Module SCI

Ecole Nationale Supérieure d'Informatique d'Alger (Ex. ini)

Compte rendu TP3

07 Avril 2022

Introduction

Dans ce TP nous nous intéressons aux interruptions et PWM.

Objectifs

1. **Partie théorique :** Compréhension et bases des concepts liés aux programmes permettant de contrôler des interruptions.
2. **Partie pratique :** Utiliser un bouton poussoir en mode interruption et contrôler une sortie analogique, en appliquant le concept sur la luminosité d'une LED.

Partie théorique

1. La ligne GPIO avec interruptions :

Le premier paramètre de attachInterrupt() est un numéro d'interruption. Normalement, vous devez utiliser digitalPinToInterrupt(pin) pour traduire la broche numérique réelle en numéro d'interruption spécifique. Par exemple, si vous vous connectez à la broche 3, utilisez digitalPinToInterrupt(3) comme premier paramètre pour attachInterrupt().

MKR pin avec interruptions : 0, 1, 4, 5, 6, 7, 8, 9, A1, A2

2. Structure d’un code avec interruptions :

La plupart du temps, si l'on veut tester l'état d'un capteur, nous interrogeons l'entrée concernée au travers d'une boucle 'for', par exemple.

Cette manière de faire peut, être couteuse en énergie car le processeur boucle à toute vitesse dans une course folle, à ne rien faire. Et si l'on teste plusieurs capteurs, le problème est multiplié d'autant. De plus, la structure du programme n'est pas vraiment optimisée.

L'idée, qui ne date pas d'hier, c'est de faire en sorte que le capteur envoie un signal sur une entrée dédiée du processeur. Le processeur arrête net ce qu'il était en train de faire, exécute la fonction de l'interruption, puis reprend sa tâche.

Facile à dire ! Et bien oui, et en plus, c'est facile à faire.

Voici un exemple très minimaliste ou le capteur, un simple interrupteur, allume ou éteint la Led notée L (située au milieu de votre Arduino Uno), à chaque fois que l'on appuie dessus. Pour information, cette Led est reliée à l'E/S 13 de l'Arduino.

Cette interrupteur, placé sur une BreadBoard, est relié à la masse au travers d'une résistance de 10k. Ainsi lorsque l'interrupteur est ouvert, l'entrée sera à 0.

Les entrées spécifiques aux interruptions sont numérotées 0, 1, 4, 5, 6, 7, 8, 9, A1, A2

Interrupt sub-routine (ISR) est la fonction par défaut utilisée pour programmer les interruptions. Elle ne prend aucun argument en entrée et elle ne retourne aucun résultat. En revanche, il faut bien préciser le vecteur d’interruption. Chaque interruption dispose d’un vecteur qui lui est propre. La carte Arduino Mega

dispose de 57 interruptions, chaque interruption est caractérisée par son vecteur et adresse.

Syntaxe de de définition de la routine d'interruption

```
ISR(Nom_vect)

{

  // Ici le code de votre interruption

}
```

```
ISR(INT0_vect)

{

  // Programme INT0

}

ISR(INT1_vect)

{

  // Programme INT1

}



ISR(INT2_vect)

{
```

Figure 01 : Structure globale d'un code à base d'interruptions

Lignes GPIO PWM :

Désigne les pins capables de prendre en sortie une valeur analogique : soit d'écrire une vague PWN dans un des composants.

Pour les Cartes de la famille MKR Wifi, les pin concernées sont les pin numériques 0-8, 10 et 11, ainsi que les pin analogiques A3 et A4, leur fréquence PWM est de 732 Hz.

Simulateurs Arduino :

En fonction de la plate-forme sur laquelle vous travaillez, vous pouvez choisir l'un ou l'autre type de simulateur pour Arduino, car il y a beaucoup de types:

- **En ligne:** ce sont des simulateurs basés sur une interface web que vous pouvez gérer depuis n'importe quelle plateforme avec un navigateur web compatible. Ils sont bons car vous n'avez pas à vous soucier de l'installation, de la mise à jour, etc. Accédez-y et utilisez-le.

- **Hors ligne:** ce sont ceux que vous installez localement, dans ce cas, ils doivent être compatibles avec votre système d'exploitation. Vous pouvez parcourir les sites Web des développeurs pour voir les packages disponibles, les télécharger et les installer.

- **Simulateurs électroniques**: Ce ne sont pas vraiment des simulateurs Arduino en tant que tels, mais ils peuvent vous aider à créer vos schémas, comme Fritzing, ou à avoir une meilleure idée de ce dont vous avez besoin pour votre projet.

Quelques-uns des les meilleurs simulateurs pour Arduino sont:

- **Autodesk Tinker Cad:** c'est une plate-forme en ligne que vous pouvez utiliser à partir de n'importe quel navigateur Web. Il est développé par la célèbre société de logiciels techniques Autodesk et permet des conceptions 3D.

- **Suite de design Porteus:** c'est un logiciel qui peut être installé sur Windows, mais aussi sur Linux et Mac. C'est un logiciel très complet pour la simulation électronique, la modélisation de PCB.

- **Autodesk Aigle:** est une autre alternative à la précédente développée par Autodesk. Un programme de simulation très professionnel et puissant. Il dispose d'un grand nombre d'outils qui le rendent très complet pour les ingénieurs et les utilisateurs avancés.

- **UnoArduSim :** C'est un simulateur gratuit pour Windows qui est très intéressant. il a une bibliothèque de plusieurs composants électroniques communs, mais c'est le plus facile à utiliser que j'ai vu. Il vous permet même d'exécuter le code source pour l'Arduino ligne par ligne pour le débogage.

- **Virtronique:** la société a cette version payante pour Linux et Windows que vous pouvez acheter pour quelques euros. La société de développement a conçu ce logiciel pour qu'il puisse être utilisé par les étudiants et les débutants dans le monde de l'électronique.

Partie pratique

Application à un bouton poussoir avec interruptions :

Code associé :

```
const int LED_PIN = 6;
const int INTERRUPT_PIN = 1;
volatile bool ledState = HIGH;

void setup() {
    pinMode(LED_PIN, OUTPUT);
    pinMode(INTERRUPT_PIN, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), myISR, CHANGE);
}

void loop() {
    digitalWrite(LED_PIN, ledState);
}

void myISR() {
    ledState = !ledState;
```

Figure 02 : code associé

Figure 03 : Assemblage réel et plan de circuit

Application pour un contrôle de l'intensité d'une LED suivant la valeur lu à partir d'un photomètre :

```
1   const int LED_PIN = A1;
2   const int INTERRUPT_PIN = 7;
3   volatile bool ldrValue = 0;
4
5   void setup() {
6       pinMode(LED_PIN, OUTPUT);
7       pinMode(INTERRUPT_PIN, INPUT);
8   }
9
10  void loop() {
11      ldrValue = analogRead(INTERRUPT_PIN);
12      analogWrite(LED_PIN, ldrValue);
13  }
```

Figure 04 : code associé

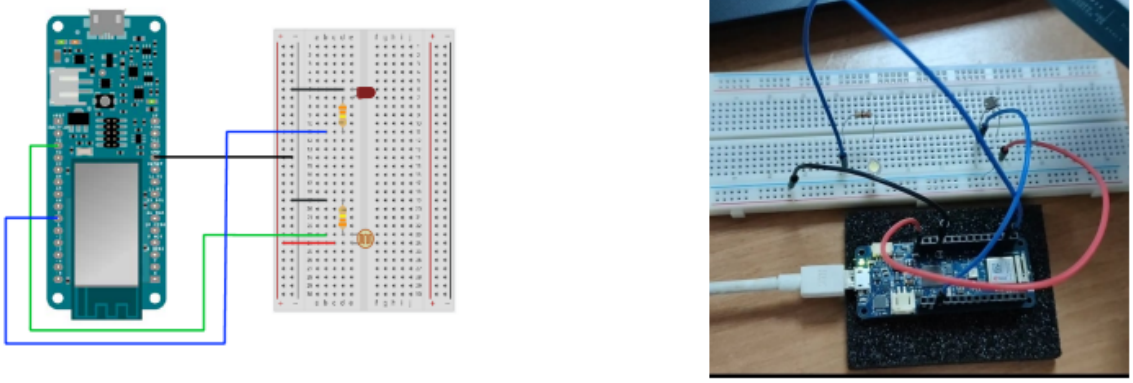

Figure 05 : Assemblage réel et plan du circuit

Utilisation d'un simulateur Arduino en ligne

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
int pushButton = 2;

// the setup routine runs once when you press reset:
void setup() {

  Serial.begin(9600);
  pinMode(pushButton, INPUT);
    pinMode(led, OUTPUT);

}

// the loop routine runs over and over again forever:
void loop() {

  int buttonState = digitalRead(pushButton);
  Serial.println(buttonState);
  if (buttonState == 1) {
      digitalWrite(led, HIGH);   // turn the LED on (HIGH is the vc
  } else {
    digitalWrite(led, LOW);    // turn the LED off by making the vc
  }
  delay(1);        // delay in between reads for stability

```

Figure 06 : code exemple pour tester l'outil en ligne

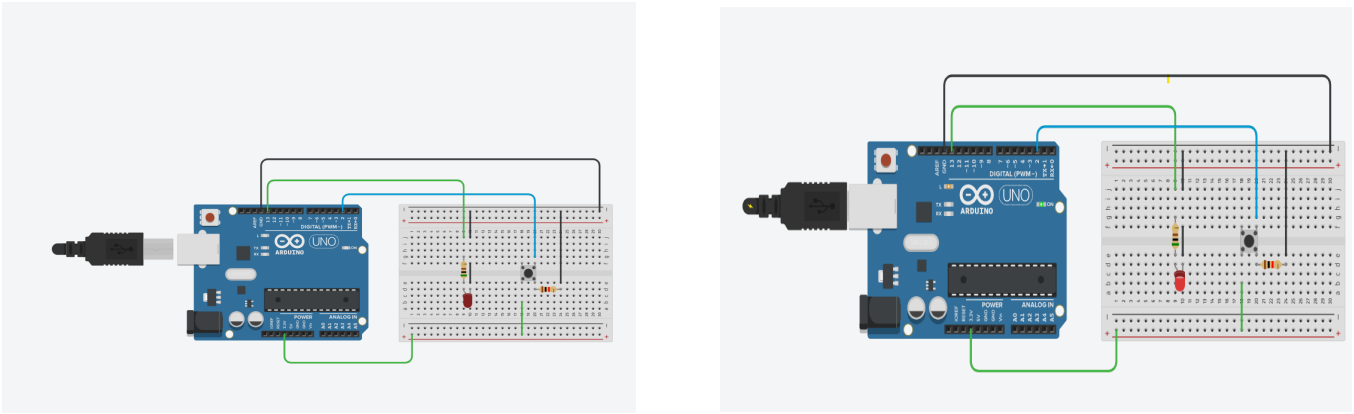

Figure 07 : Test d'un bouton poussoir

Avantages et inconvénients de l'outil:

1. Avantages
    a. Facile à utiliser pour les débutants, il permet la recherche des parties et composants.
    b. Possibilité de sauvegarder et partager des travaux/projets
    c. Gratuit et en ligne (ne nécessite aucune installation)
2. Inconvénients
    a. Les bugs et nécessité d'une très bonne connexion internet en continue sinon l'application crash
    b. Manque de documentation officielle
    c. Plusieurs composants intéressants n'existent pas encore