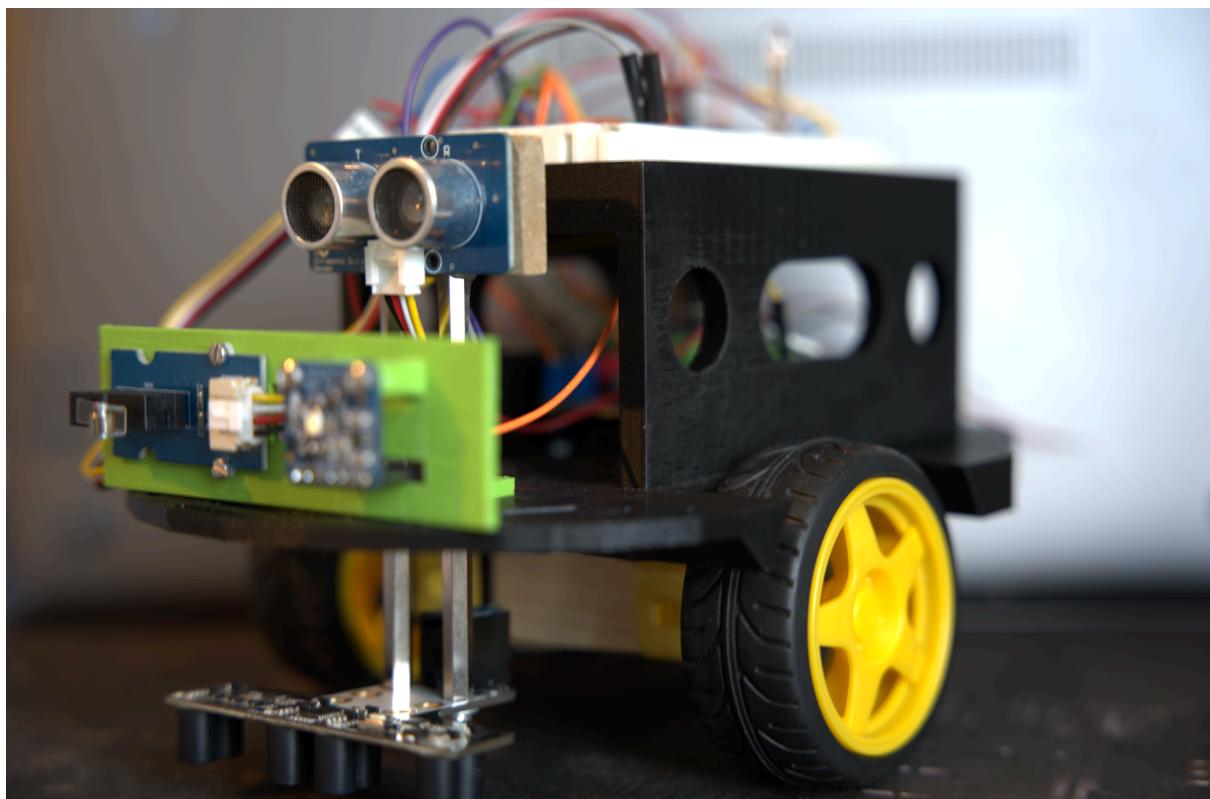


IUT D'EVRY
GEII FA S5
Constantin BALLOT
Quentin CHARRIER

SAE robot suiveur de ligne S5



Sommaire

Introduction	P 3-4
I. Description de fonctionnement	P 5-6
II. RGB line follower - Capteur de ligne	P 7-10
III. Télémètre - capteur ultrason	P 11
IV. TSC34725 - Capteur de couleurs	P 12
V. Contacteur - Capteur de contact	P 13
VI. Moteur encodeur Arduino	P 14-16
VII. Intialisation	P 17
VIII. Etape 1 du scénario	P 18
IX. Etape 2 du scénario	P 19
X. Etape 3 du scénario	P 20
XI. Etape 4 du scénario	P 21
XII. Etape 5 du scénario	P 22
XIII. Validation du montage et de la programmation	P 23
XIV. Conclusion	P 23
XV. Annexe	P 24-27

Introduction

L'objectif de ce projet est de concevoir et de configurer les capteurs afin de permettre au robot de suivre un scénario spécifique.

Voici les différentes étapes du scénario :

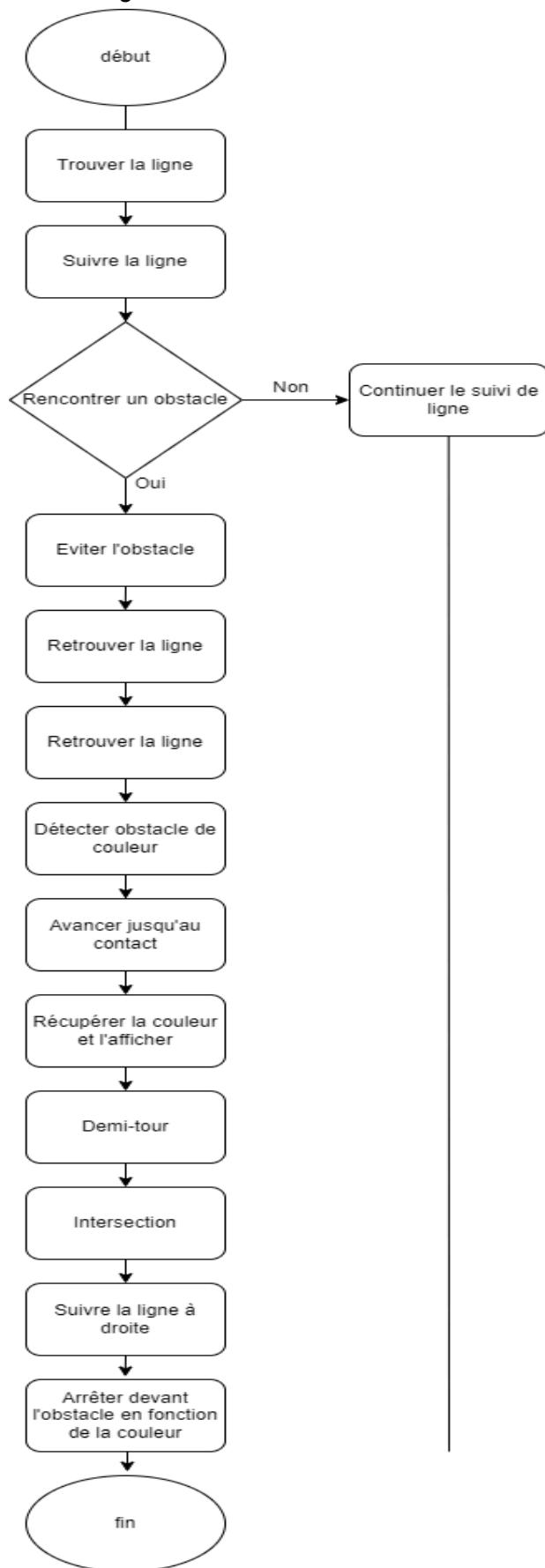
Le robot démarre son parcours et entreprend de trouver la ligne à suivre. Puis il commence à la suivre avec précision, en ajustant sa trajectoire suivant la ligne. Cependant, dès qu'il rencontre un obstacle sur son chemin, le robot contourne l'obstacle puis retrouve sa route sur la ligne.

Après avoir franchi l'obstacle, le robot se concentre à nouveau sur la ligne, la retrouve et reprend son suivi. Lorsqu'il détecte un obstacle de couleur sur son chemin. Le robot avance vers l'obstacle jusqu'à ce qu'il entre en contact avec celui-ci, permettant ainsi au capteur de couleurs de récupérer la couleur de l'obstacle. Une fois la couleur enregistrée, le robot l'affiche fièrement sur sa LED RGB, permettant ainsi une visualisation de la couleur détectée.

Après avoir accompli cette tâche, le robot effectue un demi-tour pour se préparer à revenir sur son chemin. Puis il va rencontrer une intersection où il doit choisir la direction en suivant la ligne la plus à droite.

Enfin, lorsqu'il détecte un obstacle, le robot utilise son capteur ultrason pour estimer la distance d'arrêt en fonction de la couleur enregistrée depuis l'obstacle de couleur.

Soit le diagramme suivant:

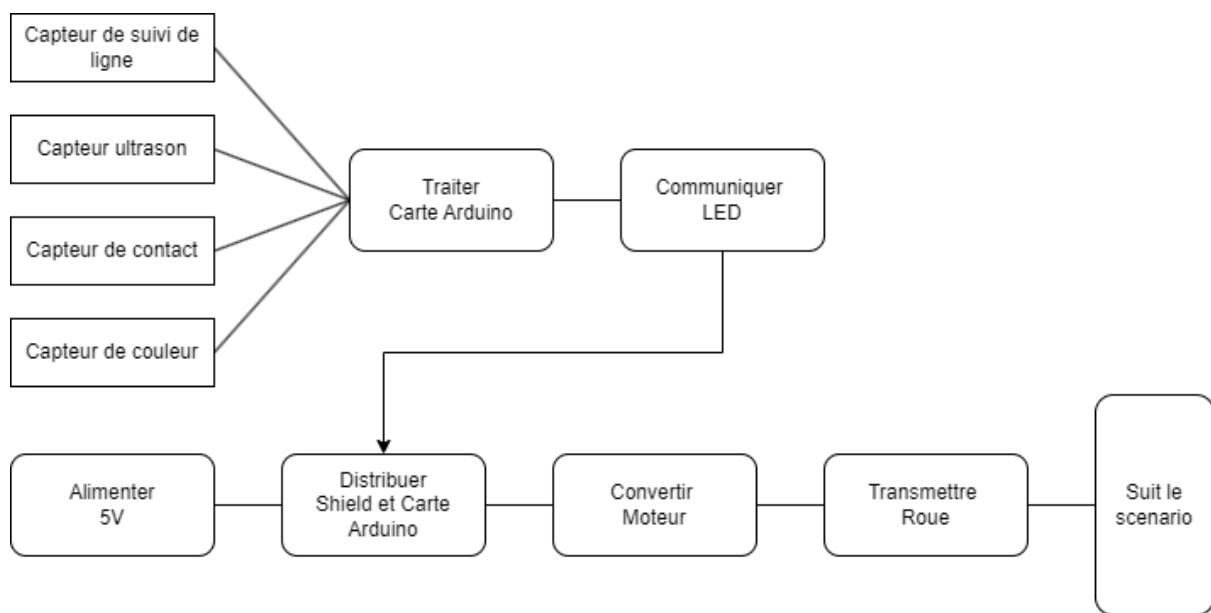


I. Description de fonctionnement

Pour que le scénario puisse fonctionner nous devons avoir:

- un capteur de suivi de ligne
- un capteur ultrason
- un capteur de contact
- un capteur de couleur
- une LED RGB
- une plaque labdek
- une carte Arduino avec Shield

Soit le schéma de fonctionnement du robot:



Pour que le robot puisse fonctionner dans de bonne condition nous avons besoin de ces bibliothèque:

intégration des fichiers externes au code nécessaire :

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <Arduino_GroveI2C_Ultrasonic.h>

#include "Adafruit_TCS34725.h"
#include "MeRGBLineFollower.h"
```

La création des variables nécessaires au bon fonctionnement ainsi qu'à l'appel des programmes externes pour les capteurs qui les nécessite, ici, le suiveur de ligne ainsi que le capteur de couleur.

```
#define ultraPin 9

MeRGBLineFollower RGBLineFollower(13, 12, ADDRESS1);
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);

int color;

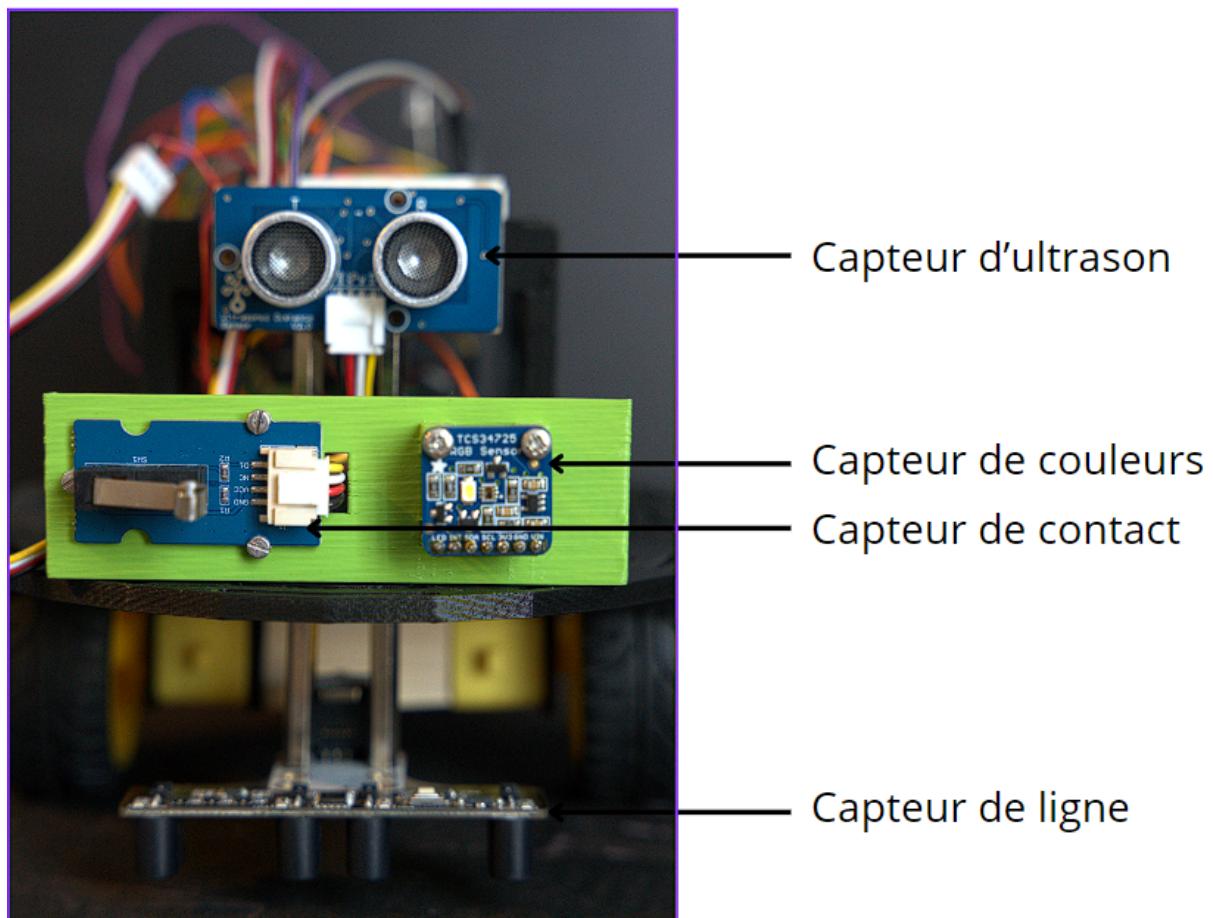
int state = RGBLineFollower.getPositionState();
int state4sensor();

int BPpin = 8;
int BPetat = digitalRead(BPpin);

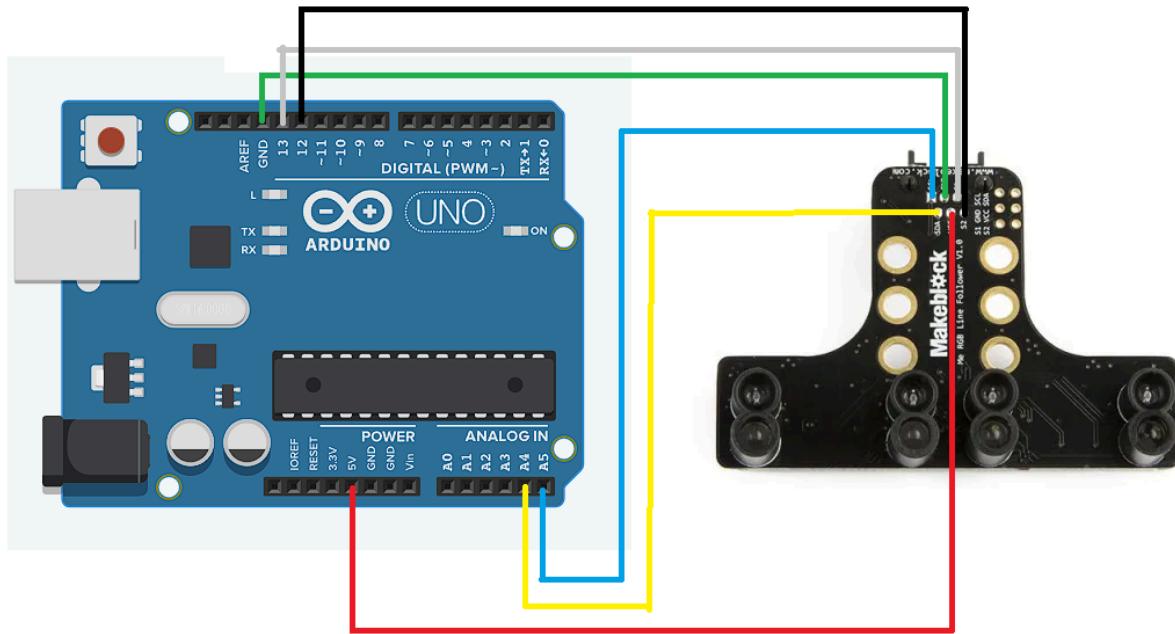
int etapes = 0;

long duration, distance_cm;

int16_t turnoffset = 0;
```



II. RGB line follower - Capteur de ligne



Ce capteur est alimenté en 5V et nous avons branché l'I2C du capteur le SDA et le SDL sur les entrées analogique A4 et A5. Puis nous avons câblé le S1 et S2 du capteur sur les ports 13 et 12 de l'arduino.

Les broches S1 et S2 du capteur sont des ports d'assignation d'adresse d'I2C, la broche SDA est l'interface de données et le SCL l'horloge de l'I2C.

Pour faire le suivi de ligne nous utilisons un capteur de suivi de ligne qui est le RGB line follower qui a plusieurs particularités par rapport aux autres capteurs de suivi de ligne.

-C'est un capteur plus précis car il dispose de 4 capteur de couleurs qui faut calibrer pour l'utilisation

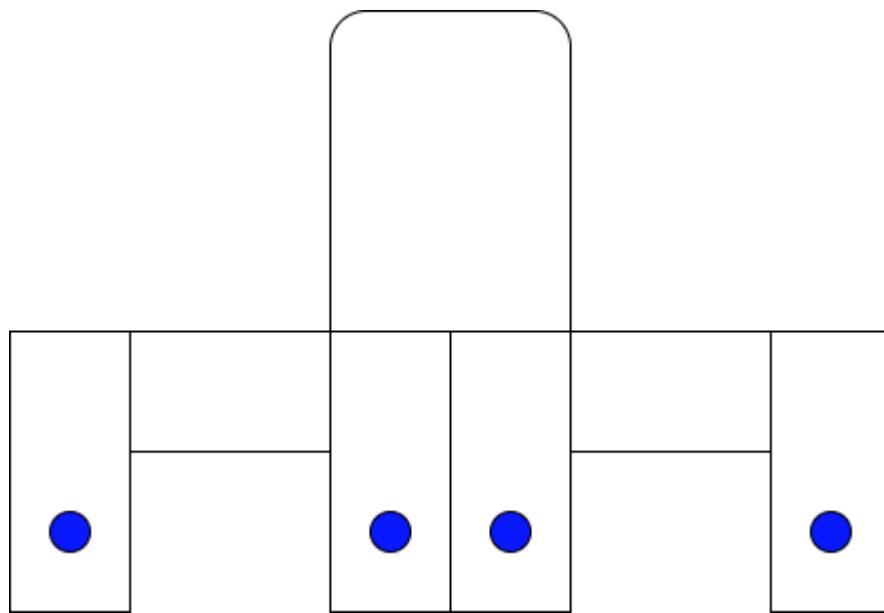
-La communication entre le micro-contrôleur et le capteur se fait en I2C

-Elle dispose de LED RGB pour afficher les couleurs qui sont demandés.

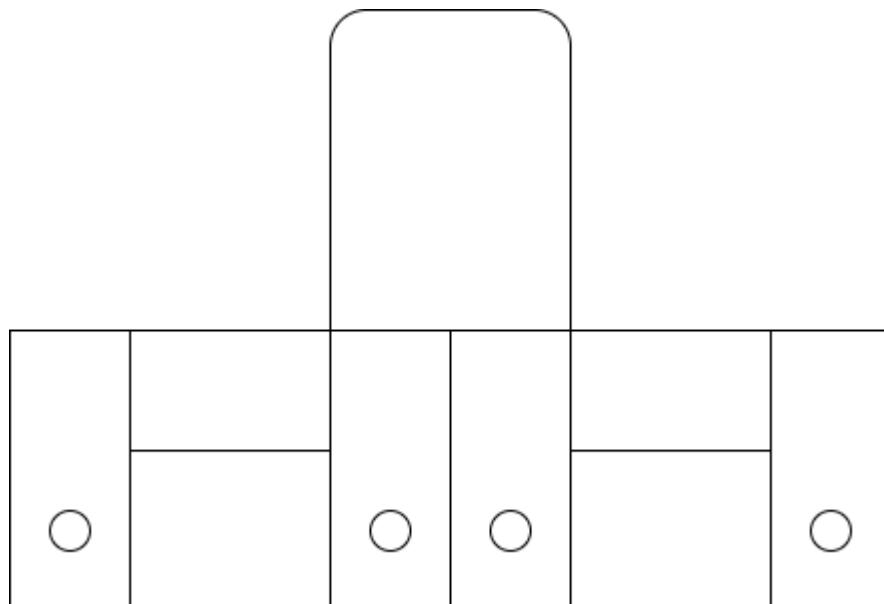
Le capteur de suivis de ligne dispose de 4 capteurs avec une LED qui indique la couleur détectée en fonction du calibrage.

Nous avons configurer pour faire en sorte que dès lors qu'il détecte du noir, les LED s'éteignent et dès lors qu'ils détectent du blanc, elles sont allumé.

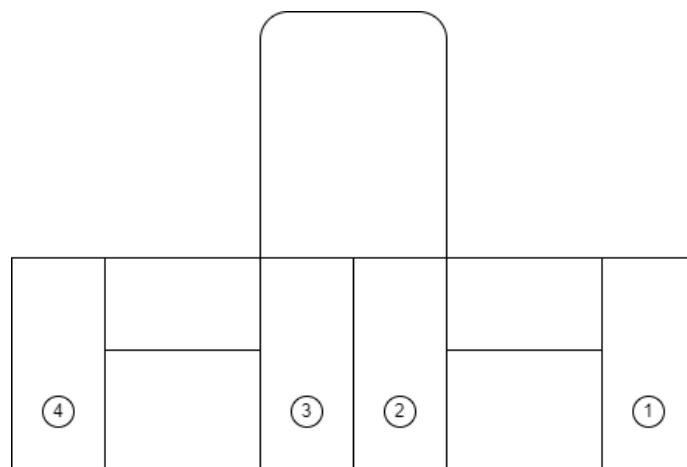
Capteur détectant du blanc:



Capteur détectant du noir:



Pour que le robot puisse suivre la ligne nous allons faire un programme pour faire en sorte que dès que la LED Bleu est éteint sur un des côtés, qu'elle tourne en fonction de cette dernière.



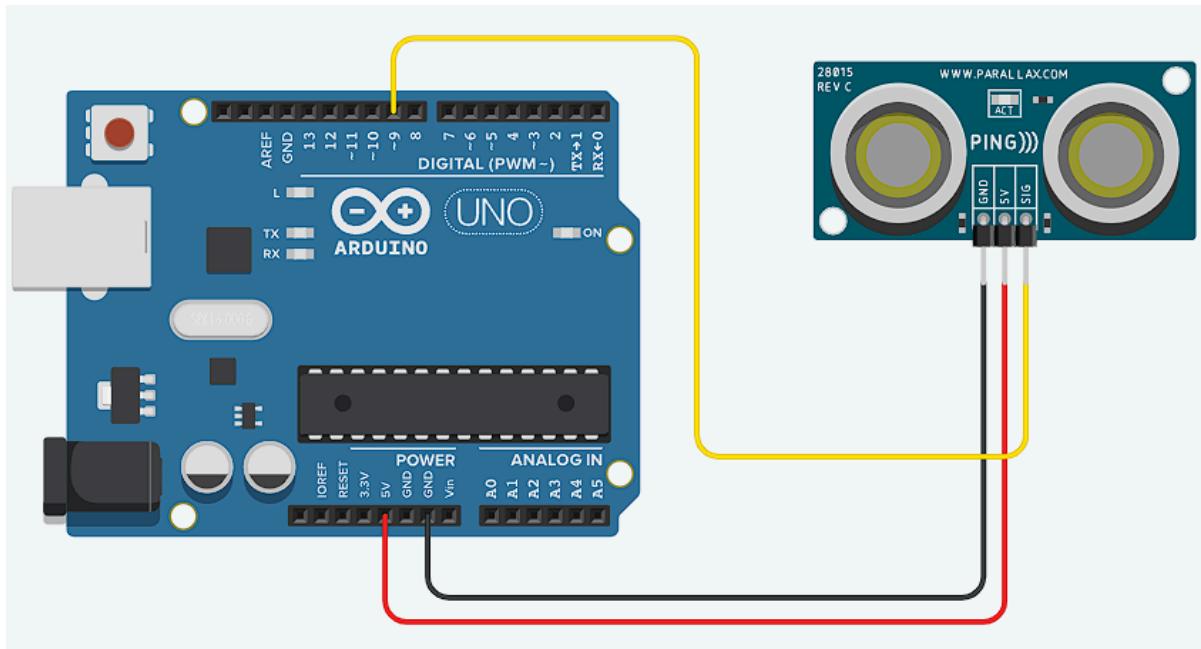
Turnoffset	1	2	3	4	Direction
0					Avancer
1				■	Droite rapide
2			■		Droite
3			■	■	Droite
4		■			Gauche
5		■		■	Gauche
6		■	■		Avancer
7		■	■	■	Droite rapide
8	■				Gauche rapide
9	■			■	Avancer
10	■		■		Droite
11	■		■	■	Droite
12	■	■			Gauche
13	■	■		■	Gauche
14	■	■	■		Récup ligne
15	■	■	■	■	Avancer

Cette fonction du code permet de faire le suivi de ligne et peut être appelé à chaque étape du scénario. d'ailleurs, on y récupère la variable "etapes" pour que la fonction puisse s'adapter à la situation.

On récupère la valeur du capteur de ligne et on applique la condition adéquate pour qu'il se déplace correctement. Le déplacement a été choisi selon le tableau qu'on a réalisé précédemment selon le "turnoffset".

```
void suivre(int etapes) //fonction chargé d'effectuer le suivi de ligne du début à la fin
{
    RGBLineFollower.loop();
    turnoffset = RGBLineFollower.getPositionState();
    Serial.print("ligne : ");
    Serial.print(turnoffset);
    Serial.print("\t");
    if (etapes == 3) //demi-tour après contact du mur
    {
        TCCR0A = 0xA3;
        PORTD &= (1 << 4);
        PORTD |= ~(1 << 7);
        OCR0B = 130;
        OCR0A = 140;
    }
    else if ((turnoffset == 13) or (turnoffset == 12) or (turnoffset == 4) or (turnoffset == 5)) {
        tourner_a_gauche();
    }
    else if ((turnoffset == 11) or (turnoffset == 10) or (turnoffset == 2) or (turnoffset == 3)) {
        tourner_a_droite();
    }
    else if (turnoffset == 1){
        droite_evitement();
        delay(300);
    }
    else if ((turnoffset == 14) and (etapes != 1)){ //récupération de la ligne après évitement de l'obstacle
        gauche_evitement();
        delay(150);
    }
    else if (turnoffset == 7){
        droite_evitement();
        delay(150);
    }
    else if (turnoffset == 8){
        gauche_evitement();
        delay(150);
    }
    else
    {
        avancer();
    }
}
```

III. Télémètre - capteur ultrason



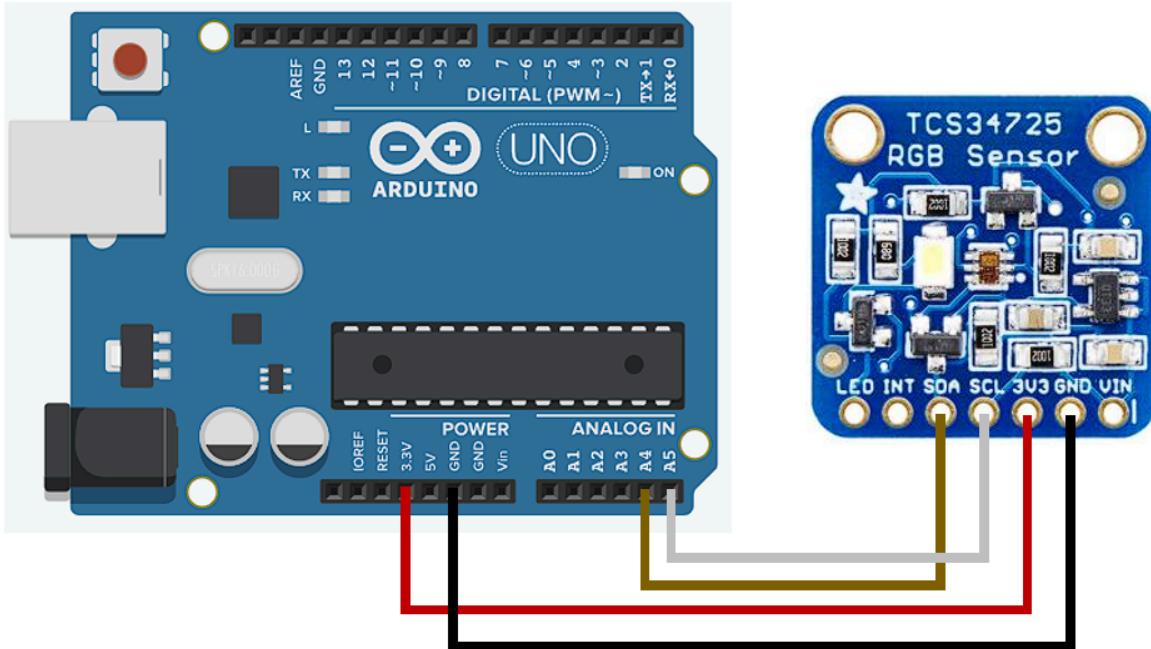
Ce capteur à ultrasons nous permet de déterminer la distance entre les obstacles, notamment pour le premier obstacle lors de l'évitement, puis pour le second lors de l'arrêt. Il fonctionne sous une alimentation de 5V, et nous récupérons les données analogiques du capteur via le port 9 de l'Arduino. Ce capteur présente la particularité de n'avoir qu'une seule sortie de signal, et il est impératif qu'il soit alimenté en 5V pour garantir l'exactitude des mesures de distance.

Ensuite sur arduino nous devons installer une bibliothèque <Arduino_GroveI2C_Ultrasonic.h> pour avoir les valeurs données par le capteur. Puis ensuite nous l'avons directement dans la variable dans "microsecondsToCentimeters(duration)" et nous l'avons directement mis dans "distance_cm".

Voici la fonction nécessaire pour que le traitement d'information venant du télémètre pour ressortir des centimètres que voici :

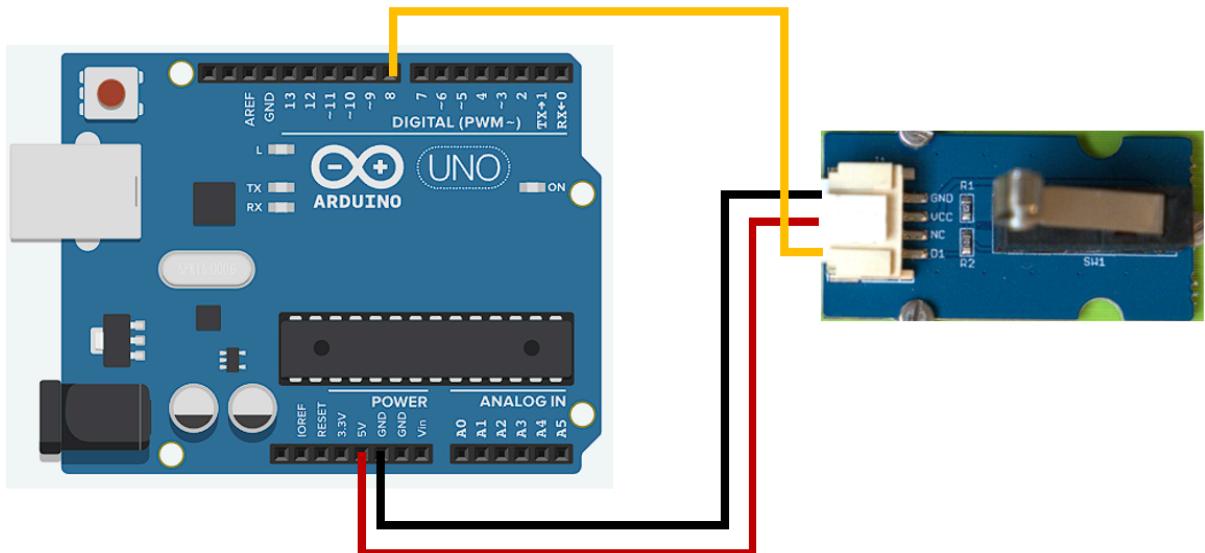
```
long microsecondsToCentimeters(long microseconds) {  
    return microseconds / 29 / 2;  
}
```

IV. TSC34725 - Capteur de couleurs



Le capteur de couleurs TCS 34725 sera utilisé pour identifier la couleur rencontrée au niveau du second obstacle. Il est conçu pour fonctionner sous une alimentation de 3,3V et utilise une communication I2C pour échanger des données avec l'Arduino. Ainsi, tout comme pour le suiveur de ligne, nous avons connecté les broches SDA sur A4 et SCL sur A5. Ce capteur transmet des informations en fonction de l'intensité de couleur détectée devant lui. Pour faire fonctionner le capteur correctement nous avons besoin de la bibliothèque "Adafruit_TCS34725.h".

V. Contacteur - Capteur de contact

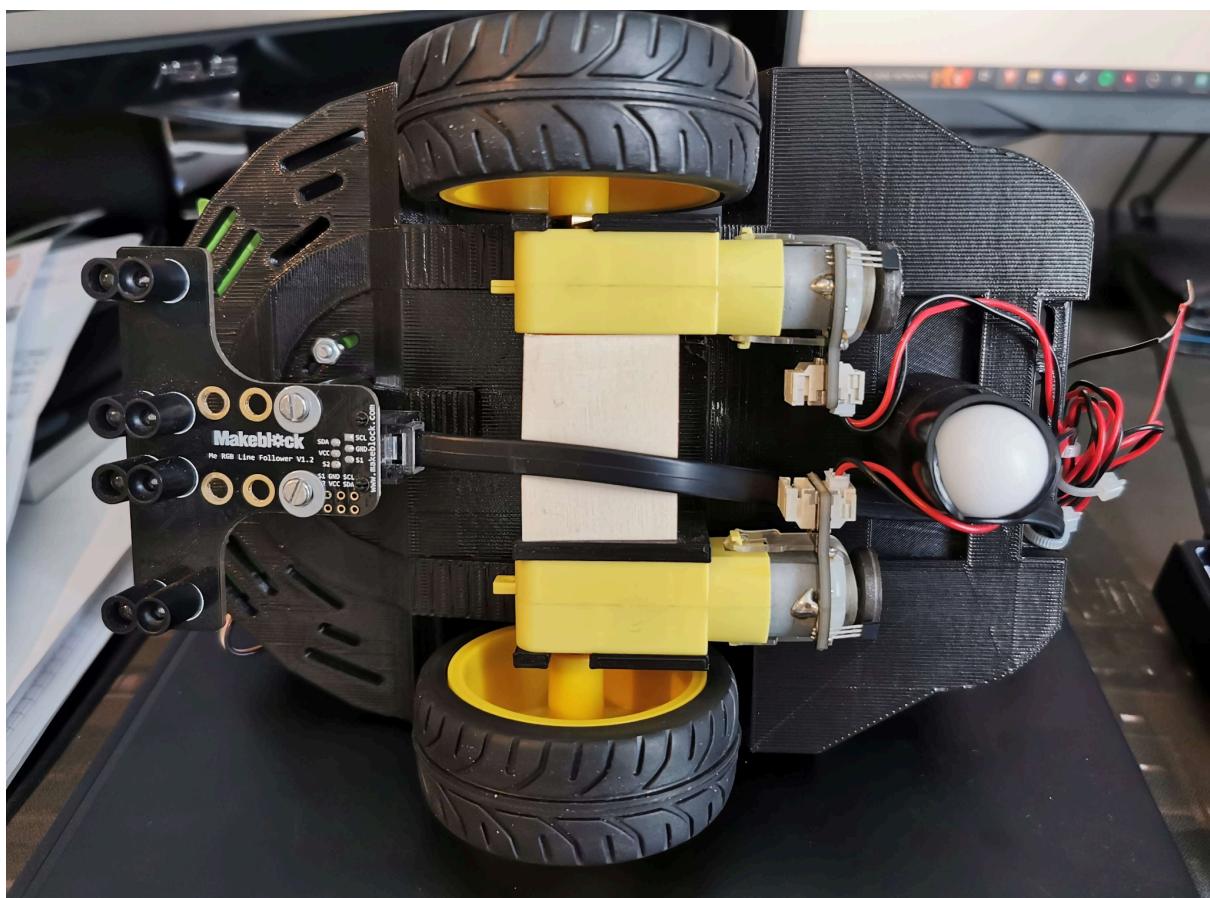


Le capteur de contact est un capteur tout ou rien, fonctionne de la même manière qu'un bouton normalement ouvert. Il fonctionne sous une alimentation de 5V et émet un signal de 5V directement sur la broche 8 de l'arduino dès qu'il détecte un contact. Ce capteur sera employé lorsque le robot entre en contact avec l'obstacle, permettant ainsi au robot de faire demi-tour.

VI. Moteur encodeur



Pour ce projet, nous avons installé des moteurs avec des encodeurs sur nos robots. Cependant, nous avons décidé de ne pas utiliser ces encodeurs, les estimant non nécessaires pour les objectifs=8 que nous nous sommes fixés car nous avons utilisé les timers qui sont plus simples à utiliser et nous avons une meilleure maîtrise de la vitesse. Les moteurs sont situés en dessous de notre robot:



Initialisation des programmes externes pour piloter les capteurs ainsi que le timer0 pour le bon fonctionnement des moteurs.

```
void setup()
{
    pinMode(BPpin, INPUT);
    RGBLineFollower.begin();
    RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);

    Serial.begin(9600);

    DDRB = 0xFE;
    DDRC = 0xE0;
    DDRD = 0x7F;

    TCCR0A = 0xA3;
    TCCR0B = 0x04;

    TCCR1A = 0xA2;
    TCCR1B = 0x1B;

    OCR0A = 50;
    OCR0B = 50;

    ICR1 = 5000;
    OCR1A = 360;

    PORTD |= (1 << 4);
    PORTD |= (1 << 7);

    ADCSRA = 0x87;
}
```

Les déplacements sont gérés selon des fonctions pour chaque sens. on va seulement prendre "avancer" et "reculer" pour expliquer car leur manière de fonctionner sont exactement les mêmes, le seul changement est la vitesse ou le sens des moteurs. Pour OCR0A et OCR0B, on choisit une valeur entre 0 et 255 pour changer le rapport cyclique de l'alimentation du moteur, ce qui permet d'interagir sur leurs vitesses indépendamment. Interagir sur les ports 4 et 7 sur le PORTD permet de choisir le sens de rotation du moteur. Les paramètres attribués aux autres fonctions de déplacement seront visibles sur le document annexe comprenant l'intégralité du code.

```
void avancer()
{
    TCCR0A = 0xA3;
    OCR0A = 115;          //moteur gauche
    OCR0B = 110;          //moteur droit
    PORTD &= ~(1 << 4);  //moteur gauche
    PORTD &= ~(1 << 7);  //moteur droit
    Serial.print("avancer \t");
}

///////////////////////////////
void reculer()
{
    TCCR0A = 0xA3;
    PORTD |= (1 << 4);   //moteur gauche
    PORTD |= (1 << 7);   //moteur droit
    OCR0A = 110;          //moteur gauche
    OCR0B = 120;          //moteur droit
    Serial.print("reculer \t");
}
```

Étape du scénario

VII. Initialisation

On passe ensuite à l'intérieur du void loop() qui comprend l'intégralité du scénario.

On commence tout d'abord par l'initialisation du télémètre car, après quelques tests, il renvoyait des valeurs erronées qui faussent la distance et qui cassait le scénario. Cette boucle ne sert donc que de stabilité au programme.

```
while (distance_cm == 0)      // INITIALISATION ULTRASON, sécurité pour ne pas échapper l'étape 1
{
    delay(100);
    pinMode(ultraPin, OUTPUT);
    digitalWrite(ultraPin, LOW);
    delayMicroseconds(2);
    digitalWrite(ultraPin, HIGH);
    delayMicroseconds(5);
    digitalWrite(ultraPin, LOW);
    pinMode(ultraPin, INPUT);

    duration = pulseIn(ultraPin, HIGH);

    distance_cm = microsecondsToCentimeters(duration);

    Serial.print("distance : ");
    Serial.print(distance_cm);
    Serial.println("\t");
}
```

VIII. Etape 1

On passe donc à la première étape. Dans cette dernière, on utilise donc le capteur ultrason et on y retrouve la fonction "suivi()" qui permet le suivi de ligne. Cette étape s'arrête lorsque le robot se trouve à 25 cm de l'obstacle d'où la condition if. Si cette condition est remplie, on passe à l'étape suivante.

```
////////////////////////////// ETAPE 1 ///////////////////////////////
while (etapes == 0)      // Suivi de ligne jusqu'à rencontre de l'obstacle détecté par le télémètre
{
    delay(20);
    pinMode(ultraPin, OUTPUT);
    digitalWrite(ultraPin, LOW);
    delayMicroseconds(2);
    digitalWrite(ultraPin, HIGH);
    delayMicroseconds(5);
    digitalWrite(ultraPin, LOW);

    pinMode(ultraPin, INPUT);
    duration = pulseIn(ultraPin, HIGH);
    distance_cm = microsecondsToCentimeters(duration);

    Serial.print("distance : ");
    Serial.print(distance_cm);
    Serial.println("\t");

    suivi(0);

    if (distance_cm <= 25){      //si distance inférieur à 25cm de l'obstacle, changement d'étape
        etapes = 1;
    }
    Serial.print("etape 1 \t");
    Serial.print("\n");
}
```

entre l'étape 1 et l'étape 2, on retrouve une partie qui s'occupe du contournement de l'obstacle avant de passer à l'étape 2 :

```
droite_evitemen();      //contournement
delay(500);
evitemen_obstacle();
delay(600);
delay(600);
```

IX. Etape 2

L'étape 2 permet de se ré-embrayer avec la ligne à suivre pour continuer le scénario. elle permet donc d'avancer et de passer à la suite dès que la ligne est détectée de nouveau :

```
////////////////// ETAPE 2 /////////////////////////////////
while (etapes == 1)           // Contournement de l'obstacle jusqu'à ce qu'une ligne soit détecté
{
    suivi(etapes);          //dès que la ligne est récupéré, il se ré axe dans son sens
    if (turnoffset < 15)
    {
        etapes = 2;
        droite_evitement();
        delay(300);
    }
    Serial.print("etape 2 \t");
    Serial.print("\n");
}
```

X. Etape 3

L'étape 3 consiste à continuer de suivre la ligne jusqu'au contact d'un mur. On retrouve donc le suivi de ligne avec une condition pour passer à l'étape suivante lorsque le bouton à l'avant du robot est pressé.

```
////////////////////////////// ETAPE 3 //////////////////////////////
while (etapes == 2)          // Suivi de ligne jusqu'à ce que l'obstacle soit détecté par le bouton
{
    suivi(0);
    int BPetat = digitalRead(BPpin);
    if (BPetat == HIGH)
    {
        Serial.print("on\t");
        etapes = 3;
    }
    else {
        Serial.print("off\t");
    }

    Serial.print("etape 3 \t");
    Serial.print("\n");
}
```

On retrouve ensuite une étape intermédiaire qui s'occupe de récupérer la couleur du mur via le capteur de couleur. Ce capteur récupère l'intensité détectée pour chaque couleur primaire (rouge, vert et bleu). on a donc créé une condition pour récupérer la couleur dominante entre les 3 couleurs primaires pour ensuite enregistrer celle affichée sur le mur sachant qu'elle peut varier entre ces 3 couleurs. Cette variable servira à la fin du scénario.

```
float red, green, blue;    //utilisation du capteur de couleur
delay(60);    // takes 50ms to read
tcs.getRGB(&red, &green, &blue);
Serial.print("R:\t"); Serial.print(int(red));
Serial.print("\tG:\t"); Serial.print(int(green));
Serial.print("\tB:\t"); Serial.println(int(blue));
if (red>(green+20) && red>(blue+20))
{
    color = 1;      //red
    RGBLineFollower.setRGBColour(RGB_COLOUR_RED);
}
else if (blue>(red+20) && blue>(green+20))
{
    color = 2;      //blue
    RGBLineFollower.setRGBColour(RGB_COLOUR_BLUE);
}

else if (green>(red+20) && green>(blue+20))
{
    color = 3;      //green
    RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);
}
```

XI. Etape 4

Une fois la couleur récupérée, le robot doit faire demi-tour ce qui est effectué en première partie par ces lignes. il permet seulement d'amorcer la manœuvre pour que le capteur suivi de ligne en sorte pour que l'étape suivante ne se trompe pas de sens.

```
reculer(); //amorçage du demi-tour
delay(300);

TCCR0A = 0xA3; //début demi-tour
PORTD &= (1 << 4);
PORTD |= ~(1 << 7);
OCR0B = 130;
OCR0A = 140;
delay(150);
```

l'intérêt de l'amorçage se comprend d'autant plus lorsqu'on voit que l'étape 4 consiste à continuer de tourner tant que la ligne n'est pas détectée par le capteur. Une fois la ligne retrouvée, on passe à la fin du scénario, l'étape 5.

```
////////////////////////////// ETAPE 4 //////////////////////////////
while (etapes == 3) // demi-tour
{
    suivie(etapes);

    turnoffset = RGBLineFollower.getPositionState();

    if (turnoffset < 14){
        etapes =4;
        /*gauche_evitement();
        Serial.print("Réaxement \t");
        delay(500); */
    }

    Serial.print("etape 4 \t");
    Serial.print("\n");
}
```

XII. Etape 5

Cette dernière étape réutilise le capteur de distance pour s'arrêter à la bonne distance du mur selon la couleur détectée sur le mur coloré précédent. Il y a également une séparation du chemin sur le retour et le scénario consiste à emprunter le chemin de droite. Lorsque le robot arrive au croisement de ce chemin, la valeur du capteur suivi de ligne a un “turnoffset” de 1. Dans la fonction “suivi();”, on retrouve donc une condition spécifique pour ce cas, qui permet au suivi de ligne de prioriser la voie de droite :

```
else if (turnoffset == 1){  
    droite_evitement();  
    delay(300);  
}
```

une fois ce chemin emprunté il ne reste plus qu'à s'arrêter à la bonne distance du mur. C'est pour ça qu'une condition de distance a été écrite pour chaque couleur pour que le robot s'arrête au bon moment quelque soit la couleur. Si l'une des 3 conditions est remplie, la ligne “break;” est exécutée, ce qui permet de sortir de la boucle “étape 5” et de rentrer dans le while(1) pour la fin du scénario et ce qui permet de le bloquer jusqu'au prochain reset.

```
////////////////// ETAPÉ 5 ///////////////////  
while(etapes == 4) // suivi de ligne en priorisant la ligne de droite  
{  
    Serial.print("etape 5 \t");  
    Serial.print("\n");  
  
    delay(20); //récupération de la distance pour s'arreter au bon moment du mur  
    pinMode(ultraPin, OUTPUT);  
    digitalWrite(ultraPin, LOW);  
    delayMicroseconds(2);  
    digitalWrite(ultraPin, HIGH);  
    delayMicroseconds(5);  
    digitalWrite(ultraPin, LOW);  
    pinMode(ultraPin, INPUT);  
    duration = pulseIn(ultraPin, HIGH);  
    distance_cm = microsecondsToCentimeters(duration);  
    Serial.print("distance : ");  
    Serial.print(distance_cm);  
    Serial.println("\t");  
  
    suivi(0);  
  
    if ((distance_cm <= 40) and (color == 2)){ // arret sur blue 40cm  
        arret();  
        break;  
    }  
    else if ((distance_cm <= 30) and (color == 1)){ // arret sur rouge 30cm  
        arret();  
        break;  
    }  
    else if ((distance_cm <= 20) and (color == 3)){ // arret sur vert 20cm  
        arret();  
        break;  
    }  
}  
while(1);
```

XIII. Validation du montage et de la programmation.

Le robot est entièrement fonctionnel et répond de manière satisfaisante à toutes les exigences du cahier des charges. Il exécute avec succès chaque étape du scénario établi, validant ainsi son bon fonctionnement et sa capacité à accomplir les tâches définies.

XIV. Conclusion et perspective

Le robot répond bien au cahier des charges, cependant des améliorations peuvent être possibles sur notre robot.

Premièrement, il serait judicieux d'améliorer son alimentation en utilisant une batterie ou des piles embarquées, ce qui pourrait accroître son autonomie et sa portabilité.

Ensuite, il est nécessaire de revoir le design du robot afin de rendre l'accès au module Arduino plus accessible. Actuellement, en cas de problème, il est difficile de dépanner le système en raison de cette limitation d'accès.

Une autre piste d'amélioration serait d'explorer des solutions pour alléger la structure du robot. Cela pourrait permettre d'améliorer son agilité et sa maniabilité.

Enfin, il pourrait être bénéfique d'augmenter la vitesse du robot pour améliorer son efficacité dans diverses applications. Cela nécessiterait une réévaluation de ses composants et de son système de propulsion.

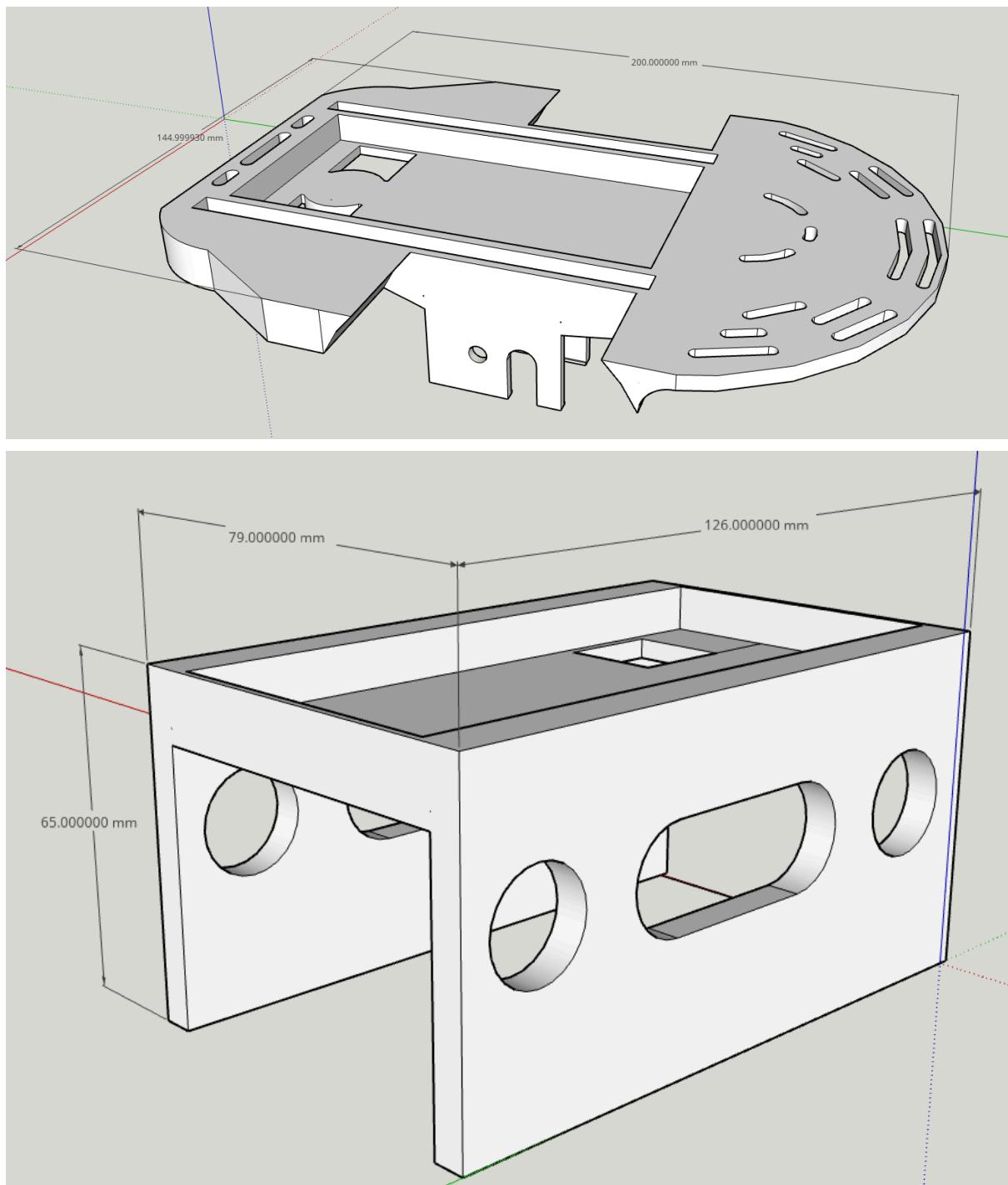
XV. ANNEXE

Chiffrage totale du robot:

Désignation	Quantité	Total HT	Total TTC
Motoréducteur 1:120	2	14,24 €	17,80 €
Paire de roue 65 mm	1	4,70 €	5,88 €
Roue libre avec bille plastique	1	5,20 €	6,50 €
Arduino Uno REV3	1	19,20 €	24,00 €
Arduino Moto Shield	1	23,04 €	28,80 €
Chassis robot	1	8,00 €	10 €
Me RGB Line Follower P3030001	1	14,78 €	18,48 €
Télémètre	1	3,84 €	4,80 €
Contacteur	1	2,13 €	2,66 €
CAPTEUR DE COULEUR RGB TCS34725	1	7,63 €	9,54 €
plaqué labdec	1	5,52 €	6,90 €
Total		108,29 €	135,36 €

Caractéristique du robot:

Dimension du châssis du robot



Puissance moyenne du robot: 1,75 W

Poid du robot: 616 g

Soit le rapport Poids/Puissance: $R = \frac{1,75}{0,616} = 2,84 \text{ W/Kg}$

