Lecture de bouton

D.Moreaux

21 octobre 2018

1 Pull-Up/Pull-Down

Une entrée utilisée de l'arduino ne peut pas rester *flottante* c'est à dire, non connectée. Une entrée flottante captera les différents parasites ambiants qui détermineront son niveau.

Pour éviter cela, on forcera un niveau (haut ou bas) à l'aide d'une résistance quand le bouton est inactif et le bouton permettra de forcer l'autre niveau quand il sera activé. On parlera de résistance de *pull-up* si elle force un niveau haut (+5V) et de résistance de *pull-down* si elle force un niveau bas (0V ou GND).

Quand le bouton est activé, les bornes d'alimentation sont reliées au travers de la résistance. Il est donc nécessaire d'utiliser une résistance assez forte que pour éviter un courant trop élevé (une valeur de l'ordre de 10kΩ par exemple).

La plupart des microcontrôleurs (dont le ATMega qui équipe l'Arduino) disposent de résistances de pull-up ou de pull-down internes. Certains n'en disposent que sur certaines broches ou uniquement des pull-up, certains proposent pull-up et pull-down sur toutes les broches. Le ATMega propose des pull-up sur toutes ses entrées.

Pour activer une broche en entrée, on utilisera

```
pinMode(3,INPUT);
pinMode(4,INPUT_PULLUP);
```

La première forme active la broche en entrée flottante, il sera nécessaire soit d'utiliser une résistance de pull-up ou pull-down externe, soit de brancher l'entrée sur un circuit qui intègre les dites résistances.

La seconde forme active la broche en entrée avec une résistance de pull-up interne à l'Arduino. Ce sera cette forme que l'on utilisera avec des boutons.

2 Connexion

Comme la broche est munie d'une résistance de pull-up, elle est par défaut reliée au +5V. Il faudra donc que le bouton amène la broche à l'autre valeur, à savoir le 0V.

On connectera donc le bouton entre la broche de l'arduino et le 0V. Attention à s'assurer que la broche est bien soit configurée en entrée, soit configurée en sortie avec un signal bas. Si la broche est une sortie à l'état haut, en poussant sur le bouton, on court-circuite la sortie et on grille l'arduino.

Une précaution qui peut être prise est de disposer d'un sketch qui force toutes les broches en entrée et l'utiliser avant de ranger l'arduino (ou de changer de montage). Un exemple de tel sketch ce trouve ci-dessous :

```
void setup() {
  int i;
  for(i=0;i<20;i++) { /* D0 - D13, A0=D14 - A5=D19 */
    pinMode(i,INPUT);
  }
}

void loop() {

}
```

3 Lecture du bouton

On pourra lire l'état de la broche à l'aide de `digitalRead(n)` qui retournera une valeur non nulle si le signal est haut et nulle si le signal est bas.

Comme par défaut, la broche est tirée vers le +5V par le pull-up, une valeur non nulle indiquera que le bouton est relaché et une valeur nulle que le bouton est enfoncé.

Pour tester si le bouton est enfoncé, on utilisera de préférence

```
b=digitalRead(BUT);
if (!b) {
  ...
}
```

ou `if (b) {` pour tester si le bouton est relaché. On pourra aussi comparer l'état du bouton avec l'état de **ce même bouton** sauvé dans une

variable [1]

4 Les rebonds

Si le bouton était parfait, le signal passerait de 1 à 0 quand on l'enfonce et de 0 à 1 quand on le relâche.

Malheureusement, la mécanique n'est pas parfaite. Quand on enfonce le bouton, on a un effet de rebond (un peu comme une balle qui touche le sol, les rebonds sont de moins en moins forts et le nombre dépendra de la nature de la balle et du sol). Lors de ces rebonds, le signal passe de 0 à 1 et de 1 à 0 jusqu'à ce que les rebonds soient terminés.

Si il existe des moyens pour supprimer les rebonds au niveau électronique [2] ils ont le défaut de demander des composants supplémentaire et donc d'engendrer des coûts supplémentaires. On cherchera donc à neutraliser les rebonds au niveau logiciel.

La méthode est très simple : on attend que le niveau soit stable depuis un certain temps avant de valider le changement.

```
int butoldread;
int butoldtime;
int butstate;

void loop() {
  int but;

  but=digitalRead(BUT);
  if (but != butoldread) {
    butoldtime=millis();
    butoldread=but;
  }
  if (millis() - butoldtime > 20)  { /* Bouton stable depuis 20ms */
    butstate=but;
  }
}
```

La variable `butstate` contiendra le dernier état stable du bouton. Si le bouton est de très mauvaise qualité, on peut augmenter le délai de 20ms. Si

1. on évitera de comparer avec des valeurs 0 ou 1
2. condensateur, porte en *trigger de Schmidt*, ...

au contraire il provoque peu de rebonds, on peut le diminuer pour améliorer le temps de réaction.

5 Le changement d'état

Jusqu'à maintenant, on s'est intéressé à connaître l'état instantanné d'un bouton. Souvent, on sera plus intéressé par les changements d'état (appui sur le bouton, relâchement du bouton).

Cela peut être fait aisément en comparant l'état du bouton avec l'état précédent. Combiné au code précédent, on comparera l'état stable avec l'état stable précédent.

```
int butoldread;
int butoldtime;
int butoldstate;

void loop() {
  int but;

  but=digitalRead(BUT);
  if (but != butoldread) {
    butoldtime=millis();
    butoldread=but;
  }
  if (millis() - butoldtime > 20)  { /* Bouton stable depuis 20ms */
    if (!but && butoldstate) { /* Le bouton est enfoncé et était relaché */
...
    }
    butoldstate=but;
  }
}
```

On peut évidemment changer le dernier if pour détecter le relachement du bouton. Il est nécessaire de faire tous les tests avant de mettre à jour `butoldstate`.