# Compte rendu : SAE n°2 arduino

KUZDOWICZ Théo Groupe A5

## SOMMAIRE :

# INTRODUCTION :

L'objectif de cette SAE est de manipluer la carte arduino afin de comprendre comment l'utiliser.
Nous allons tout d'abord l'utiliser avec des tâches faciles puis de plus en plus dure.
La carte arduino permet de transmettre, d'appliquer un programme.

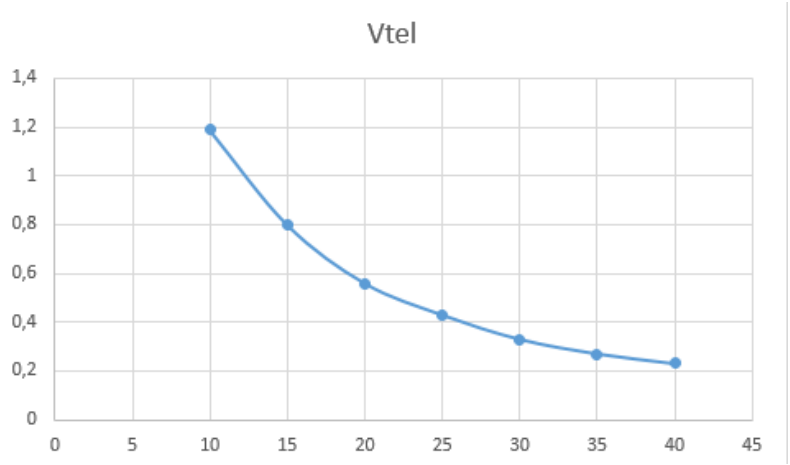
# I. Allumer et éteindre une LED :

```
/*
  Ce programme permet d'allumer une LED et de l'éteindre à plusieurs reprises toutes les secondes
*/
void setup()    // setup permet d'exécute la fonction lorsque l'on appuie sur reset ou lorsque la carte est mise sous tension
{
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);     /*pinMode(13, OUTPUT);*/   // ici nous expliquons à quell endroit nous voulons executer cette function soit LED_BUILTIN ou soit la broche 13 de la carte arduino
}
void loop()    // la fonction de boucle permet d'exécute la fonction à l'infini
{
  digitalWrite(LED_BUILTIN, HIGH);   // allumer la LED (HIGH est le niveau de tension)
  delay(100);                  // attends une seconde
  digitalWrite(LED_BUILTIN, LOW);    // éteignez la LED en réduisant la tension
  delay(100);                  // attends une seconde
```

# II. Lecture d'une entrée analogique sur le moniteur série :

```
/*
Ce programme permet de lire une entrée analogique sur la broche A0, puis affiche  le résultat sur le moniteur série.
*/
void setup() {   // setup permet d'exécute la fonction lorsque l'on appuie sur reset ou lorsque la carte est mise sous tension
  Serial.begin(9600);   // initialise la communication série à 9600 bits par seconde :
}
void loop()   // la fonction de boucle permet d'exécute la fonction à l'infini
{
  float sensorValue = analogRead(A0);    // lit l'entrée sur la broche analogique 0 :
  sensorValue=sensorValue*5/1023;
  Serial.print("La valeur en Volt est :");    /*Pour afficher du texte*/
  Serial.println(sensorValue);  // affiche la valeur que vous lisez :
  delay(1000);       // délai entre les lectures pour plus de stabilité
}
```

| f(d) | Vtel |
|---|---|
| 10 | 1,19 |
| 15 | 0,8 |
| 20 | 0,56 |
| 25 | 0,43 |
| 30 | 0,33 |
| 35 | 0,27 |
| 40 | 0,23 |



# III. Ecouter une mélodie :

## 1)

```
/*
Ce programme permet de jouer une mélodie
*/
#include "pitches.h"
// notes dans la mélodie :
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};
void setup()    // setup permet d'exécute la fonction lorsque l'on appuie sur reset ou lorsque la carte est mise sous tension
{
  for (int thisNote = 0; thisNote < 8; thisNote++)
 {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);   // pour distinguer les notes, nous définissons  un temps minimum entre elles.
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);   // arrête le son
    noTone(8);
  }
}
```

## 2)

```
/*
Ce programme est le même que le précédent cependant ici on ajoute une fonction qui permet que le
programme soit répétitif
*/
#include "pitches.h"
// notes dans la mélodie :
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};
void setup()    // setup permet d'exécute la fonction lorsque l'on appuie sur reset ou lorsque la carte est mise
sous tension
{
  for (int thisNote = 0; thisNote < 8; thisNote++)
 {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);   // pour distinguer les notes, nous définissons  un temps minimum
entre elles.

    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(8);
  }
}
void loop()  // répète la fonction à l'infini
 {
   for (int thisNote = 0; thisNote < 8; thisNote++)
{
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);   // pour distinguer les notes, nous définissons  un temps minimum
entre elles.

    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(8);
}
delay(5000);   //on ajoute une durée
}
```

# 3)

```
/*
Ce programme est le même que le précédent cependant ici on ajoute une fonction qui permet de faie varier la rapidité du morceau avec un potentiomètre
*/
#include "pitches.h"

int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};
void setup()    // setup permet d'exécute la fonction lorsque l'on appuie sur reset ou lorsque la carte est mise sous tension
{
  for (int thisNote = 0; thisNote < 8; thisNote++)
{
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);   // pour distinguer les notes, nous définissons  un temps minimum entre elles.
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(8);
  }
}
void loop()   // la fonction de boucle permet d'exécute la fonction à l'infini
 {
   for (int thisNote = 0; thisNote < 8; thisNote++)
{
  float x = analogRead(A0);

  x=x*0.0049;

    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);  // pour distinguer les notes, nous définissons  un temps minimum entre elles.
    int pauseBetweenNotes = noteDuration * x;
    delay(pauseBetweenNotes);
    noTone(8);
}
}
```

## 5)

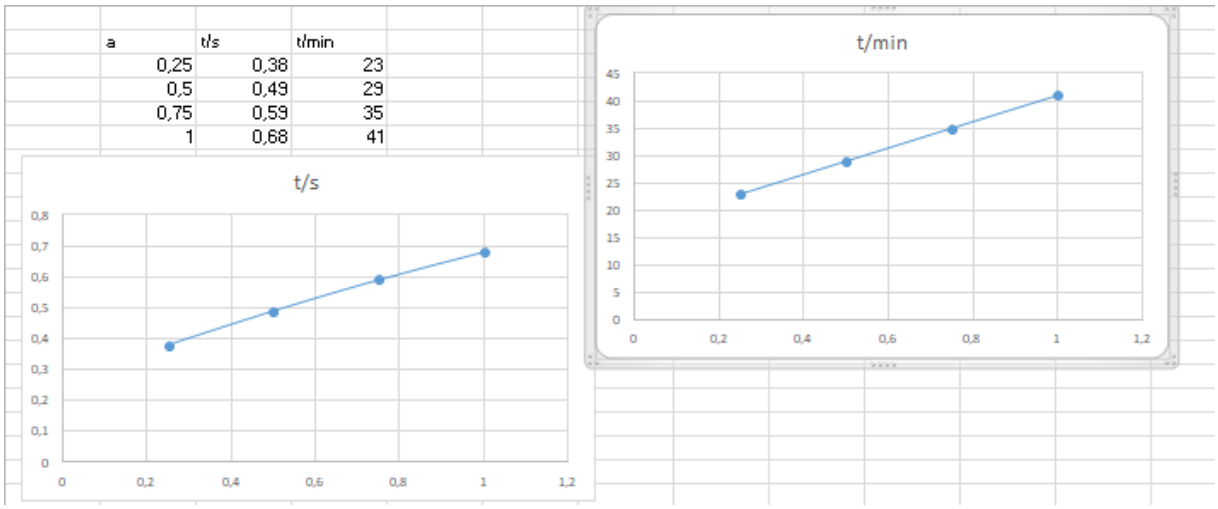
```
/*
Ici nous modifions les notes jouer pour jouer la marseillaise
*/
#include "pitches.h"

int melody[] = {
  NOTE_D3,  NOTE_D3,  NOTE_D3, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_A3, NOTE_D4, NOTE_B3, NOTE_G3,
NOTE_G3, NOTE_B3, NOTE_G3, NOTE_E3, NOTE_C4, NOTE_A3, NOTE_FS3, NOTE_G3,
};
int noteDurations[] = {
 8, 8, 8, 4, 4, 4, 4, 6, 8, 8, 8, 8, 8, 4, 2, 8, 8, 2,
};
void setup()   // setup permet d'exécute la fonction lorsque l'on appuie sur reset ou lorsque la carte est mise
sous tension
{
 for (int thisNote = 0; thisNote < 8; thisNote++)
{
   int noteDuration = 1000 / noteDurations[thisNote];
   tone(8, melody[thisNote], noteDuration);  // pour distinguer les notes, nous définissons  un temps minimum
entre elles.

   int pauseBetweenNotes = noteDuration * 1.30;
   delay(pauseBetweenNotes);
   noTone(8);
 }
}
void loop()    // la fonction de boucle permet d'exécute la fonction à l'infini
{
  for (int thisNote = 0; thisNote < 18; thisNote++)
{
 float x = analogRead(A0);

 x=x*0.0049;

   int noteDuration = 1000 / noteDurations[thisNote];
   tone(8, melody[thisNote], noteDuration);   // pour distinguer les notes, nous définissons  un temps minimum
entre elles.

   int pauseBetweenNotes = noteDuration * x;
   delay(pauseBetweenNotes);
   noTone(8);
}
}
```

# IV. Utilisation sortie PWM :

```
/*
Lit une broche d'entrée analogique, affiche  le résultat sur une plage de 0 à 255 et l'utilise pour définir la
modulation de largeur d'impulsion (PWM) d'une broche de sortie.
Imprime également les résultats sur le moniteur série.
*/
// Ces constantes ne changeront pas.
const int analogInPin = A0;  // Broche d'entrée analogique à laquelle le potentiomètre est attaché
const int analogOutPin = 9; // Broche de sortie analogique à laquelle la LED est attachée
float sensorValue = 0;       // valeur lue
int outputValue = 0;       // sortie de valeur vers le PWM
void setup()  // setup permet d'exécute la fonction lorsque l'on appuie sur reset ou lorsque la carte est mise
sous tension
{
  Serial.begin(9600);   // initialise les communications série à 9600 bps
}
void loop()   // la fonction de boucle permet d'exécute la fonction à l'infini
{
  sensorValue = analogRead(analogInPin);  // lit l'analogique en Value
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(analogOutPin, outputValue);  // change la valeur de la sortie analogique :
float cycliqueValue = sensorValue/1023;
  // affiche les résultats sur le moniteur série :
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);
  Serial.print("\t la valeur du rapport cyclique est : ");
  Serial.println(cycliqueValue , 2);
// attend 2 millisecondes avant la prochaine boucle
  delay(1000);
}
```

# V. Variation de vitesse d'une MCC :

```
/*
Lit une broche d'entrée analogique, affiche  le résultat sur une plage de 0 à 255 et l'utilise pour définir la
modulation de largeur d'impulsion (PWM) d'une broche de sortie.
Imprime également les résultats sur le moniteur série.
*/
// Ces constantes ne changeront pas.
const int analogInPin = A0;  // Broche d'entrée analogique à laquelle le potentiomètre est attaché
const int analogOutPin = 9; // Broche de sortie analogique à laquelle la LED est attachée
float sensorValue = 0;       // valeur lue
int outputValue = 0;       // sortie de valeur vers le PWM
void setup()  // setup permet d'exécute la fonction lorsque l'on appuie sur reset ou lorsque la carte est mise
sous tension
{
  Serial.begin(9600);   // initialise les communications série à 9600 bps
}
void loop()   // la fonction de boucle permet d'exécute la fonction à l'infini
{
  sensorValue = analogRead(analogInPin);  // lit l'analogique en Value
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(analogOutPin, outputValue);  // change la valeur de la sortie analogique :
float cycliqueValue = sensorValue/1023;
  // affiche les résultats sur le moniteur série :
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);
  Serial.print("\t la valeur du rapport cyclique est : ");
  Serial.println(cycliqueValue , 2);
// attend 2 millisecondes avant la prochaine boucle
  delay(1000);
}
```



# VI. Projet Robot suiveur de ligne avec priorité à droite :

```
void setup()   // setup permet d'exécute la fonction lorsque l'on appuie sur reset ou lorsque la carte est mise
{
Serial. begin(9600);     // initialise les communications série à 9600 bps
pinMode(10, OUTPUT);  // Broche de sortie analogique à la borne 10
pinMode(12, OUTPUT);    // Broche de sortie analogique à la borne 12
}
void loop()          // la fonction de boucle permet d'exécute la fonction à l'infini
{
float capteurD;
float capteurG;
float sensorValueD = analogRead(A0);
float sensorValueG = analogRead(A2);
capteurD = sensorValueD*5/1023;
capteurG = sensorValueG*5/1023;
digitalWrite (10, HIGH);
digitalWrite (12, HIGH);
if( capteurD <= 4)
{
digitalWrite(10, HIGH);
digitalWrite(12, LOW);
}
else if (capteurG <= 4)
{
digitalWrite(10, LOW);
digitalWrite(12, HIGH);
}
else if(capteurD >= 4 && capteurG >=4)
{
digitalWrite (10, HIGH);
digitalWrite (12, HIGH);
}
}
delay(5);
```

// Malheureusement pour cette partie ci du projet je n'ai pas pu aller plus loins le programme semble correcte mais même en changent des choses du programme le robot ne veut pas suivre la ligne, il ne s'arrête pas, il va toujours tout droit comme s'il ne voyait pas la ligne.
Le câblage étais correcte également.

# VII. CONCLUSION :

Pour conclure, nous avons donc manipulé la carte arduino, nous avons compris comment écrire et modifier un programme pour qu'il fasse ce qu'on lui demande à l'aide d'un cahier des charges pour chaque exemple et bien câblé les différents ports