Projet SAE4 :

Bensebah Kheireddine
Behillil Sofian

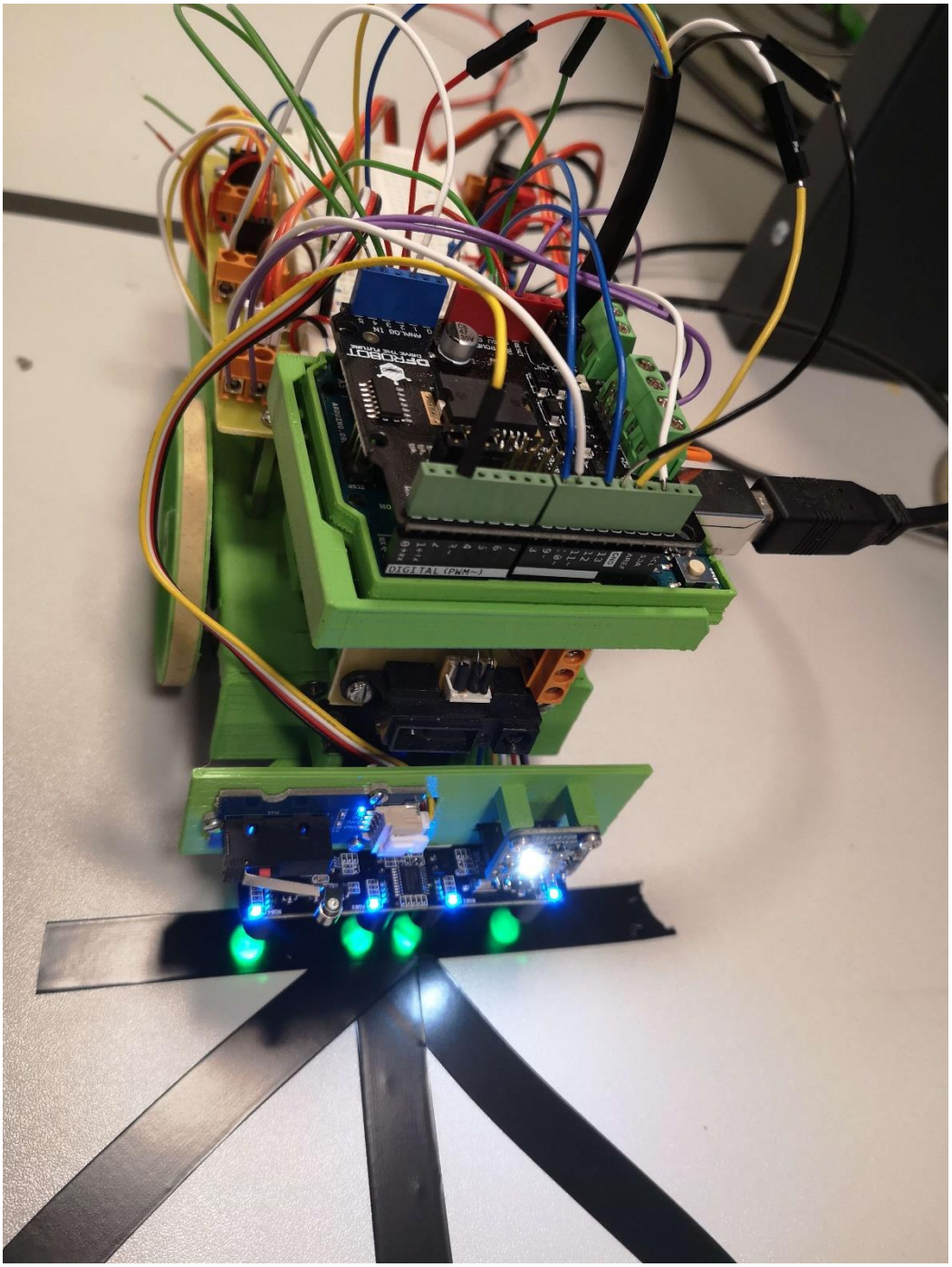


Table des matières

I. Introduction

L'objectif de cette SAE consiste à réaliser un robot suiveur de ligne capable de répondre à un scénario donné. En effet, celui-ci doit être capable de parcourir un chemin et passer les différents obstacles, voici donc,

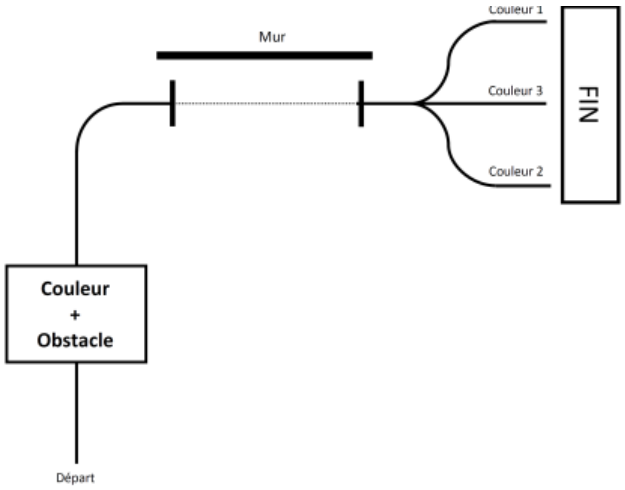
Le cahier des charges :

Le robot doit fonctionner en suiveur de ligne avec les différentes fonctions avancer, reculer, etc...

- Un capteur de couleur
- Un capteur de ligne
- Un interrupteur
- Une carte Arduino Uno (µC Atmega 328P)
- Une carte Arduino motor shield

Le parcours sera défini par différentes étapes par lesquels le robot doit passer :

1. Le départ
    a. Le robot suit la ligne à l'aide de son capteur
2. Obstacle 1
    a. Le robot réduit sa vitesse et récupère la couleur de l'obstacle
    b. Le robot recule jusqu'à une certaine distance de l'obstacle et entreprend une manœuvre pour l'éviter
    c. Le robot reprend le suivi de ligne
3. Arrivé à la première ligne horizontale
    a. Le télémètre tourne à gauche et le robot longe le mur à l'aide de celui-ci pour garder une distance constante par rapport au mur
4. Arrivé à la seconde ligne horizontale
    a. Le robot reprend son suivi de ligne et remet le télémètre en place
5. Fourchette
    a. Par rapport à la couleur récupérée à l'étape 2.a le robot prend un chemin spécifique et s'arrête à une certaine distance de l'obstacle



II. Description du fonctionnement

1)Schéma fonctionnel

Schéma bloc moteur :

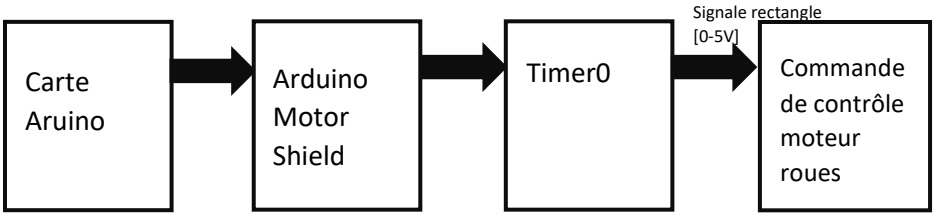


Schéma bloc télémètre :

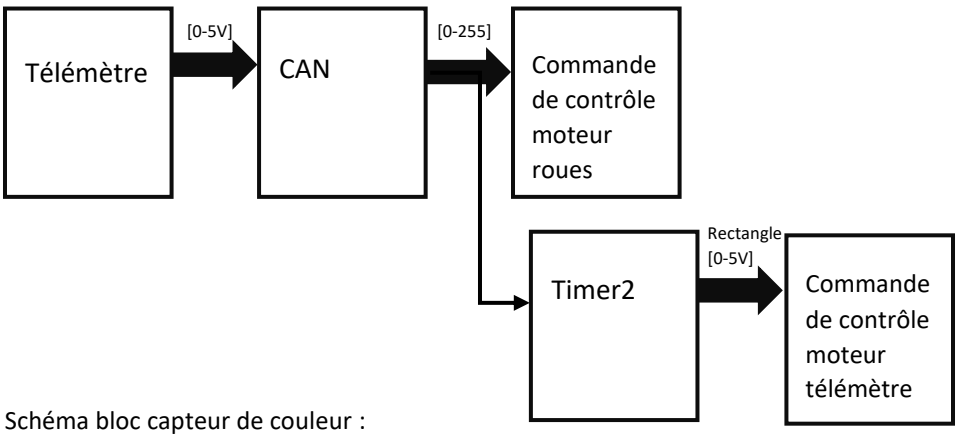


Schéma bloc capteur de couleur :

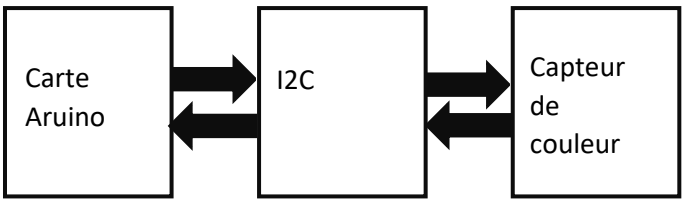


Schéma bloc capteur de l'interrupteur :

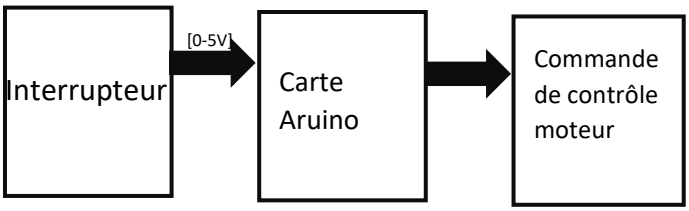
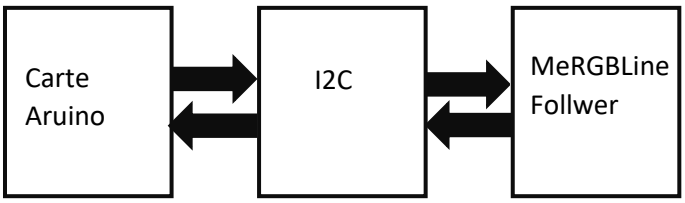


Schéma bloc capteur de ligne :



2)Explication des schémas bloc

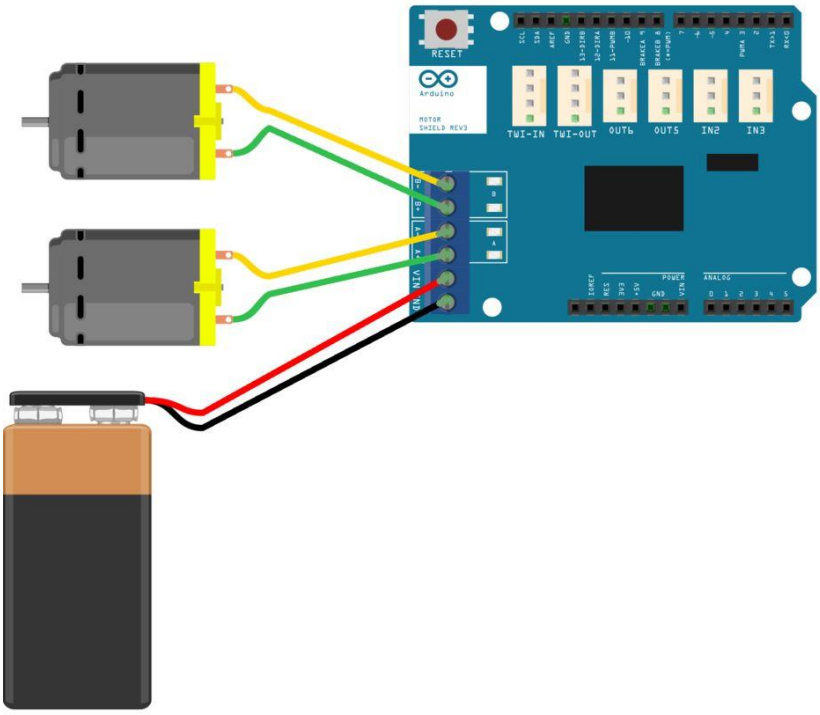
Schéma bloc moteur : Grâce à cette partie, nous pouvons faire fonctionner les roues afin de faire avancer, reculer, tourner à gauche ou à droite le robot.
Schéma bloc télémètre : Cette partie nous permet de contrôler les roues du robot et le moteur du télémètre par rapport au télémètre.
Schéma bloc capteur de couleur : Ici par une communication I2C entre l'Arduino et le capteur de couleur, nous pouvons déterminer quelle couleur est présente sur l'obstacle
Schéma bloc interrupteur : Cette partie nous permet de contrôler les roues du robot par rapport à l'état de l'interrupteur.
Schéma bloc suiveur de ligne : Pour finir, le capteur MeRGBLineFollower nous a permis de suivre la ligne.

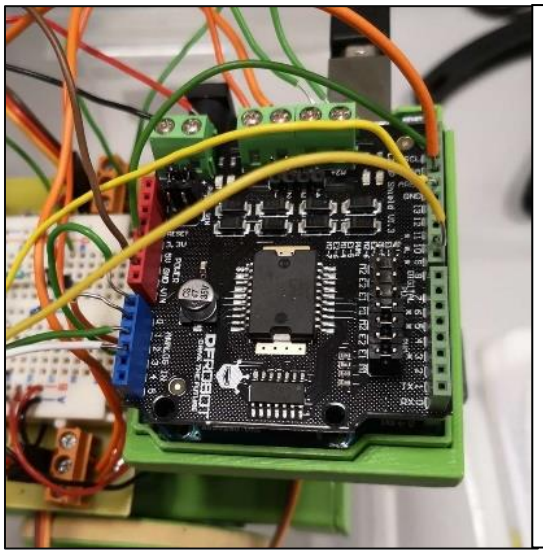
Fonctionnement du robot :

Le robot avance en mode suiveur de ligne. Le capteur MeRGBLineFollower détecte selon la ligne sur le robot est sur la ligne noire avec le fond clair afin de prendre ces décisions d'avancer, tourner à droite, etc... Quand le robot s'approche à une certaine distance de l'obstacle, il ralentit et s'arrête jusqu'à que l'interrupteur tape l'obstacle. Le capteur de couleur récupère la couleur qui est sur l'obstacle et la conserve en mémoire pour plus tard. Le robot recule de quelques centimètres pour tourner sur le côté afin que le télémètre qui tournera sur la gauche soit face à l'obstacle à éviter. Le robot va donc contourner l'obstacle jusqu'à détection de la ligne pour reprendre le suivi de ligne jusqu'à la partie suivante. Le robot détecte une ligne horizontale, le télémètre tourne sur la gauche et le robot longe le mur grâce au télémètre en maintenant une distance par rapport au mur avant d'arriver au même endroit que le point de départ à la première ligne horizontale. Arrivé donc à la seconde ligne horizontale, le robot reprend le suivi de ligne pour arriver à la fourchette. En fonction de la couleur prise sur l'obstacle du début, le robot avance et prend le chemin donné.

III. Fonctionnement du moteur des roues

1)Câblage





Pour le câblage, nous avons alimenté en 5 V par un générateur les moteurs. Et nous avons connecté les pins au driver (l298P) M1+ pour la vitesse d'une des roues et M1- pour le sens d'une des roues. On établit le même processus pour M2+ et M2- la seconde roue. À savoir que ces pins sont reliés à la carte Arduino aux broches 4(sens), 5(vitesse), 6(vitesse) et 7(sens).

2)Explication du code et calculs

Pour le fonctionnement du robot, nous avons donc repris le schéma de fonctionnement que le projet précédent.

Cela nous a permis un gain de temps, car les tests pouvaient être faits directement avec un code de test que nous avions programmé avec les registres du Timer0 sans passer par les codes fournit par Arduino. (Voir annexe 1)

Pour obtenir ce code, nous avions donc défini une période de 20ms et le Timer0 que nous utilisons, car les pins nécessaires correspondent aux broches 4, 5, 6 et 7 du Arduino motor shield et donc du PortD correspondant au Timer0.

Les registres TCCR0A/B sont configurés de cette façon :

Pour TCCR0A :

Bit	7	6	5	4	3	2	1	0
	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM00
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Nous avons utilisé le mode 3 connecté en pwm rapide donc nous avons mis les bits 5 et 7 à 1.
Et ensuite, nous avons mis à 1 les deux premiers bits pour le pwm rapide.
Donc, TCCR0A=0xA3.

Pour TCCR0B :

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM02	CS0[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Les 3 premiers bits servent à donner la pré-division de notre timer pour cela, on doit procéder à un calcul : On sait que la fréquence de quartz de l'Arduino, donc de l'Atmega 328P est de 16MHZ, donc T=1/f=1/16×10^6=6.25×10^-8 s. Comme dit précédemment, la période de notre signal est de 20ms donc pour calculer la pré-division nécessaire, on prend la demi-période c'est-à-dire 10ms et on calcule la pré-division n= demi-période/2^16*T=2.44 Donc la pré-division recommandée ici sera de 8. Pour cela, on met le bit 2 à 1.
Donc, TCCR0B=0x02

CA02	CA01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

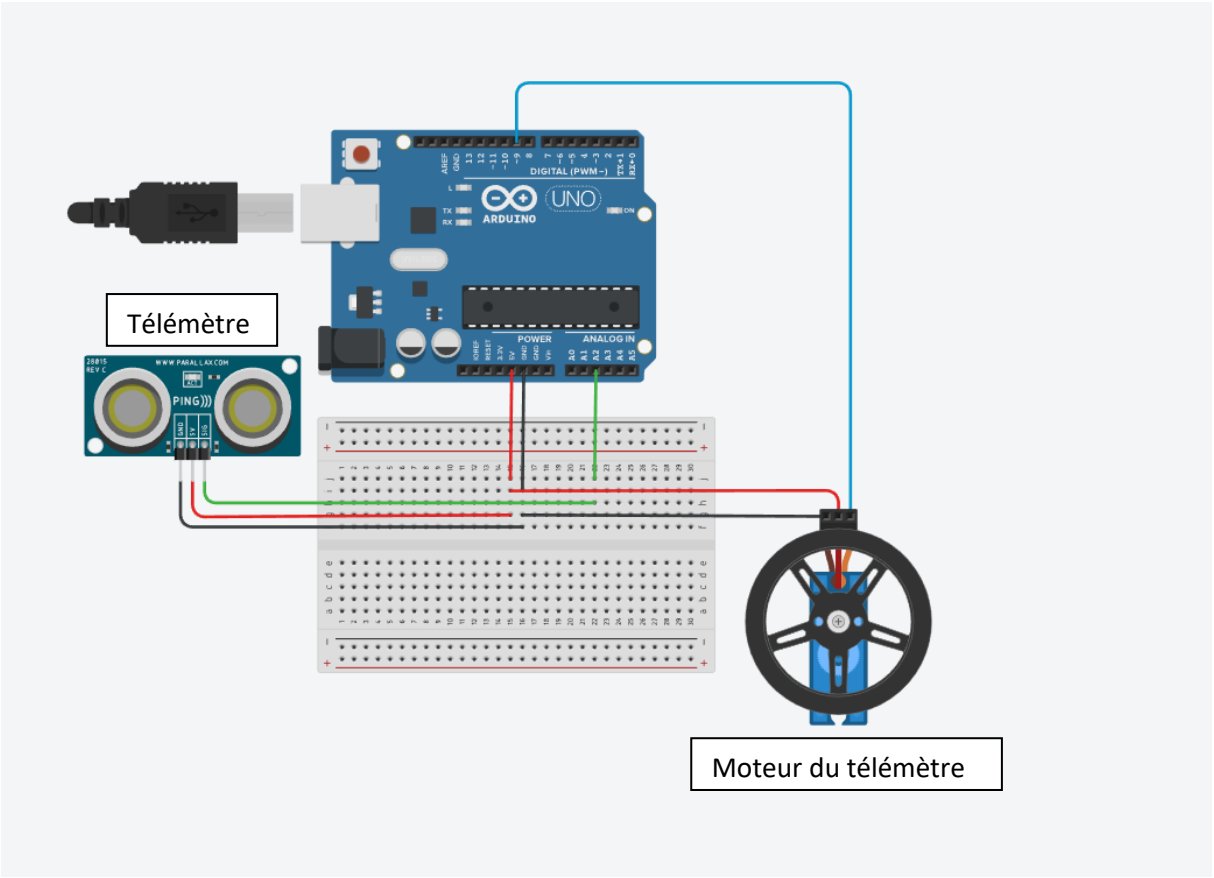
Enfin, pour définir la vitesse des roues sachant que le robot n'avait pas tout le temps la même vitesse selon des paramètres que nous ignorons, on augmente ou diminue la valeur d'OCR0A/B pour espérer que le robot avance le plus droit possible.
En ce qui concerne le sens de rotation des roues, il suffit de mettre le port en sortie associe aux pins de l'Arduino soit à l'état haut pour avancer, soit à l'état bas pour reculer. (Voir annexe 3)

3)Validation théorique

Après des tests et réglage, le robot roule correctement et nous pouvons donc passer à l'étape suivante.

IV. Fonctionnement du moteur du télémètre

1)Câblage



Pour le câblage, on imagine donc que le télémètre est donc fixé au moteur. Celui-ci est alimenté en 5V par l'Arduino et le câble de commande au pin 9, car il permet d'envoyer des signaux pwm. Pour le télémètre, il est aussi connecté à l'Arduino en 5V et la broche qui permet de recevoir les signaux sera au pin A2 qui est une broche analogique qui permet de recevoir les signaux.

2) Mesures et code

Tout comme pour le moteur, nous avons pu réutiliser le code du projet précédent (annexe 2).

Pour contrôler le servo moteur du télémètre, nous avons utilisé le timer1. Nous avons utilisé pour celui-ci le mode pwm 14 déconnectés avec une pré-division de huit en utilisant le même calcule que pour les moteurs des roues.

Pour TCCR1A :

Bit	7	6	5	4	3	2	1	0
	COM1	COM1	COM1	COM1			WGM11	WGM10
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Nous avons donc mis les bits 7, 5 et 2 à 1 ce qui définit notre mode 14 déconnecté d'après la datasheet ne fournit pas le constructeur.

Pour TCCR1B :

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Nous avons mis les bit 4, 3 et 2 pour le pwm mode 14 et pour la pré-division de 8 on a mis le bit 2 à 1.
Pour programmer le registre ICR1 on a procédé à un calcul qui est le suivant :
On divise la période de 20ms par la période du quartz, le tout divisé par la pré-division de 8.

20*10^-3/6.25*10^-8/8=40000
On prendra donc ICR1=40000.

Ensuite pour OCR1B que nous n'utilisons pas, car la broche associée au timer1 n'est pas utilisé, car le branchement du servo moteur ne nécessite que 1 câble (hors alimentation), il sera donc toujours à 0.

Pour définir OCR1A, on procède au même calcul que pour ICR1 en remplaçant la période de 20ms par le temps haut associé à l'angle de rotation.

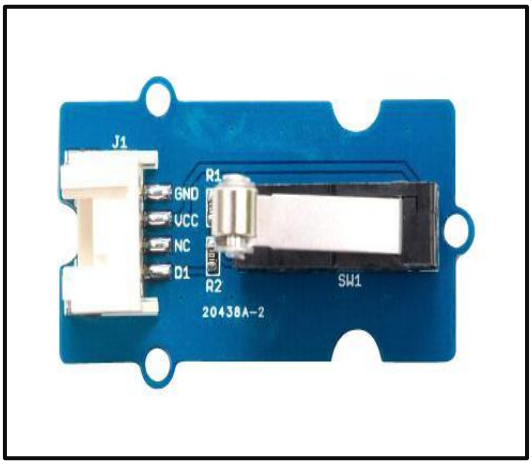
OCR1A(70°)=1.25ms/6.25*10^-8/8=2500
OCR1A(0°)=500µs/6.25*10^-8/8=1000
OCR1A(160°))2.2ms/6.25*10^-8/8=4400

3) Validation théorique

Le télémètre tourne bien dans les deux sens.

V. Interrupteur

1) Câblage



L'interrupteur est branché en 5V à la carte Arduino, avec la masse et la broche D1 de l'interrupteur au pin 2 de l'Arduino toujours.

2)Explication

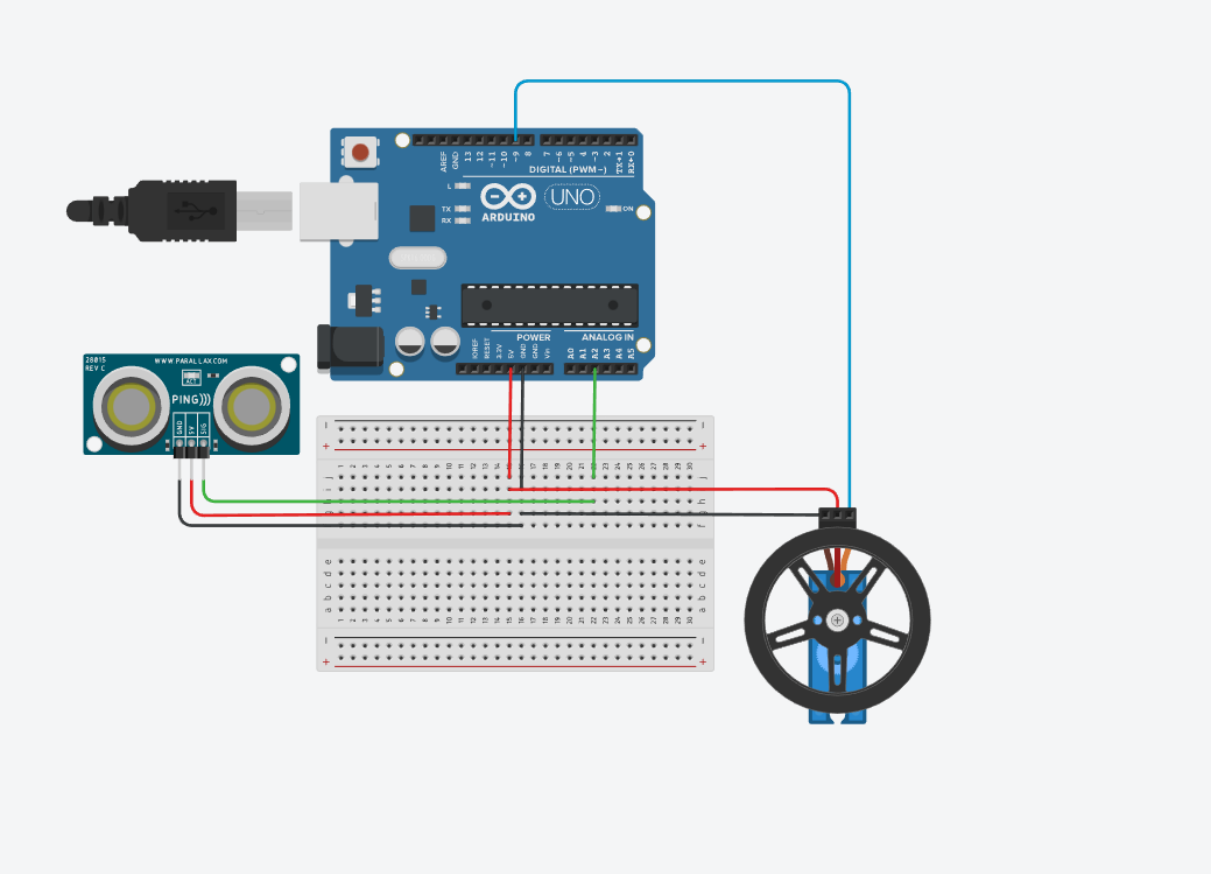
Grâce à cet interrupteur, le robot est en mesure de s'arrêter lorsque le switch est actionné. Pour cela, on a donc branché l'interrupteur et la broche D1 nous donnera l'état de l'interrupteur. Il nous suffit donc de programmer l'Arduino pour faire arrêter le robot lorsqu'il y a un changement d'état. On a donc créé une variable interruptrice qui nous donnera l'état de celui-ci, on a aussi mis le pin en maintient à l'état haut pour différentier quand on a un changement d'état. Pour terminer, on ajoute une série de conditions pour connaître l'état de l'interrupteur et agir en conséquence. (Annexe 3)

3)Validation

Le switch fonction et le changement d'état est bien perçu, le robot pourra donc s'arrêter devant le premier obstacle.

VI. CAN

1)Câblage



Le télémètre est connecté à l'Arduino en 5V et la broche qui permet de recevoir les signaux sera au pin A2 qui est une broche analogique qui permet de recevoir les signaux.

2)Explication du code

Pour pouvoir utiliser le télémètre, nous avons eu besoin d'utiliser un CAN qui servira à convertir le signal analogique fourni par le capteur en un nombre numérique. Pour convertir ces valeurs, nous devons donc configurer les registres ADCSRA et ADMUX :

Pour ADCSRA :

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Pour activer la conversion, on met le bit 7 à 1. Et ensuite, on met à 1 les bit 0, 1 et 2 pour avoir une division de 128 entre la fréquence d'horloge du système et l'horloge d'entrée ADC. ADCSRA=0x87.

Dans notre programme (annexe 4), on lancera donc la conversion en mettant le bit 6 à 1. On attend le flag pour dire que la conversion est terminée, alors on remet le flag à zéro puis on récupèrera la valeur convertie de

notre signal analogique en un nombre dans ADCH. Nombre que nous utiliserons par exemple pour le suivi de ligne dans une suite de condition en « if » quand la valeur d'ADCH est inférieur à une certaine valeur cela veut dire qu'on est sûr du noir donc le robot tourne à gauche… Pour le télémètre, si ADCH est inférieur à une certaine valeur, c'est que le robot s'approche d'un obstacle…

Pour ADMUX :

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

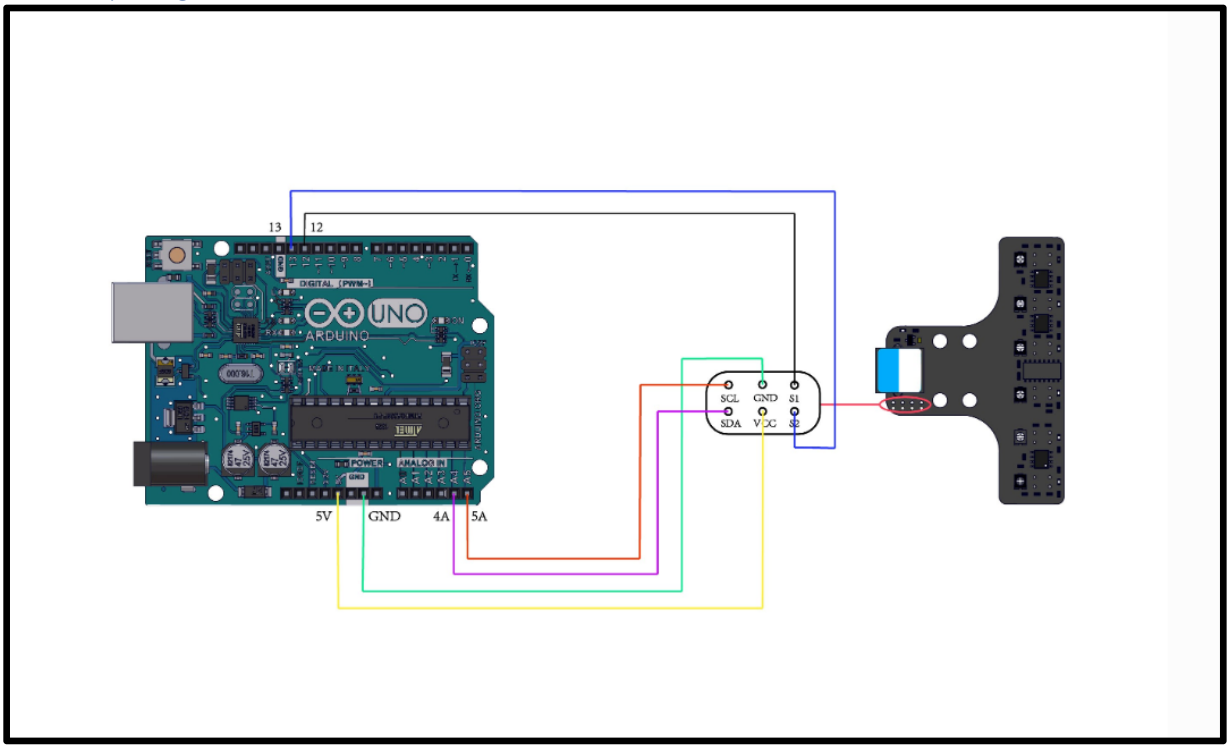
Dans notre fonction de conversion du signal analogique, on a programmé le registre ADMUX pour mettre la conversion sur les bits de poids fort et ensuite, on a programmé les bits mux0 à mux2 selon nos besoins, si on voulait un signal sur la broche A3 par exemple, on mettait mux1 et mux0 à 1.

3)Validation théorique

La conversion de la distance entre le capteur et l'obstacle fonctionne.

VII. Suiveur de ligne

1)Câblage



Le capteur MeRGBLineFollower est alimenté en 5V par l'Arduino. Ensuite aux broches analogiques A4 et A5, nous avons SCL et SDA. Et pour terminer, S1 et S2 qui permettent d'assigner l'adresse I2C du capteur sont reliés aux broches 12 et 13 de l'Arduino.

2)Explication code

En ce qui concerne le code, contrairement aux étapes précédentes, nous n'avons pas besoin de coder de registre. Nous avons tous simplement à l'aide du professeur adapté les bibliothèques du capteur pour qu'on soit capable de l'utiliser sur Arduino.

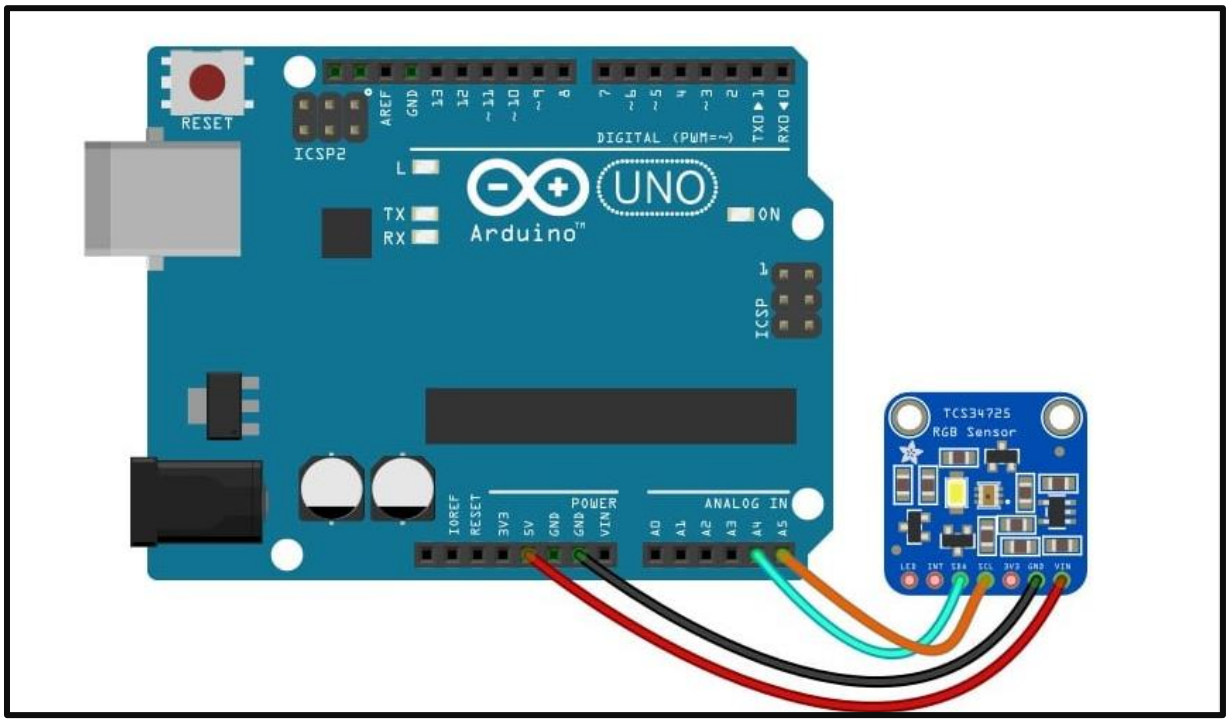
Par la suite, nous avons donc implémenté les bibliothèques nécessaires au capteur dans le code principal. Ensuite, nous avons créé une instance de classe afin de spécifier les différentes données du capteur tel que son adresse pour la communication I2C. Pour suivre, on a déclaré la variable que nous utiliserons pour suivre la ligne nommée « state » qui prendra la valeur de la somme en binaire des capteurs actionnés, c'est-à-dire que si tous les capteurs sont actionnés « state » =15 à l'inverse si tous les capteurs sont éteints « state » =0. Ensuite dans le setup, on démarre le suivi de ligne et en fonction de l'activation des différents capteurs le robot ira tout droit, à gauche ou à droite. (Voir annexe 5)

3)Validation

Le code que nous avons programmé fonctionne et nous permet donc de faire suivre le chemin au robot.

VIII. Capteur de couleur

1)Câblage



Le capteur de couleur TCS34725 est alimenté en 3V3 par le pin correspondant sur le capteur (le câble rouge sur l'image ci-dessus n'est pas sur le bon pin). Ensuite les broches SDA et SCL sont branchés tout comme le suiveur de ligne aux entrées analogique A4 et A5. Et enfin ce dernier branchement est en fonction de nos besoins, mais si nous le voulons, on peut brancher le pin « LED » à la masse afin d'éteindre la LED blanche du capteur.

2)Explication code

En ce qui concerne le code, la plupart des codes que nous pouvons trouver sur internet utilisent directement la bibliothèque associée au capteur. Mais pour aller plus loin, nous avons décidé de ne pas le faire donc en s'inspirant de ce que nous trouvons comme exemple et en reprenant les paramètres que nous trouvons dans la bibliothèque du TCS34725 nous avons programmé la communication I2C entre le capteur de couleur et l'Arduino.
Pour cela, nous avons défini tous les paramètres les plus importants à utiliser pour le bon fonctionnement du capteur (voir annexe 6).

On se retrouve donc avec cette configuration de capteur :

Configuration	Utilité
0x80	Active le capteur
0x10	Configure le temps d'intégration à 16 cycles de l'horloge interne
0x08	Désactive la répétition de la mesure de couleur
0x02	Active la mesure de couleur
0x01	Commande pour activer l'oscillateur interne
0x03	Commande pour configurer le temps d'attente à 2,4 ms
0x81	Mis à jour les valeurs des canaux de couleur
0x8F	Commande pour configurer le temps d'intégration et le gain
0x03	Temps d'intégration de 101ms
0x10	Gain x16

L'ordre utilisé dans le code est très important, car il définit précisément chaque action du capteur, notamment pour la configuration du gain et du temps d'intégration qui sont deux commandes utilisées deux fois, mais totalement distinctes. Elles sont d'ailleurs très importantes, car le gain permet de mieux différentier les nuances de couleur et le temps d'intégration lui plus est élevé plus, il permet de détecter les différents niveaux de couleur dans de faibles niveaux de luminosité.

Une fois que le capteur est configuré, il ne nous reste plus qu'à récupérer les valeurs de couleur et récupérer celle qui domine par rapport aux autres, ce qui définira si le robot verra du rouge, du vert ou encore du bleu.

3)Validation

Après testé le capteur face aux différentes couleurs, celui-ci est bel et bien fonctionnel.

IX. Evitement d'obstacle

1)Explication

Dans cette partie, nous allons détaille une des parties les plus importantes, c'est-à-dire la façon dont on a procédé pour que le robot évite le premier obstacle à l'aide du télémètre. (Annexe 7)

Pour différentier les deux moments les plus importants dans cette manœuvre, nous avons créé une variable partie qui va dans un premier temps concerner le moment où le robot recule de 5cm en arrière et se met sur le côté pour que lorsque le télémètre tourne à gauche soit face à l'obstacle. Ensuite dans la seconde partie, le robot va se focaliser à garder une distance constance entre lui et l'obstacle jusqu'à détection de la ligne pour se retourner et être dos au mur pour reprendre son suivi de ligne. A la fin de cette manœuvre pour chaque différente partie du parcours, on définit un mode et en fonction de ce mode le robot applique les consignes données pour réaliser les différentes tâches, précisément ici à la fin de l'évitement de l'obstacle le robot passera en mode 3, c'est-à-dire qu'il continuera en mode suiveur de ligne.

(La partie où le robot longe le mur fonctionne de la même façon sans l'étape où le robot se met sur le côté)

2)Validation

Le robot évite donc bien l'obstacle de la même façon qu'il arrive à longer le mur.

X. Fourchette

1)Explication

Pour terminer ce compte rendu, nous allons détailler la partie qui concerne la fourchette, c'est-à-dire le moment où le robot prend le chemin par rapport à la couleur qu'il avait prise lors du contact avec le premier obstacle. (Annexe 8)

Tout comme pour l'évitement d'obstacle, nous avons ajouté une variable nommée « p » pour partie, il définira les différents moments les plus importants pour bien choisir l'endroit où aller.

Une fois donc que le robot a capté une couleur et qu'il arrive au moment de la fourchette, nous avons trois différentes conditions par rapport à une variable nommée « couleurfinale » définit par rapport à la couleur obtenue où couleurfinale=1→couleur rouge, couleurfinale=2→couleur verte et couleurfinale=3→couleur bleue. Si le robot doit aller tout droit au moment de la fourchette, celui-ci doit attendre que le suiveur de ligne ait tous ces capteurs allumés, cela définira le bon moment pour manœuvrer, car on sera donc tout près des différents chemins, pour aller tout droit on ajoutera un légé delay pour que le robot avance et que les capteurs sur les côtés ne soient pas affectés par les autres chemins afin de reprendre un suivi de ligne normal et s'arrêter à quelques centimètres du mur de fin. Et même chose pour les deux autres chemins sur les côtés, si le robot doit aller à gauche, il attendra la ligne horizontale pour que tous les capteurs soient allumés afin de tourner à gauche avec un delay court de 1s et il s'ajustera par rapport à la ligne de gauche, c'est-à-dire que tant que le capteur opposé (celui de droite) n'est pas le seul allumer le robot continue d'aller à gauche et lorsqu'il est le seul allumé le robot reprend un suivi de ligne normal pour après aussi s'arrêter.

XI. Conclusion

Le robot fonctionne totalement, grâce à notre travail, nous avons pu finir 3ieme de la compétition entre nos groupes de GEII.

Lien fonctionnement robot :

https://youtube.com/watch?v=NQ4vZcodQXs&feature=share8

Annexes :

Annexe1 :

moteur.ino

```
void setup() {
  Serial.begin(9600);
  DDRD = 0xFF; //Pin en sortie pour les roues
  //Registres pour les roues
  TCCR0A = 0xA3; //nomal co pwm
  TCCR0B = 0x02;
}

void loop()
{
 avancer();
}

//Fonctions moteur
void arret()
{
  OCR0B = 0;
  OCR0A = 0;
  PORTD &= ~(1<<PD4);
  PORTD &= ~(1<<PD7);
  PORTD &= ~(1<<PD5); //pour que les roues ne tournent vraiment plus au cas ou ca tourne vraiment très légèrement
  PORTD &= ~(1<<PD6); //pour que les roues ne tournent vraiment plus au cas ou ca tourne vraiment très légèrement
}

void reculer()
{
  OCR0B = 100;
  OCR0A = 100;
  PORTD &= ~(1<<PD4);
  PORTD &= ~(1<<PD7);
}

void droite()
{
  OCR0B = 100;
  OCR0A = 0;
  PORTD |= (1<<PD4);
  PORTD |= (1<<PD7);
}

```

Annexe2 :

telemoteur.ino

```
void setup() {
  Serial.begin(9600);
  DDRB = 0xFF;//Pin en sortie pour le servo moteur du telemetre
  //Resgistre pour le servo du telemetre
  TCCR1A = (1<<WGM11)| (1<<COM1B1)| (1<<COM1A1); //0x82
  TCCR1B = (1<<WGM13)| (1<<WGM12)|(1<<CS01); //0x1A
  ICR1 = 40000;
  OCR1A = 2500;
}

void loop()
{
 teledroite();
}

//Fonction telemetre
void teledroite()//tele à droite
{
  OCR1A = 1000;
  OCR1B=0;
}

void teledroit() //tele droit
{
  OCR1A = 2500;
  OCR1B=0;
}

void telegauche()//tele à gauche
{
  OCR1A = 4400;
  OCR1B=0;
}

```

Annexe3 :

interrupteur.ino

```
// interrupteur connectée à l'entrée 2 de l'Arduino
int interrupteur_pin = 2;

void setup() {
  // Initialisation de la communication série
  Serial.begin(9600);
  // Configuration de l'entrée numérique pour l'interrupteur
  pinMode(interrupteur_pin, INPUT_PULLUP); //Résistance pullup interne pour maintenir à l'état haut
}

void loop() {
  // Lecture de l'état de l'interrupteur
  int etat_interrupteur = digitalRead(interrupteur_pin);
  // Si l'interrupteur est actionné, afficher un message
  if (etat_interrupteur == LOW) {
    Serial.println("L'interrupteur n'est pas actionné !");}
  else
  {
    Serial.println("L'interrupteur est actionné !");
  }
    // Attendre 1 seconde pour éviter l'affichage répété du message
    delay(1000);
}

```

Annexe4 :

CANcapteurs.ino

```
#include <avr/io.h>

void setup() {
  Serial.begin(9600);
  DDRC=0; // Configuration des broches de PORTC en entrée pour récupérer les valeurs des capteurs
  ADCSRA = 0x87; // Configuration de l'ADC
}

// Télémètre CAN
int telemetre() {
  ADMUX = (1<<ADLAR); // Conversion sur les bits de poids fort

  ADMUX = ADMUX & ~(1<<MUX0); // Signal sur A2

  ADMUX = ADMUX | (1<<MUX1); // Signal sur A2

  ADCSRA = ADCSRA | (1<<ADSC); // Lancement de la conversion

  while((ADCSRA & (1<<ADIF)) == 0); // Attente du flag de fin de conversion

  ADCSRA = ADCSRA | (1<<ADIF); // Mise à 1 du flag de fin de conversion

  Serial.println(ADCH); // Affiche la valeur convertie (ADCH) sur le moniteur série

  if (ADCH > 67) {
    return 1; // Obstacle détecté
  } else {
    return 0; // Pas d'obstacle détecté
  }
}

void loop() {
  telemetre(); // Appel de la fonction de lecture du télémètre
}

```

Annexe5 :

MeRGBLineFollower_exemple.ino ● | MeRGBLineFollower.cpp | MeRGBLineFollower.h | MeRGBLineFollower_exemple2.ino | save.ino

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include "MeRGBLineFollower.h"

MeRGBLineFollower RGBLineFollower(13, 12, 0x20);


int16_t state = 0;

void setup()
{
 Serial.begin(9600);
 RGBLineFollower.begin();
 RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);
}

void loop()
{
 RGBLineFollower.loop();

 state = RGBLineFollower.getPositionState();
 Serial.println(state);
 Serial.print("\n");
 delay(400);

 if(state==0||state==2||state==4||state==5||state==6||state==7||state==9||state==10||state==11||state==13||state==14||state==15)
 {
  Serial.print("droit");
 }
 else if (state==1||state==3)
 {
  Serial.print("tourner gauche");
 }
 else if (state==8||state==12)
 {
  Serial.print("tourner droite");
 }

}

```

Annexe 6 :

capteur_de_couleur.ino

```
#include <Wire.h>

int TCS34725_ADDR = 0x29; // Adresse du capteur

void setup() {
  Wire.begin();
  Serial.begin(9600);

  // Configuration du capteur
  Wire.beginTransmission(TCS34725_ADDR);
  Wire.write(0x80); // Commande pour activer le capteur
  Wire.write(0x10); // Commande pour configurer le temps d'intégration à 16 cycles de l'horloge interne
  Wire.write(0x08); // Commande pour désactiver la répétition de la mesure de couleur
  Wire.write(0x02); // Commande pour activer la mesure de couleur
  Wire.write(0x01); // Commande pour activer l'oscillateur interne
  Wire.write(0x03); // Commande pour configurer le temps d'attente à 2,4 ms
  Wire.endTransmission();
  delay(10);
  Wire.beginTransmission(TCS34725_ADDR);
  Wire.write(0x81); //mettre à jour les valeurs des canaux de couleur
  Wire.endTransmission();
  delay(10);
  Wire.beginTransmission(TCS34725_ADDR);
  Wire.write(0x8F); // Commande pour configurer le temps d'intégration et le gain
  Wire.write(0x03); // Temps d'intégration de 101ms
  Wire.write(0x10); // Gain x16
  Wire.endTransmission();
  delay(10);
}
```

```
void loop() {
  // Lecture des valeurs de couleur
  uint16_t clear, red, green, blue; // Déclaration de variables entières non signées
  Wire.beginTransmission(TCS34725_ADDR);
  Wire.write(0x94); // Commande pour lire les valeurs de couleur
  Wire.endTransmission();
  Wire.requestFrom(TCS34725_ADDR, 8); // Demande de 8 octets de données
  clear = Wire.read() | (Wire.read() << 8); // Clear
  red = Wire.read() | (Wire.read() << 8); // Rouge
  green = Wire.read() | (Wire.read() << 8); // Vert
  blue = Wire.read() | (Wire.read() << 8); // Bleu

  // Affichage des valeurs sur le moniteur série
  Serial.print("Clear: "); Serial.print(clear); Serial.print("\t");
  Serial.print("R: "); Serial.print(red); Serial.print("\t");
  Serial.print("G: "); Serial.print(green); Serial.print("\t");
  Serial.print("B: "); Serial.print(blue); Serial.println();

  // Détermination de la couleur dominante
  if (red > green && red > blue) {
    Serial.println("La couleur dominante est rouge");
  } else if (green > red && green > blue) {
    Serial.println("La couleur dominante est vert");
  } else if (blue > red && blue > green) {
    Serial.println("La couleur dominante est bleu");
  } else {
    Serial.println("Aucune couleur ne domine");
  }

  // Attendre une seconde avant de lire à nouveau
  delay(1000);
}
```

Annexe 7 :

```
330   void deplacement2() //le robot évite l'obstacle grâce au télémètre
331   {
332   RGBLineFollower.loop();
333   state = RGBLineFollower.getPositionState();
334   telemetre();
335
336   if (partie==1)
337   {
338     reculer();
339     delay(200);
340     //télé gauche
341     telemetregauche();
342     //partie=2;
343     //on fait tourner le robot sur place pour être sur le côté avec le télémètre face à l'obstacle à évité
344     OCR0B = 80;
345     OCR0A = 70;
346     PORTD &= ~(1<<PD7);
347     PORTD |= (1<<PD4);
348     delay(200);
349     arret();
350     partie=2;
351   }
352
353
354   if (partie==2)
355   {
356     //on suit le mur en restant à une certaine distance
357     if (ADCH>=46 && ADCH<= 64)
358     {avancer();}
359     else if (ADCH>46)
360     {droite();}
361     else if (ADCH<46)
362     {gauche();}
363     if (state==15 || state==7 || state==14) //quand les capteurs détectent la ligne le robot se remet tout droit avec le télémètre tout droit aussi et on change de mode pour continuer le parcours
364     {
365     arret();avancer();delay(130);telemetredroit();
366     OCR0B = 80;
367     OCR0A = 70;
368     PORTD &= ~(1<<PD7);
369     PORTD |= (1<<PD4);
370     delay(140);
371     arret();
372     delay(500);
373     mode=3;
374     }
```

Annexe 8 :

```
void deplacement6()// en fonction de la couleur que nous avons eu le robot suit différent chemin
{
RGBLineFollower.loop();
state = RGBLineFollower.getPositionState();
telemetre();
int p;
p=0;
  if (couleurfinale==2&&p==0)
    {
      gauche();
      delay(100);
      p=1;
      if (state==0||state==1||state==2||state==3||state==4||state==5||state==6||state==7||state==9||state==11||state==12||state==13||state==14||state==15 && p==1)
      {gauche();}
      else if (state==8 || state==10&& p==1)
      {arret();mode=7;RGBLineFollower.setRGBColour(RGB_COLOUR_RED);delay(200);RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);}
    }

   else if (couleurfinale==1)
    {
      if (state==15||state==11||state==0||state==5||state==7||state==9||state==10||state==13||state==14||state==8||state==1||state==6)
      {avancer();}
   else if (state==3||state==2)
  {gauche();Serial.print("gauche");}

  else if (state==12||state==4)
  {droite();Serial.print("droite");}

   if (ADCH>=46 && ADCH<=103 && couleurfinale==1)
    {arret();mode=8;RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);delay(200);RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);}
    }

    else if (couleurfinale==3  &&p==0)
    {
      droite();
      delay(40);
      p=1;
       if (state==0||state==2||state==4||state==5||state==6||state==7||state==8||state==9||state==10||state==11||state==12||state==13||state==14||state==15&&p==1)
      {droite();}
      else if (state==1 || state==3 &&p==1)
      {arret();mode=7;RGBLineFollower.setRGBColour(RGB_COLOUR_BLUE);delay(200);RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);}
    }
}
```