Chapitre 20 - Les fonctions électroniques en électronique programmable

La meilleure façon d'apprendre, c'est encore de pratiquer. Voici quelques exercices à réaliser qui vous démontreront que l'électronique programmable peut remplacer l'électronique câblée. Ces exercices seront également l'occasion de découvrir la programmation, les fonctions, les variables. Ce que vous ferez là pourra ensuite vous servir pour votre réseau de train miniature.

Commençons par réaliser le petit montage de la figure 20.1 avec un module Arduino et deux boutons poussoirs reliés à la masse, **ce qui signifie qu'il faut activer les résistances de pull-up du microcontrôleur**.

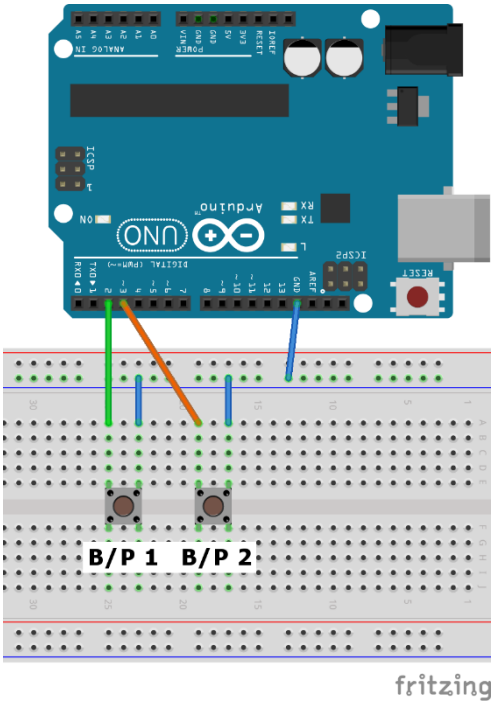

Figure 20. 1

Le B/P 1 est relié à l'entrée 2 et le B/P 2 à l'entrée 3. Nous utilisons la LED du module Uno reliée à la sortie 13.

Fonctionnement d'un B/P avec résistance pull-up interne

Ouvrez le programme DigitalInputPullup donné en exemple dans l'IDE.

Fichiers > Exemples > 02.Digital > DigitalInputPullup

Dans la fonction setup, supprimez la ligne Serial.begin(9600) ; et dans la fonction loop la ligne Serial.println(sensorVal) ; car l'utilisation du moniteur ne sert pas à grand-chose. Enregistrez le programme sous un nouveau nom avec Fichier > Enregistrer sous, en ajoutant par exemple vos initiales au nom initial.

La fonction setup initialise la ligne 2 en entrée avec la résistance de pull-up interne et la ligne 13 en sortie (celle qui est reliée à la LED du module).

Le programme allume la LED quand on appuie sur le B/P.

La fonction principale lit l'entrée numérique 2 et place le résultat dans la fonction sensorVal (sensorVal vaut 0 ou LOW si B/P est appuyé, et 1 ou HIGH si B/P n'est pas appuyé).

Le programme exécute un test (if … else) sur la valeur de sensorVal ; si c'est HIGH, on éteint la LED avec le digitalWrite (13, LOW), si c'est LOW (autrement, soit else) on l'allume avec digitalWrite (13, HIGH).

En fait, le test n'est pas nécessaire ; il suffit de remarquer qu'on écrit sur la sortie l'inverse de ce qu'on a lu sur l'entrée.

Remplacer le contenu de la fonction loop par cette simple instruction :

digitalWrite (13, !digitalRead(2)) ;

Le résultat est le même. En effet, **le point d'exclamation équivaut à NOT** ou INVERSE, et on n'a pas utilisé la variable intermédiaire sensorVal.

Fonction multivibrateur astable (oscillateur)

Le multivibrateur astable a été vu auparavant, lorsque nous avons essayé le programme Blink qui réalise un oscillateur.

Fonction monostable

Lors d'un bref appui sur le B/P 1, la LED doit s'allumer pendant un temps plus long (ici deux secondes). Durant ces deux secondes, un nouvel appui sur le B/P 1 est ignoré : on obtient un monostable **non redéclenchable**.

Voici le programme :

Fonction_monostable

```
void setup() {
  //configure pin 2 et pin 3 comme entrée avec pull-up interne
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop() {
  while (digitalRead(2) == HIGH) {// Ne rien faire
    }
    // Ici, bouton appuyé lu comme LOW
  digitalWrite (13, HIGH);
  delay (2000);
  digitalWrite (13, LOW);
}
```

Fonction bistable

Un appui sur le B/P 1 provoque l'allumage de la LED (entrée Set) et un appui sur le B/P 2 provoque son extinction (entrée Reset). La fonction obtenue est un bistable analogue à une bascule RS.

Voici le programme :

Fonction_bistable

```
void setup() {
  //configure pin 2 et pin 3 comme entrée avec pull-up interne
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(13, OUTPUT);
  digitalWrite (13, LOW);
}

void loop() {
  while (digitalRead(2) == HIGH) {// Ne rien faire
    }
    // Ici, bouton 2 appuyé lu comme LOW
  digitalWrite (13, HIGH);
  while (digitalRead(3) == HIGH) {// Ne rien faire
  }
  // Ici bouton 3 appuyé lu comme LOW
  digitalWrite (13, LOW);
}
```

Fonction OR

La LED s'allume si on maintient appuyé le B/P 1 ou le B/P 2 ou les deux en même temps. Nous avons réalisé la fonction logique OR avec une ligne de code. L'opérateur OR (ou OU) est le ||.

Voici le programme :

Fonction_OR

```
void setup() {
  //configure pin2 as an input and enable the internal pull-up resistor
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite (13, !digitalRead(2) || !digitalRead(3));
}
```

Fonction AND

La LED ne s'allume que si les deux B/P sont appuyés en même temps.

Voici le programme :

Fonction_AND

```
void setup() {
  //configure pin2 as an input and enable the internal pull-up resistor
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite (13, !digitalRead(2) && !digitalRead(3));
}
```

Fonction XOR

Cette fonction OU Exclusif a été définie dans le chapitre 9 : la LED doit s'allumer si on appuie sur le B/P 1 ou sur le B/P 2 mais pas si on appuie sur les deux B/P en même temps.

Voici le programme :

Fonction_XOR

```
void setup() {
  //configure pin 2 et pin 3 en entrée avec pull-up interne
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite (13, (!digitalRead(2) && digitalRead(3)) || (digitalRead(2) && !digitalRead(3)));
}
```

Rappelez-vous que nous avions démontré au chapitre 9, que :

A XOU B = (A ET NON B) OU (NON A ET B)

Ce qui allume la LED dans le montage avec pull-up interne est l'inverse de ce qui est lu pour le B/P. Or l'inverse de l'inverse (ou NON NON) s'annule (**deux points d'exclamation s'annulent**). Ce qui explique la ligne de code de la fonction loop.

Autres possibilités

Vous pouvez modifier le montage de la figure 20.1 et rajouter d'autres LED ou d'autres B/P ; vous en savez assez au sujet des fonctions digitalRead et digitalWrite pour fabriquer vos propres fonctions et vous entrainer à la programmation. D'autres fonctions du langage Arduino seront expliquées ultérieurement.

À retenir sur les fonctions électroniques :

- La façon d'activer les résistances de pull-up internes.
- La boucle while.
- L'opérateur de comparaison ==
- L'opérateur NOT qui s'écrit !
- L'opérateur AND qui s'écrit &&
- L'opérateur OR qui s'écrit ||
- Toutes les fonctions électroniques (astable, monostable, bistable, etc.) ou logiques (AND, OR, NOT, XOR) peuvent être obtenues avec peu de lignes de programme.