# Chapitre 18 – Comment gérer les signaux d'entrée et les signaux de sortie

## Signaux d'entrée et signaux de sortie

Nous avons vu au chapitre 15 que le module Uno reçoit des signaux de la part du réseau et envoie les signaux qu'il élabore au réseau. Le module Uno travaille en 5 V ; les signaux d'entrée doivent respecter cette tension. Il sera donc parfois nécessaire d'adapter les signaux récupérés en amont qui peuvent être dans une tension différente. Les signaux de sortie sont également en 5 V et sont souvent trop faibles pour produire une action sur le réseau ; ils devront alors être amplifiés car les sorties du module Arduino ne délivrent que 40 mA au maximum.

### Signaux analogiques et signaux numériques

Un signal analogique est un signal qui varie de façon continue en augmentant ou en diminuant ; son amplitude devant être au plus égale à 5 V, ce signal va donc varier entre 0 et 5 V de façon continue. Un signal numérique ne peut avoir que deux valeurs possibles : 0 V (état bas ou LOW) et 5 V (état haut ou HIGH) et il passe de l'une à l'autre très rapidement. La figure 18.1 montre la différence entre signal analogique et numérique. Attention, le module Uno n'admet aucune tension négative et lui en proposer pourrait l'endommager.

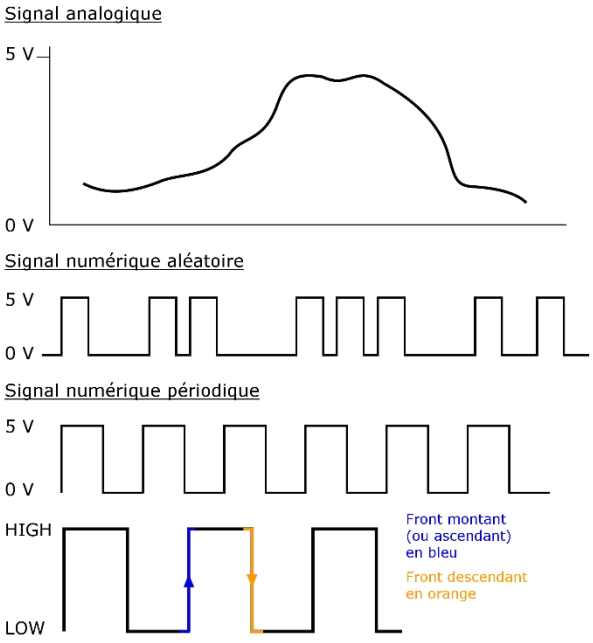


*Figure 18. 1*

La figure 18.1 montre également ce qu'on appelle front montant ou un front descendant dans un signal numérique.

### Connecteurs sur le module Uno

Le module Uno possède 14 entrées-sorties numériques, notées E/S (ou I/O en anglais pour Input/Output) ; nous pouvons les voir entourées de rouge sur la figure 18.2. On voit qu'elles sont repérées par le mot « Digital » qui signifie numérique et elles sont numérotées de 0 à 13 (ce qui fait bien 14 en tout). Il faut se rappeler qu'en informatique, **on commence toujours à compter par le chiffre 0** (comme il existe, ce serait dommage de s'en priver). Les sorties repérées par le signe ~ (tilda) sont les sorties sur lesquelles on peut faire de la PWM (voir le chapitre 12) ; elles sont au nombre de 6 (les sorties 3, 5, 6, 9, 10 et 11). Les E/S 0 et 1 sont aussi appelées Rx et Tx et elles permettent de recevoir

(R) ou de transmettre (T) des signaux en provenance de ou vers l'ordinateur lorsque le module Uno est branché avec le câble USB ; **nous éviterons, si possible, de les utiliser dans nos programmes**.

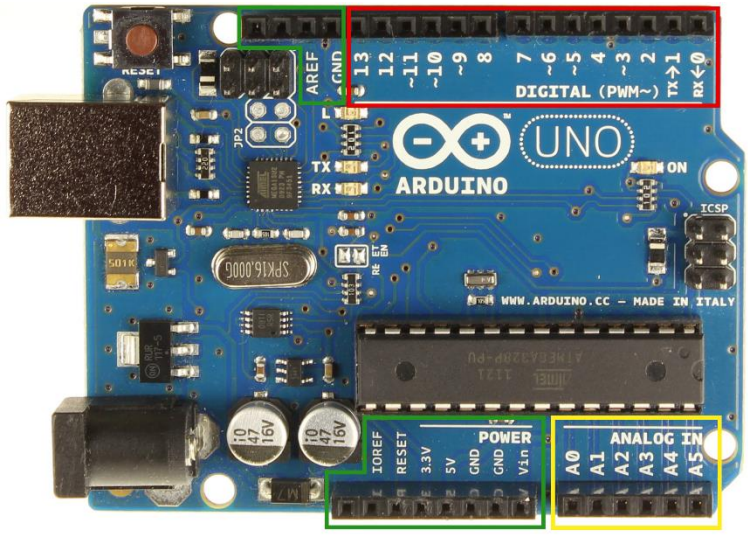


*Figure 18. 2*

Toutes ces E/S sont à la fois des entrées et des sorties, mais pas les deux en même temps. Comme entrées, elles vont recevoir des signaux de l'extérieur (le réseau par exemple), comme sorties, elles vont émettre des signaux vers l'extérieur. Il est donc **nécessaire de déclarer les E/S** soit en entrée, soit en sortie (on dit « initialiser »). Dans le langage Arduino, la fonction à utiliser pour initialiser les E/S est pinMode (nous y reviendrons en temps voulu).

Comme le nom l'indique, **les 14 E/S numériques ne peuvent recevoir ou émettre que des signaux numériques**.

Six autres entrées servent à recevoir des signaux analogiques ; ce sont les **entrées analogiques** numérotées A0 à A5 qui sont entourées en jaune sur la figure 18.2 et elles sont repérées par les mots « Analog In » qui signifient entrée analogique. Le microcontrôleur comporte dans son boitier un convertisseur analogique-numérique capable de convertir ce signal analogique (qui doit être compris entre 0 et 5 V) en une valeur numérique comprise entre 0 et 1023.

Chacune de ces six entrées analogiques peut aussi être utilisée en E/S numérique (c'est un choix à faire dès le départ du programme) ; dans ce cas, elle doit être initialisée en entrée ou en sortie par la fonction pinMode.

Tous les signaux provenant de l'extérieur ou allant vers l'extérieur **doivent être reliés à ces connecteurs et à ceux-là uniquement**. Les connecteurs entourés en vert sur la figure 18.2 servent à autre chose (récupérer des tensions ou la masse GND par exemple).

Communication réseau-microcontrôleur

Dans le chapitre 15, nous avons vu l'architecture d'un système de gestion de réseau par un microcontrôleur ou par un module Arduino (voir la figure 15.3). Des capteurs disposés sur le réseau envoient des signaux qui sont mis en forme par l'interface de signaux d'entrée, puis ceux-ci sont exploités par le microcontrôleur qui va alors, dans certaines circonstances, agir sur le réseau. Pour cela, il va élaborer des signaux de sortie qui seront amplifiés pour agir sur des organes du réseau (aiguilles, barrières de PN, signalisation lumineuse, alimentation des voies, etc.).

Nous aurons donc à étudier quels sont les capteurs à disposer sur le réseau, quels types de signaux ces capteurs vont délivrer et comment mettre en forme ces signaux pour qu'ils soient compatibles avec le microcontrôleur. Ceci constitue la partie électronique de l'interface des signaux d'entrée.

À partir de là, le microcontrôleur reçoit des informations qu'il nous faudra apprendre à traiter pour décider d'actions à mener sur le réseau. Ceci constitue la partie informatique de l'application qu'on veut créer.

Enfin, le microcontrôleur envoie des signaux au réseau en positionnant ses sorties numériques, mais le courant qu'elles délivrent étant trop faible, il est nécessaire de l'amplifier. C'est à nouveau la partie électronique de l'interface des signaux de sortie.

Il faut se rappeler qu'un microcontrôleur est à la fois un composant électronique et un composant informatique. Les deux côtés sont liés dans une application et il est nécessaire de bien comprendre l'interaction entre les deux pour bien programmer le microcontrôleur. Heureusement pour nous, la partie électronique à mettre autour d'un module Arduino est souvent réduite à sa plus simple expression (par exemple, un simple transistor). Heureusement pour nous aussi, la partie informatique ne demande qu'un petit apprentissage si on se limite aux instructions du « langage » Arduino.

À retenir sur les signaux d'entrée et les signaux de sortie :

- La différence entre un signal analogique et un signal numérique.
- Le module Uno dispose de 14 entrées-sorties numériques et 6 entrées analogiques qui peuvent être utilisées en tant qu'E/S numériques.
- Les signaux doivent respecter une tension maximum de 5 V, que ce soit en analogique ou en numérique.
- Les E/S numériques doivent être initialisées en entrée ou bien en sortie dans le programme en fonction de ce qu'on veut leur faire faire.
- Les signaux de sortie sont trop faibles pour être utilisés ainsi et doivent être amplifiés.

## Interfaçage simple des entrées et des sorties

Afin de pouvoir commencer à utiliser les entrées et sorties de notre module Uno, nous allons voir comment on peut les interfacer très simplement. Bien entendu, dans la suite du cours, nous étudierons d'autres possibilités d'interfaçage avec des capteurs pour les entrées et avec d'autres périphériques (moteur, servo, écran) pour les sorties.

### Entrée

Une façon simple d'exploiter une entrée est de la relier à un bouton-poussoir (B/P) ; la figure 18.3 montre comment raccorder un simple bouton poussoir à une entrée numérique.

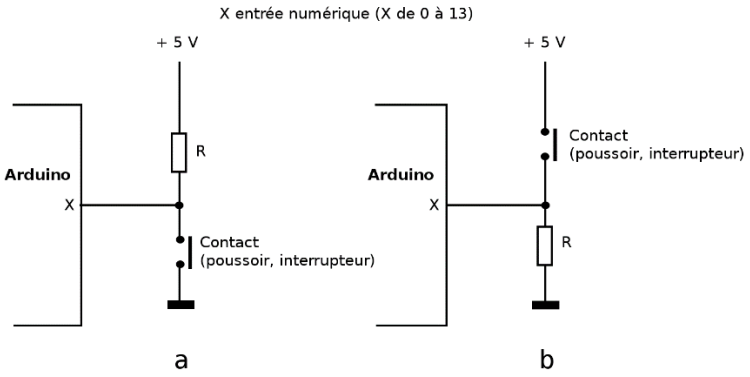


*Figure 18. 3*

**En a)** le microcontrôleur voit une tension de 5 V sur son entrée tant que le bouton poussoir n'est pas appuyé (il n'y a pas de chute de tension à travers la résistance car pratiquement aucun courant n'entre dans le microcontrôleur du fait que les entrées sont à haute impédance). Lorsque le bouton poussoir est appuyé, l'entrée est reliée au 0 V.

**En b)** le microcontrôleur voit une tension de 0 V sur son entrée tant que le bouton poussoir n'est pas appuyé (il n'y a pas de chute de tension à travers la résistance car pratiquement aucun courant n'entre dans le microcontrôleur du fait que les entrées sont à haute impédance). Lorsque le bouton poussoir est appuyé, l'entrée est reliée au 5 V.

Dans le programme, l'entrée-sortie concernée sera initialisée en entrée grâce à la fonction pinMode et l'état de cette entrée sera obtenu grâce à la fonction digitalRead. Lors de son appel, la fonction digitalRead retournera soit HIGH soit LOW **en fonction de la position du bouton poussoir (appuyé ou non) et de la façon de le monter**. Le bouton poussoir étant appuyé, la fonction digitalRead retournera LOW en a) et HIGH en b).

Les résistances R de la figure 18.3 sont appelées **résistance de pull-up** dans le cas a) (elle tire l'entrée vers le haut) et **résistance de pull-down** dans le cas b) (elle tire l'entrée vers le bas). Une valeur de 10 kOhm convient très bien.

Le microcontrôleur ATmega328P contient des résistances de pull-up dont la valeur est de l'ordre de 20 à 50 kOhms qu'on peut activer par programmation. La fonction à utiliser pour activer la résistance de pull-up sur une entrée est :

pinMode(pin, INPUT_PULLUP) ;

Dans ce cas, le bouton poussoir est disposé **entre la sortie et la masse** (ce qui permet l'économie de la résistance), et **lorsqu'il est appuyé, la fonction digitalRead retourne LOW**.

Le microcontrôleur ATmega328P ne contient pas de résistances pull-down, mais certains microcontrôleurs en sont pourvus.

Sortie

Une façon simple d'exploiter une sortie est de la relier à une LED par l'intermédiaire d'une résistance qui sert à limiter le courant traversant la LED. La figure 18.4 montre qu'il y a deux montages possibles.

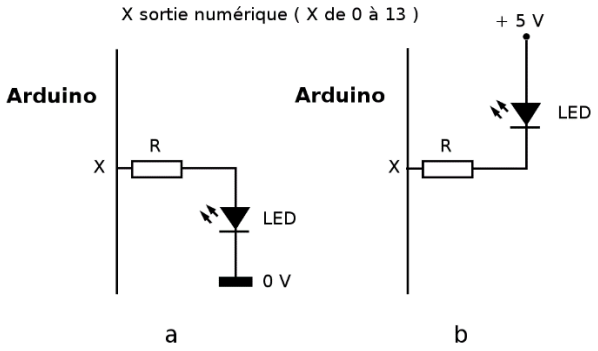


*Figure 18. 4*

**Dans le cas a)**, pour que la LED soit allumée, il faut que la sortie numérique X soit à l'état haut (5 V). **Dans le cas b)**, il faut qu'elle soit à l'état bas (0 V).

Dans le programme, l'entrée-sortie concernée sera initialisée en sortie grâce à la fonction pinMode et l'état de cette entrée sera positionné grâce à la fonction digitalWrite :

digitalWrite(pin,value); ou encore digitalWrite (pin, value) ; (ce qui est plus lisible vu que les espaces ne sont pas pris en compte par le compilateur (le programme qui transforme ce que vous avez écrit dans le langage machine du microcontrôleur ATmega328P). Ne pas oublier le point-virgule à la fin de la ligne d'instruction ni la majuscule à Write.

pin est le numéro de la broche et value est soit HIGH, soit LOW. Si c'est HIGH, la broche est à 5 V, si c'est LOW, elle est à 0 V.

Le calcul de la résistance se fait avec la loi d'Ohm en tenant compte de la chute de tension dans la LED de façon à limiter le courant (limitation de la LED mais aussi du module Uno comme on l'a expliqué dans les chapitres précédents).

Dans le cas a), le module Uno (c'est-à-dire le microcontrôleur) fournit du courant, dans le cas b), il absorbe du courant.

Comment utiliser pinMode ?

Avant de décrire les entrailles du microcontrôleur, faisons un petit rappel sur cette fonction pinMode. Déjà, comme vous le constatez, cette fonction **s'écrit avec un M majuscule** ; ceci est important. La syntaxe à adopter est la suivante :

pinMode(pin,mode);

Ne pas oublier le point-virgule à la fin de chaque instruction. De plus, vous pouvez, pour plus de lisibilité, mettre des espaces car ils sont ignorés par le compilateur (le programme qui transforme ce que vous avez écrit dans le langage machine du microcontrôleur ATmega328P). On peut donc écrire :

pinMode (pin, mode) ;

pin est le numéro de la broche que vous voulez initialiser.

mode est le mode possible : entrée c'est INPUT, sortie c'est OUTPUT.

Il existe un troisième mode qui sert à initialiser la broche en entrée et à utiliser les résistances de pull-up : INPUT_PULLUP.

À retenir sur l'interfaçage des entrées-sorties :

- Le rôle des résistances de pull-up ou de pull-down.
- Les résistances de pull-up internes à l'ATMega328P.
- La façon d'initialiser l'entrée pour en bénéficier.
- Les deux façons possibles d'interfacer une LED avec le microcontrôleur.
- La présence d'une résistance en série avec la LED pour limiter le courant.