données d'identification de la carte d'extension et renseigne le Pi au démarrage (grâce au mécanisme du *Device Tree*) sur la configuration des ressources matérielles et logicielles nécessaires à son fonctionnement.