Compte-Rendu n°3 Projet

L'objectif de cette séance était de faire fonctionner deux moteurs simultanément en fonction des détecteurs d'obstacles.

I-Résolution de la panne rencontrée et Soudage.

Problème rencontré : Au cours de la séance les moteurs ainsi que les ponts en H Cytron ont cessé de fonctionner. J'ai dû entièrement revoir les câblages puis refaire le montage ce qui a freiné l'avancée de la séance et m'a fait perdre un certain temps. Ce problème était en fait lié à une mauvaise connexion de certains fils utilisés soudés aux moteurs : j'ai déboudé les fils soudés pendant la précédente séance puis remplacé ces derniers par des fils neufs.

II- Fonctionnement de deux moteurs simultanément en fonction des détecteurs d'obstacles.

Le problème rencontré lors de la dernière séance était le suivant : avec le précédent code (comportant des delays()), un des deux moteurs ne s'activait pas lorsque le capteur ultrasonore respectivement associé à ce moteur ne détectait pas d'obstacle.

Option A- Brancher un moteur et un capteur par carte Arduino Uno

Au cours de la séance j'ai voulu tester le fait de brancher un moteur et un capteur par carte Arduino Uno (c'est à dire d'utiliser deux cartes Arduino pour la motorisation arrière de notre robot). Cette solution fonctionne convenablement.

Problème rencontré dans la manipulation : Je ne pouvais pas importer mon code sur une des deux cartes Arduino Uno (la carte était en réalité défaillante, je l'ai récupérée dans une boîte ne contenant que des Arduino Uno non fonctionnelles)

Avantage de cette solution :
-Facilité : code facile à mettre en place
-Cable Management : La gestion des fils semble plus simple à réaliser

Désavantage de cette solution :
-Coûteux : nécessite davantage de cartes Arduino Uno (4 cartes minimum au lieu de 2 pour la motorisation complète de note robot)
-Alimentation : Je n'ai pas trouvé de solution viable sans batterie externe pour pouvoir alimenter les deux cartes séparément ce qui m'a nécessité d'utiliser deux alimentation 12V branchées sur secteur et ce qui peut éventuellement poser problème pour la suite de notre projet.
-Décentralisation : Chaque carte aura son propre code, et on ne sera en aucun cas dans une optique de posséder une seule carte(=un microcontrôleur central) qui s'occupera de toutes les tâches à réaliser.
-Encombrement : L'ajout de plusieurs cartes entraîne une perte de place

Ainsi cette solution, bien que plus simple à mettre en place ne semble pas être la plus optimale. On s'intéresse alors à l'option B :

Option B- Remplacement de la fonction delay() par la fonction millis()

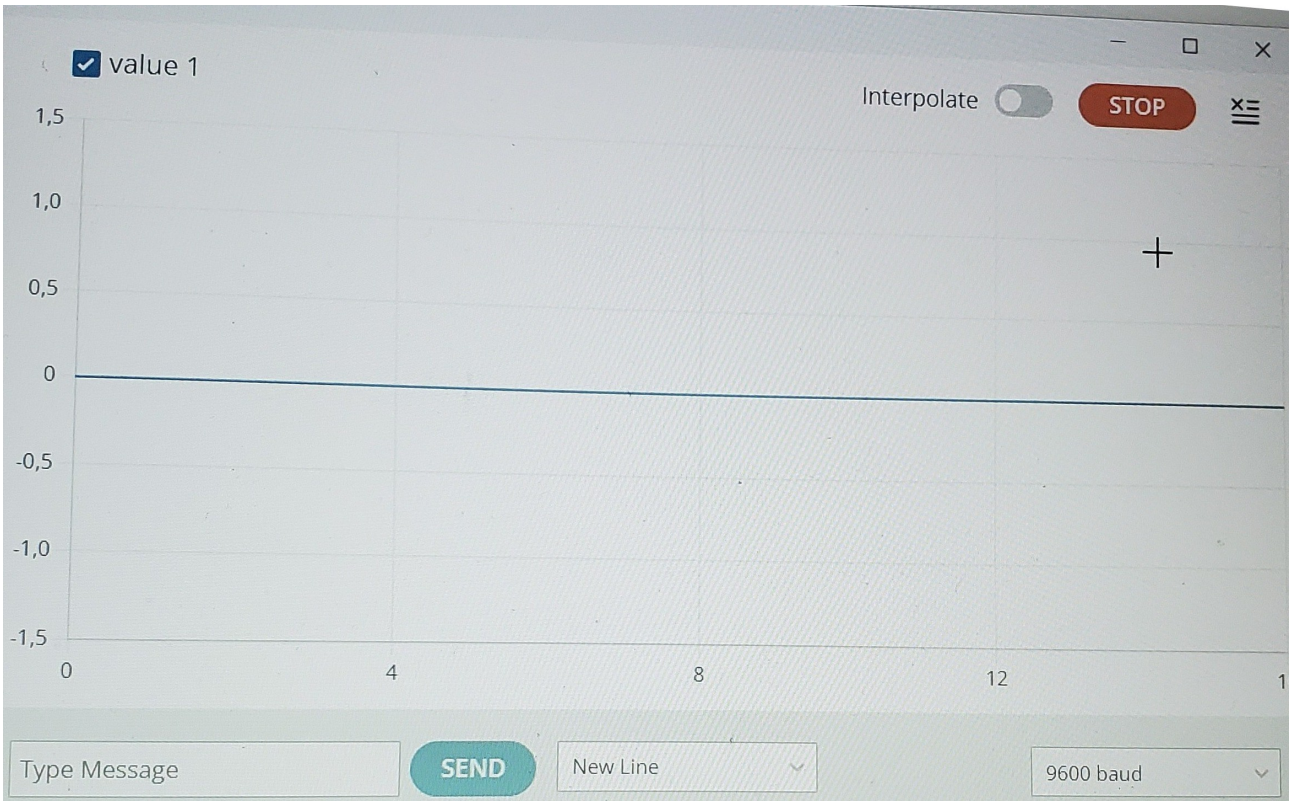
Après des recherches, ce problème est en fait lié au fonctionnement bloquant de la fonction delay() qui stoppe tout le programme. L'objectif est alors de supprimer tous les delays et de les remplacer par des millis() ou des micros() (lié aux délais de fonctionnement du capteur ultrasonore en microsecondes).

Les moteurs fonctionnent parfaitement et continuent à tourner. Toutefois avec le nouveau code, les capteurs d'obstacles ne semblent plus fonctionner et la distance affichée entre le capteur et un obstacle est constamment égale à 0.

Voici plusieurs captures d'écran du code utilisé et de la valeur constante nulle renvoyée par le capteur ultrasonore 1.

```
// Inclure la bibliothèque pour le capteur ultrasonique

// Définir les broches des capteurs ultrasoniques
#define TRIG1  9
#define ECHO1  10

#define TRIG2  11
#define ECHO2  12

// Définir les broches de contrôle des moteurs
#define MOTOR_PIN_1 3
#define MOTOR_PIN_2 5
int impulseTime1;
int impulseTime2;
int distance1;
int distance2;

// Créer des instances de la classe NewPing pour chaque capteur


// Variables pour le contrôle du temps
unsigned long previousMillis = 0;
unsigned long currentMillis;

boolean motorActivated_1 = false;
boolean motorActivated_2 = false;

void setup() {
  // Initialiser les broches de contrôle des moteurs en tant que sorties
  Serial.begin(9600);
  pinMode(MOTOR_PIN_1, OUTPUT);
  pinMode(MOTOR_PIN_2, OUTPUT);
  pinMode(TRIG1, OUTPUT);
  pinMode(TRIG2, OUTPUT);
  pinMode(ECHO1, INPUT);
  pinMode(ECHO2, INPUT);
  currentMillis=micros();
  digitalWrite(MOTOR_PIN_1,HIGH);
  digitalWrite(MOTOR_PIN_2,HIGH);
}

```

```
void loop() {

digitalWrite(TRIG1,LOW);
digitalWrite(TRIG2,LOW);

if (currentMillis-previousMillis>4){
  digitalWrite(TRIG1,HIGH);
  digitalWrite(TRIG2,HIGH);
  previousMillis=currentMillis;
  }

if (currentMillis-previousMillis>12){
digitalWrite(TRIG1,LOW);
digitalWrite(TRIG2,LOW);
previousMillis=currentMillis;
 }

impulseTime1=pulseIn(ECHO1,HIGH);
distance1=impulseTime1*0.017;
impulseTime2=pulseIn(ECHO2,HIGH);
distance2=impulseTime2*0.017;
Serial.println(distance1);

 // Vérifier le premier capteur
  if (distance1 == 0 || distance1 > 10) {
    // Aucun obstacle détecté à moins de 10 cm, activer le premier moteur
    if (!motorActivated_1) {
      motorActivated_1 = true;
      digitalWrite(MOTOR_PIN_1, HIGH);

    }
  } else {
    // Obstacle détecté à moins de 10 cm, désactiver le premier moteur après la durée spécifiée
    if (motorActivated_1) {
      motorActivated_1 = false;
      digitalWrite(MOTOR_PIN_1, LOW);
    }
  }



  // Vérifier le deuxième capteur
  if (distance2 == 0 || distance2 > 10) {
    // Aucun obstacle détecté à moins de 10 cm, activer le deuxième moteur
    if (!motorActivated_2) {
      motorActivated_2 = true;
      digitalWrite(MOTOR_PIN_2, HIGH);
    }
  } else {
    // Obstacle détecté à moins de 10 cm, désactiver le deuxième moteur après la durée spécifiée
    if (motorActivated_2) {
      motorActivated_2 = false;
      digitalWrite(MOTOR_PIN_2, LOW);
    }
  }
}
```



Objectif pour la prochaine séance: Faire fonctionner le code contenant des millis()