Montage 1

Le premier sketch

Au sommaire :

- la déclaration et initialisation d'une variable ;
- la programmation d'une broche numérique en sortie (`OUTPUT`) ;
- l'instruction `pinMode()` ;
- l'instruction `digitalWrite()` ;
- l'instruction `delay()` ;
- le sketch complet ;
- l'analyse du schéma ;
- la réalisation du circuit ;
- un exercice complémentaire.

Le phare "Hello World"

Eh bien Ardus, les choses deviennent maintenant sérieuses, sérieuses certes mais pas vraiment difficiles car nous allons avancer doucement. La plupart des manuels sur les langages de programmation commencent par la présentation d'un programme appelé `Hello World`. Ce programme est généralement le premier que le débutant est amené à découvrir. Il donne un premier aperçu de la syntaxe du nouveau langage de programmation et imprime le texte `"Hello World"` dans une fenêtre. Le nouveau langage de programmation se présente ainsi à vous et au reste du monde, et semble dire `"Eh je suis là ! Utilisez-moi"`.

Nous avons à présent un petit problème car notre Arduino n'a pas d'écran à l'origine, et donc pas de console de visualisation pour nous

informer. Alors que faire ? Si aucune communication sous une forme écrite n'est possible, alors peut-être l'est-elle avec des signaux optiques ou acoustiques? Nous optons pour la variante optique car une diode électroluminescente – également appelée LED – se branche sans problème à l'une des sorties numériques et attirera à coup sûr l'attention. J'ai été moi-même très étonné quand ça a marché du premier coup.

Composants nécessaires

L'exemple étant très simple, seules une LED et une résistance série sont nécessaires.

1 LED rouge

1 résistance de 220 Ω

Dans le chapitre 3 sur les principes de base Arduino, je vous ai dit que la carte comportait entre autres quelques LED, dont l'une était reliée directement à la broche numérique 13 et possédait sa propre résistance série. Ceci dit, aucun composant externe ne devrait avoir besoin d'être branché sur la carte.

Cette LED se trouve à gauche près du logo Arduino.

Pour aller plus loin

Quand vous reliez pour la première fois une carte Arduino flambant neuve à votre ordinateur, cette LED incorporée s'allume pendant une seconde. Un premier sketch incluant cette fonctionnalité de base a donc été chargé à l'usine une fois la carte assemblée.

Code du sketch

Le code du sketch est le suivant pour ce premier exemple :

```
int ledPin = 13; //Déclarer + initialiser broche numérique 13
                 //comme sortie
void setup(){
   pinMode(ledPin, OUTPUT);  //Broche numérique 13 comme sortie
}

void loop(){
   digitalWrite(ledPin, HIGH); //LED au niveau HIGH  (5V)
   delay(1000);                //Attendre une seconde
   digitalWrite(ledPin, LOW);  //LED au niveau LOW  (0V)
   delay(1000);                //Attendre une seconde
}
```

Vous pouvez vérifier le code si vous l'avez transmis dans l'éditeur. Le compilateur essaye alors de le traduire. Le tableau 1-1 indique les deux étapes fondamentales.

Icône	Fonction
	Commencer la vérification par compilation.
	Commencer la transmission au microcontrôleur une fois la compilation terminée.

Tableau 1-1
Étapes pour compiler et transmettre

À la fin, un message indique que la transmission a réussi. La mémoire nécessaire y est donnée en octets, de même que la mémoire totale à disposition.

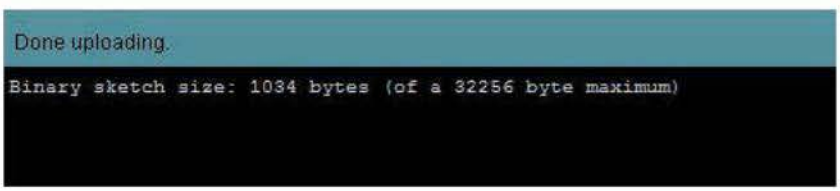


Figure 1-1
Message d'état et affichage des informations sur la mémoire

J'ai une question avant que vous ne poursuiviez. Comment doit-on considérer les mots OUTPUT, HIGH ou LOW ? S'agit-il de mots-clés ? L'IDE les fait en tout cas apparaître en couleurs.

Bien vu Ardus ! Il me faut ici développer un peu. Si vous initialisez des variables avec des valeurs dont vous êtes seul, de prime abord, à connaître la signification, les autres personnes appelées à travailler avec le code en question auront certainement du mal à comprendre. Que peut bien vouloir dire par exemple le nombre 42 ? Un tel style de

programmation n'est pour moi pas très convaincant. Il se trouve que nous avons pris précisément le nombre `13` pour désigner la broche de la sortie numérique, mais nous entendons rendre le code de programme un peu plus évocateur à l'avenir. De telles valeurs suspectes survenant dans le code source sont au fait appelées *magic numbers*. Mais revenons aux mots indiqués en couleurs. Il s'agit ici de *constantes*, c'est-à-dire des indicateurs qui, tout comme les variables, ont été initialisés avec une valeur mais ne peuvent plus être modifiés. C'est pourquoi on les appelle des constantes. Ce nom en dit déjà beaucoup plus que n'importe quelle valeur ésotérique. Nous reviendrons bientôt sur les instructions `pinMode` et `digitalWrite` et j'expliquerai alors l'importance de ces constantes.

Revue de code

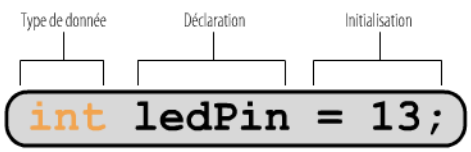
Nous déclarons et initialisons au départ une *variable globale* portant le nom de `ledPin`, dont le type exact de donnée est `int` (*int* = integer) et qui apparaît dans toutes les fonctions avec la valeur `13`.

Tableau 1-2 ▶
Variable nécessaire et son objet

Variable	Objet
`ledPin`	Contient le numéro de broche pour la LED sur la broche de sortie numérique 13.

L'initialisation équivaut à une affectation de valeur avec l'opérateur d'affectation `=`. Déclaration et initialisation occupent ici une ligne. Le mode d'écriture est donc plus court que la variante à deux lignes.

Figure 1-2 ▶
Déclaration et initialisation de variable



Si vous décidez d'écrire ces deux actions séparément, aucun problème mais les deux lignes ne doivent pas se suivre. Cet exemple produit une erreur :

```
int ledPin;   //Déclarer la variable
ledPin = 13;  //Initialiser la variable avec valeur 13 -> Erreur !!!
```

La déclaration de la variable globale `ledPin` se fait hors des fonctions `setup` et `loop`. L'initialisation se fait quant à elle dans la fonction `setup` qui n'est appelée qu'une fois. Le code du sketch correct est alors le suivant :

```
int ledPin;     //Déclarer la variable
void setup(){
   ledPin = 13; //Initialiser la variable avec valeur 13
                // …
}
```

Vous auriez très bien pu aussi opérer sans variable et entrer la valeur `13` partout dans les instructions `pinMode` et `digitalWrite`. Seulement, il y a un inconvénient. Si vous décidez plus tard de changer de broche, vous devez revoir tout le code du sketch pour procéder à tous les changements. C'est fastidieux et surtout générateur d'erreurs. Vous risquez d'oublier un endroit quelconque à éditer et d'avoir ensuite un problème. Ce ne pas très grave dans ce court exemple mais si le sketch était plus long, vous sauriez combien ce rudiment de programmation est vraiment utile. Faisons donc bien les choses dès le début. Tout va bien jusqu'ici ? La fonction `setup` est appelée une seule fois au début pour démarrer le sketch et la broche numérique `13` est programmée en tant que sortie. Revoyons pour ce faire l'instruction `pinMode`.

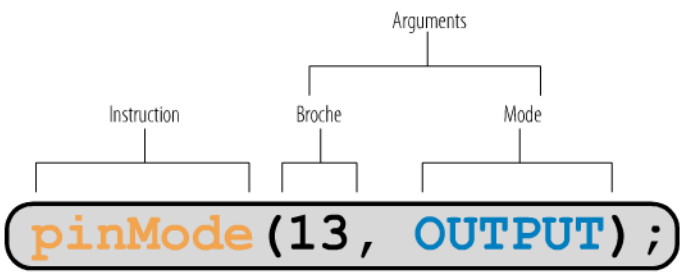


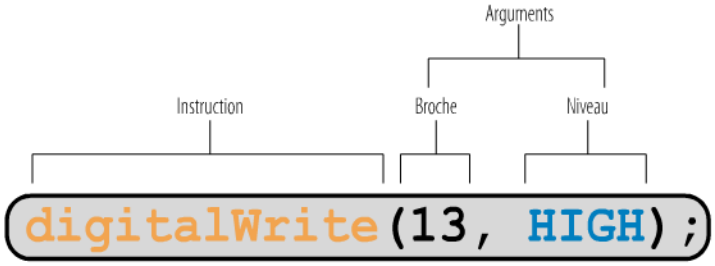
Figure 1-3
L'instruction `pinMode` avec ses arguments

Elle présente deux arguments, le premier pour la broche ou le port à configurer et le deuxième pour définir son comportement comme entrée ou sortie. Mettons que vous vouliez raccorder une LED et que vous ayez besoin pour cela d'une broche de sortie. L'instruction exige deux arguments numériques, le deuxième étant une constante avec une certaine valeur qui définit le mode relatif à la direction des informations. Derrière la constante `OUTPUT` se cache la valeur `1`. Que dites-vous donc par conséquent de l'instruction suivante :

```
pinMode(13, 1);
```

On a du mal à comprendre ce qui se passe. La forme initiale est bien plus explicite et on sait tout de suite de quoi il s'agit. Il en va de même pour l'instruction `digitalWrite`, qui présente également deux arguments.

Figure 1-4 ▶
L'instruction `digitalWrite` avec ses arguments



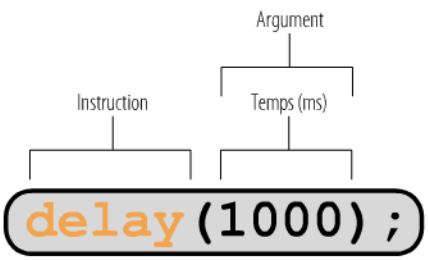
On trouve ici aussi une constante dont le nom est `HIGH`, censée servir d'argument pour un niveau `HIGH` sur la broche 13. Elle est équivalente à la valeur numérique `1`. Vous trouverez les valeurs correspondantes dans le tableau 1-3.

Tableau 1-3 ▶
Constantes avec valeurs numériques correspondantes

Constante	Valeur	Explication
`INPUT`	`0`	Constante pour l'instruction `pinMode` (programme la broche en tant qu'entrée)
`OUTPUT`	`1`	Constante pour l'instruction `pinMode` (programme la broche en tant que sortie)
`LOW`	`0`	Constante pour l'instruction `digitalWrite` (met la broche au niveau `LOW`)
`HIGH`	`1`	Constante pour l'instruction `digitalWrite` (met la broche au niveau `HIGH`)

La dernière instruction utilisée, `delay`, sert à la temporisation. Elle interrompt l'exécution du sketch pour un temps correspondant à la valeur donnée qui exprime la durée en millisecondes (ms).

Figure 1-5 ▶
L'instruction `delay`



La valeur `1 000` signifie une attente de 1 000 ms précisément, soit 1 seconde, avant de continuer.

Entrons maintenant plus avant dans le sketch. La fonction `loop`, ici c'est une boucle sans fin, démarre. Voici les différentes étapes de travail.

1. Allumez la LED de la broche 13.
2. Attendez une seconde.

3. Éteignez la LED de la broche 13.
4. Attendez une seconde.
5. Revenez au point 1 puis, recommencez.

Schéma

Le schéma rend les choses plus claires.

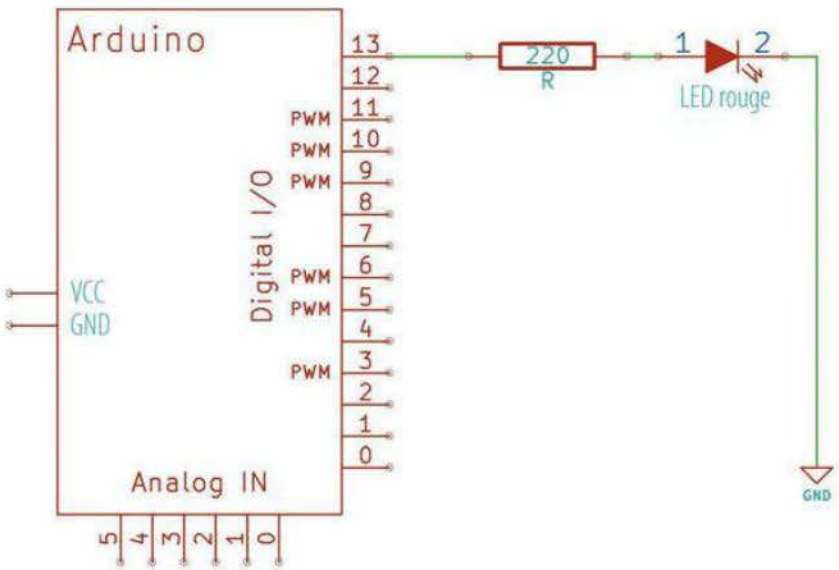


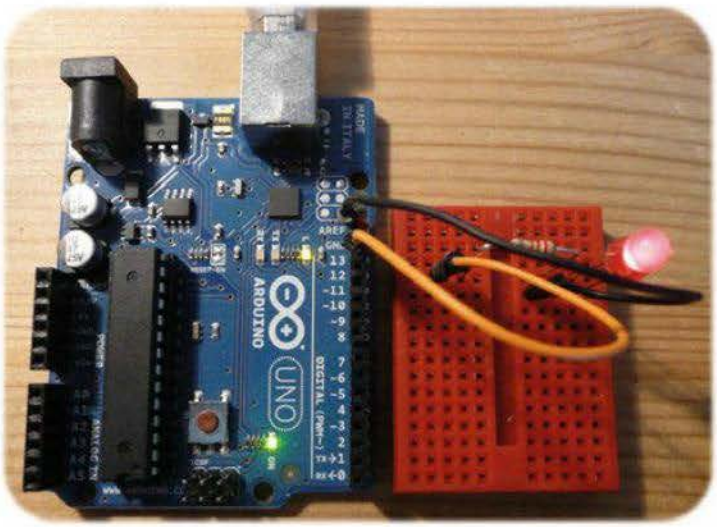
Figure 1-6
Carte Arduino avec une LED sur la broche 13

L'anode (ici, l'électrode 1) de la LED est reliée à la broche 13 via la résistance série, tandis que l'autre extrémité ou cathode (ici, l'électrode 2) de la LED est reliée à la masse de la carte Arduino.

Réalisation du circuit

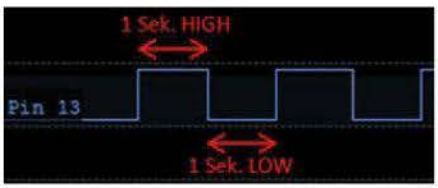
La réalisation du circuit est simple. Toutefois, veillez à ce que la polarité de la LED soit correcte, sinon vous n'obtiendrez qu'une belle LED éteinte. La LED soudée sur la carte clignotera quand même. Une LED mal polarisée n'abîmera rien mais mieux vaut bien faire les choses.

Figure 1-7 ▶
LED clignotante servant de phare pour notre premier sketch



C'est difficile à voir mais en regardant bien, on s'aperçoit que la LED « onboard » clignote en même temps que la LED reliée extérieurement. Les LED sont censées commencer à clignoter aussitôt après la transmission réussie sur la carte. Voyons maintenant de plus près le déroulement chronologique. La LED clignote toutes les deux secondes.

Figure 1-8 ▶
Chronogramme



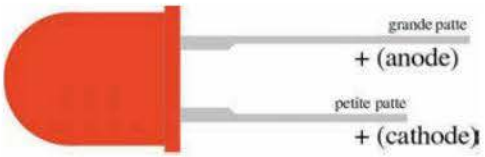
Attention !

On trouve sur Internet des schémas de circuits où une diode électroluminescente est branchée directement entre masse et broche 13. Les deux fiches femelles se trouvant l'une à côté de l'autre, côté broches numériques, il est très aisé de brancher une LED. Je vous mets expressément en garde contre cette variante car la LED est utilisée sans résistance série. Ce n'est pas tant pour la LED mais bel et bien pour votre microcontrôleur que je m'inquiète. J'ai mesuré une fois que l'intensité du courant atteignait 60 mA. Cette valeur est de 50 % au-dessus du maximum et donc assurément trop élevée. Rappelez-vous que le courant admis par une broche numérique du microcontrôleur est de 40 mA au maximum.

Problèmes courants

Si la LED ne s'allume pas, plusieurs choses peuvent en être la cause ainsi que nous l'avons déjà dit.

- La LED peut avoir été mal polarisée. Rappelez-vous les deux différentes connexions d'une LED qui sont l'anode et la cathode.

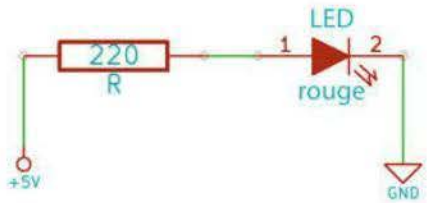


- La LED a peut-être été grillée par une surtension lors des montages précédents. Testez-la avec une résistance sur une source d'alimentation de 5 V.
- Vérifiez les fiches de la barrette de raccordement qui sont reliées à la LED ou à la résistance série. S'agit-il bien de GND et de la broche 13 ?
- Vérifiez le sketch que vous avez entré dans l'éditeur de l'IDE. Peut-être avez-vous oublié une ligne ou commis une erreur ou peut-être le sketch a-t-il mal été transmis ?
- Si la LED qui se trouve sur la carte clignote, le LED branchée doit elle aussi clignoter. Le sketch fonctionne dans ce cas correctement.

Qu'avez-vous appris ?

- Vous avez appris à déclarer et initialiser des variables globales en une ou plusieurs lignes.
- Vous avez déterminé le sens de transmission des données pour une certaine broche comme `OUTPUT` au moyen de l'instruction `pinMode`, si bien que vous avez pu envoyer, au moyen de l'instruction `digitalWrite`, un signal numérique (`HIGH` ou `LOW`) à la sortie où la LED est branchée.
- Vous avez créé un temps d'attente dans l'exécution du sketch au moyen de l'instruction `delay`, si bien que la LED restait allumée ou éteinte un certain temps.
- Vous savez que pour utiliser une LED, il faut une résistance série dimensionnée en conséquence. Vous trouverez ci-après un

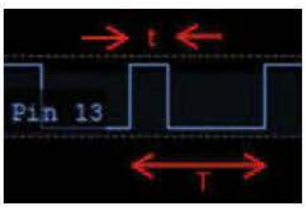
schéma de connexion d'une LED avec une résistance de 220 ohms en série.



Exercice complémentaire

Dans notre premier exercice, je vous propose de modifier le sketch de telle sorte que les temps durant lesquels la LED est allumée ou éteinte soient déterminés par deux variables, afin de pouvoir changer facilement le rapport cyclique. Ce dernier peut être défini, dans le cas d'une suite périodique d'impulsions, par le rapport entre la durée d'une impulsion et la durée de la période. Le résultat est la plupart du temps exprimé en pourcentage. Le chronogramme de la figure 1-9 montre les différentes durées pour *t* ou *T*.

Figure 1-9 ▶
Chronogramme d'une impulsion



t = durée de l'impulsion

T = durée de la période

La formule pour calculer le rapport cyclique est la suivante :

$$Rapport\ cyclique = \frac{t}{T}$$

Programmez le sketch de telle sorte que la LED reste allumée pendant 500 ms et éteinte pendant 1 s. Le rapport cyclique est alors calculé comme suit :

$$Rapport\ cyclique = \frac{500\ \text{ms}}{1\ 500\ \text{ms}} = 0{,}33$$

Ceci correspond à un rapport cyclique de 33 %. Par rapport à la durée complète de la période, la LED est allumée pendant 33 % du temps.