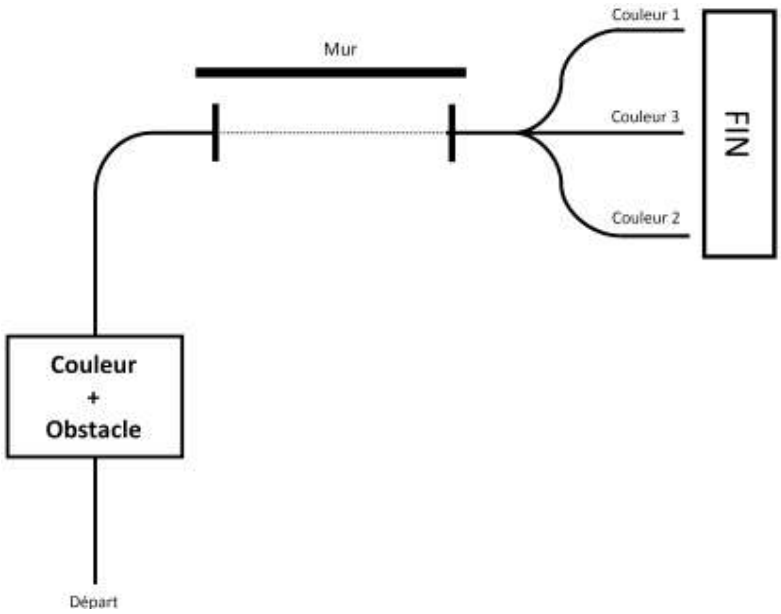
Projet robot SAE semestre 4

KUZDOWICZ Théo
HADDOU Bilal

- Introduction

L'objectif de notre projet consiste à réaliser un robot suiveur de ligne pouvant contourner des obstacles tout en récupérant la couleur de celui-ci, faire un suivi de mur sans ligne, récupérer la ligne et lorsqu'il rencontre trois chemins il doit choisir le chemin de la couleur qu'il a récupérer sur l'obstacle.
Cahier des charges :
Le robot doit fonctionner en suiveur de ligne avec des fonctions avancer et tourner.
Il possède les éléments suivants :
- un télémètre orientable via un servomoteur.
- un capteur de couleur.
- deux moteurs.
- un interrupteur.
- un capteur de ligne.
- une carte Arduino Uno (µC Atmega 328P)
- une carte Arduino motor shield.

Au départ le robot suit la ligne à l'aide du capteur de ligne, il ralentit à l'approche d'un obstacle, il actionne l'interrupteur à l'avant du robot avec l'obstacle, il récupère la couleur de l'obstacle, puis entame l'évitement de l'obstacle, il récupère ensuite la ligne pour faire le suivi de ligne jusqu'à une ligne perpendiculaire qui permettra au robot de savoir qu'il n'y a plus de ligne afin de suivre le mur à une distance constante jusqu'à une nouvelle ligne perpendiculaire où le robot reprendra la ligne jusqu'au moment où trois chemin différents seront proposer où le robot choisira le chemin de la couleur de l'obstacle.
Le robot s'arrêtera enfin à une certaine distance du mur de fin.



- Description du fonctionnement
- Schéma fonctionnel

Schéma bloc moteurs :

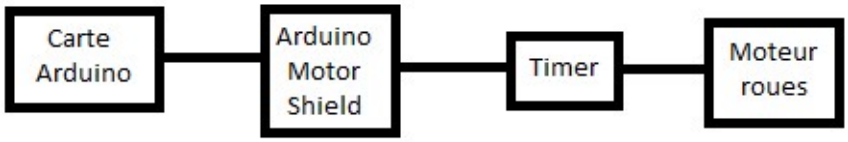


Schéma bloc télémètre :

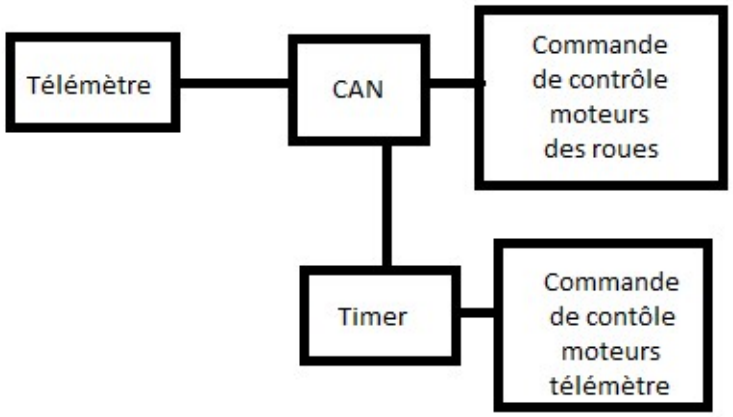


Schéma bloc capteur de couleur :

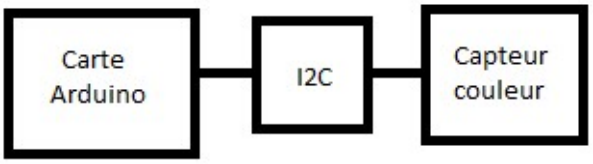


Schéma bloc interrupteur :

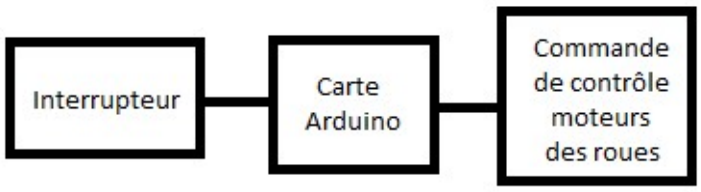
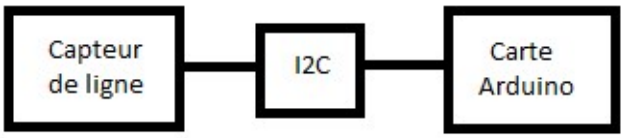


Schéma bloc capteur de ligne :



- Explication du principe du fonctionnement

Schéma bloc moteur : Cette partie sert à faire tourner faire fonctionner les roues afin de le faire avancer, reculer, tourner à droite, tourner à gauche ou encore d'arrêter le robot.
Schéma bloc capteur de ligne : Cette partie sert à commander les moteurs du robot en fonction des quatre capteurs du capteur de ligne qui sont sous le robot pour le permettre de suivre la ligne.
Schéma bloc du capteur et du moteur du télémètre : Cette partie sert à commander les moteurs du télémètre en fonction du capteur du télémètre.
Schéma bloc interrupteur : Cette partie sert à faire une impulsion pour rentrer en contact avec l'obstacle.
Schéma bloc capteur de couleur : Cette partie permet de récupérer la couleur de l'obstacle afin de permettre au robot de choisir le chemin de la couleur.

- Suivi de ligne

```
// Ci-dessous nous avons les bibliothèques nécessaires pour le code
#include <Arduino.h>
#include <Wire.h> // Pour la communication en I2C
#include <SoftwareSerial.h> // Pour la communication série logicielle
#include "MeRGBLineFollower.h" // Pour le capteur de suivi de ligne RGB

MeRGBLineFollower RGBLineFollower(13,12, 0x20); // Broches 12 et 13 pour le capteur

// Variables nécessaires pour le capteur de suivi de ligne
int16_t turnoffset = 0;
uint8_t sensorstate;
int16_t set_speed = 0;
uint8_t study_types = 0;
int16_t Etat = 0;

// Le setup() permet d'exécutée le programme dans celui-ci au démarrage.
void setup()
{
 Serial.begin(9600);

 RGBLineFollower.begin(); // Pour initialiser le capteur de suivi de ligne
 RGBLineFollower.setKp(0.3);
 RGBLineFollower.getPositionOffset();

 DDRD = 0xFF; // Broches pour les roues

 TCCR0A = 0xA3; // Registres pour les roues
 TCCR0B = 0x04; // Registres pour les roues

 ICR1 = 40000; // Pour la période du signal
 }

// La fonction loop() permet d'exécuter le programme à l'intérieur en boucle
void loop()
{
 SuiviDeLigne();
}

// La fonction Arret() va nous permettre d'arrêter les moteurs des roues
void Arret()
{
 OCR0B = 0; // Roue gauche à l'arrêt
 OCR0A = 0; // Roue droite à l'arrêt
 PORTD &= ~(1<<PD4); // Broches à l'état bas
 PORTD &= ~(1<<PD7); // Broches à l'état bas
 PORTD &= ~(1<<PD5); // Broches à l'état bas
 PORTD &= ~(1<<PD6); // Broches à l'état bas
}
```

```
// La fonction DriftDroite() va nous permettre de tourner à droite tout en avançant
void DriftDroite()
{
  OCR0B = 200; // Roue gauche à la vitesse maximum
  OCR0A = 50; // Roue droite à la vitesse minimale
  PORTD |= (1<<PD4); // Broches à l'état haut
  PORTD &= ~(1<<PD7); // Broches à l'état bas
}

// La fonction DriftGauche() va nous permettre de tourner à gauche tout en avançant

void DriftGauche()
{
  OCR0B = 50; // Roue gauche à la vitesse minimale
  OCR0A = 200; // Roue droite à la vitesse maximum
  PORTD |= (1<<PD4); // Broches à l'état haut
  PORTD &= ~(1<<PD7); // Broches à l'état bas
}

// La fonction Avancer() va nous permettre de faire rouler le robot en ligne droite
void Avancer ()
{
  OCR0B = 200; // Roue gauche à la vitesse maximum
  OCR0A = 200; // Roue droite à la vitesse maximum
  PORTD |= (1<<PD4);  // Broches à l'état haut
  PORTD &= ~(1<<PD7); // Broches à l'état bas
}

// La fonction Droite() va nous permettre de tourner à droite sur place
void Droite ()
{
  OCR0B = 0; // Roue gauche à l'arrêt
  OCR0A = 200; // Roue droite à la vitesse maximum
  PORTD |= (1<<PD4); // Broches à l'état haut
  PORTD |= (1<<PD7); // Broches à l'état haut
}

// La fonction Gauche() va nous permettre de tourner à gauche sur place
void Gauche ()
{
  OCR0B = 200; // Roue gauche à la vitesse maximum
  OCR0A = 0; // Roue droite à l'arrêt
  PORTD &= ~(1<<PD4); // Broches à l'état bas
  PORTD |= (1<<PD7); // Broches à l'état haut
}

// La fonction Reculer() va nous permettre de faire reculer le robot en ligne droite
void Reculer ()
{
```

```
  OCR0B = 200; // Roue gauche à la vitesse maximum
  OCR0A = 200; // Roue droite à la vitesse maximum
  PORTD &= ~(1<<PD4); // Broches à l'état bas
  PORTD |= (1<<PD7); // Broches à l'état haut
}

// Ici nous utilisons la fonction qui permet de faire le suivi de ligne
void SuiviDeLigne()
{
  if ((Etat == 0x09)|(Etat == 0x0F)|(Etat == 0x05)|(Etat == 0x06)|(Etat == 0x0A)|(Etat ==
0x02)|(Etat == 0x04)|(Etat == 0)|(Etat == 0x08)|(Etat == 0x01))
  {
    Avancer();
  }
  else if ((Etat == 0x0C)|(Etat == 0x0E))
  {
    Gauche();
  }
  else if ((Etat == 0x03)|(Etat == 0x07))
  {
    Droite();
  }
  else if ((Etat == 0x0D))
  {
    DriftGauche();
  }
  else if ((Etat == 0x0B))
  {
    DriftDroite();
  }
}
```

Dans ce code-ci nous faisons fonctionner le robot en mode suivi de ligne grâce au RGBLineFollower que nous faisons fonctionner avec les « Etat » qui fonctionnent ici en hexadécimal.
Le RGBLineFollower fonctionne avec des LED qui s'allument lorsqu'ils ne détectent pas de ligne et éteint lorsqu'il y a une ligne.
Lorsque toutes les LED sont allumées nous avons 0x0F et lorsque tout est éteint nous avons 0.
De face le capteur de droite correspond au premier, si c'est le seul allumer c'est 0x01 et ainsi de suite.
Le capteur RGBLineFollower est initialisé avec les broches 13 et 12 pour la communication.
Nous avançons donc en permanence et lorsque c'est nécessaire le robot corrige sa trajectoire.

- Le robot détecte un obstacle
- Le robot réduit sa vitesse à l'approche de l'obstacle

Pour permettre à mon robot de ralentir à l'approche de l'obstacle j'ajoute le télémètre qui va récupérer la distance

```
int T = 0; // J'initialise le télémètre à 0

//Dans le setup()
DDRC = 0; // Broches pour le télémètre
TCCR1A = (1<<WGM11)|(1<<COM1B1)|(1<<COM1A1); // Registres pour le télémètre
TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS01); // Registres pour le télémètre

int Telemetre()
{
 ADMUX = (1<<ADLAR);
 ADMUX = ADMUX&~(1<<MUX0);
 ADMUX = ADMUX|(1<<MUX1);
 ADCSRA = ADCSRA|(1<<ADSC);
 while((ADCSRA&(1<<ADIF)) == 0) ;
 ADCSRA = ADCSRA|(1<<ADIF);
 if (ADCH > 40)    //  ADCH > 40 équivaut à peu près à 15cm
 {
  return 1;   // Renvoie 0 si la valeur est inférieure ou égale à 40
 }
 else
 {
  return 0;    // Renvoie 1 si la valeur est supérieure à 40
 }
}
```

Une fois le télémètre ajouté je rajoute des fonctions pour les moteurs pour qu'ils roulent lentement.

```
void AvancerLow()
{
 OCR0B = 75; // Roue gauche à la vitesse minimale
 OCR0A = 75; // Roue droite à la vitesse minimale
 PORTD |= (1<<PD4);  // Broches à l'état haut
 PORTD &= ~(1<<PD7); // Broches à l'état bas
}

// Dans la fonction SuiviDeLigne()
 else if ((Etat == 0x0E)&&(T == 1))
 {
  GaucheLow();
 }
```

Dans le code principal j'ajoute donc le télémètre qui va récupérer en permanence la distance en face de lui, lorsqu'il détecte l'obstacle à moins de 15cm il nous retourne un 1 et un 0 si l'obstacle est loin.
Dans les conditions pour effectuer les actions j'ajoute le télémètre, il doit être à 1 pour

ralentir le robot et à 0 pour rouler plus rapidement.

- Le robot rentre en collision avec l'obstacle.

Pour la collision avec l'obstacle nous avons un interrupteur à l'avant du robot.

int I = 0; // J'initialise l'interrupteur à 0

```
int BrocheInterrupteur = 8; // Je déclare l'interrupteur sur la broche 8 de l'Arduino

// Dans le setup()
pinMode(BrocheInterrupteur, INPUT); // Pour mettre l'interrupteur en entrée

void loop()
{
  Main();
}

void Main()
{
  RGBLineFollower.loop();
  Etat = RGBLineFollower.getPositionState();
  I = digitalRead(BrocheInterrupteur);   // Lit l'état de la broche et stock le résultat dans la
variable I.
  if (I == HIGH)   // Si interrupteur est activé
  {
    Arret();
  }
  else if (I == LOW)   // Si interrupteur est désactivé
  {
    SuiviDeLigne();
  }
}
```

Le robot suit la ligne ralentit à l'approche de l'obstacle puis si l'interrupteur est désactivé, il continu le suivi de ligne et s'il est activé le robot s'arrête.

- Récupération de la couleur de l'obstacle

Nous ajoutons ensuite le capteur de couleur, le TCS34725 qui va nous permettre de récupérer la couleur de l'obstacle pour choisir le chemin correspondant à la couleur.

```
#include "Adafruit_TCS34725.h" // Bibliothèque pour contrôler le capteur de couleur
```

```
int Chemin = 0; // Initialisation du Chemin à 0

float red, green, blue; // Pour stocker les valeurs des couleurs mesurées

#define redpin 3
#define greenpin 5
#define bluepin 6
#define commonAnode true

byte gammatable[256];

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_4X);

//Dans le setup()
if (tcs.begin()) // Pour initialiser le capteur de couleur
 {
 }
 else
 {
  while (1);
 }
 #if defined(ARDUINO_ARCH_ESP32)
  ledcAttachPin(redpin, 1);
  ledcSetup(1, 12000, 8);
  ledcAttachPin(greenpin, 2);
  ledcSetup(2, 12000, 8);
  ledcAttachPin(bluepin, 3);
  ledcSetup(3, 12000, 8);
 #else
  pinMode(redpin, OUTPUT);
  pinMode(greenpin, OUTPUT);
  pinMode(bluepin, OUTPUT);
 #endif
  for (int i=0; i<256; i++)
  {
   float x = i;
   x /= 255;
   x = pow(x, 2.5);
   x *= 255;
   if (commonAnode)
   {
    gammatable[i] = 255 - x;
   }
   else
   {
    gammatable[i] = x;
   }
 }
```

```
//Dans le Main()
if ((red>130)&&(green<95)&&(blue<75)) // Si le rouge est le plus présent
  {
    Chemin = 1;
  }
  else if ((green>95)&&(red<130)&&(blue<75)) // Si le vert est le plus présent
  {
    Chemin = 2;
  }
  else if ((blue>75)&&(green<95)&&(red<130)) // Si le bleue est le plus présent
  {
    Chemin = 3;
  }
}

//Pour lire les valeurs des composantes rouge, verte et bleue
void Couleur()
{
 tcs.setInterrupt(false);
 delay(60);
 tcs.getRGB(&red, &green, &blue);
 tcs.setInterrupt(true);
#if defined(ARDUINO_ARCH_ESP32)
 ledcWrite(1, gammatable[(int)red]);
 ledcWrite(2, gammatable[(int)green]);
 ledcWrite(3, gammatable[(int)blue]);
#else
 analogWrite(redpin, gammatable[(int)red]);
 analogWrite(greenpin, gammatable[(int)green]);
 analogWrite(bluepin, gammatable[(int)blue]);
#endif
}
```

Nous ajoutons donc ce code qui va nous permettre de récupérer la couleur de l'objet lorsque l'interrupteur est activé.
Selon la couleur plus présente le « Chemin » change de valeur et nous savons donc de quelle couleur est l'obstacle.
Nous pouvons le faire en ajoutant dans le loop() : Serial.print(Chemin) qui nous retournera dans le moniteur série la valeur de « Chemin ».

- Evitement de l'obstacle

```
int Obstacle = 0;
int EvitementObstacle = 0;

//Dans le setup()
DDRB = 0xFE;   //Servo moteur du telemetre
TBase();

void Main()
```

```
{
  RGBLineFollower.loop();
  Etat = RGBLineFollower.getPositionState();
  I = digitalRead(BrocheInterrupteur);
  if ((I == HIGH)&&(Obstacle == 0))
  {
    Arret();
    delay(250);
    Obstacle = 1;
  }
  else if ((I == LOW)&&(Obstacle == 0))
  {
    RGBLineFollower.setRGBColour(RGB_COLOUR_BLUE);
    SuiviDeLigne();
    Obstacle = 0;
  }
  else if (Obstacle == 1)
  {
    Arret();
    delay(250);
    Couleur();
    if ((red>130)&&(green<95)&&(blue<75))
   {
     Chemin = 2;
   }
   else if ((green>95)&&(red<130)&&(blue<75))
   {
     Chemin = 1;
   }
   else if ((blue>75)&&(green<95)&&(red<130))
   {
     Chemin = 3;
   }
    Obstacle = 2;
  }
  else if (Obstacle == 2)
  {
     Evitement();
  }
}


void Evitement()
{
  T = Telemetre();
  if ((T == 1)&&(EvitementObstacle == 0))
  {
    ReculerHigh();
    EvitementObstacle = 0;
  }
```

```
else if ((T == 0)&&(EvitementObstacle == 0))
{
 TGauche();
 DriftDroite();
 delay(350);
 EvitementObstacle = 1;
}
else if ((T == 1)&&(EvitementObstacle == 1))
{
 if (Etat == 0x0F)
 {
  DriftDroite();
 }
 else
 {
  EvitementObstacle = 2;
 }
}
else if ((T == 0)&&(EvitementObstacle == 1))
{
 if (Etat == 0x0F)
 {
  DriftGauche();
 }
 else
 {
  EvitementObstacle = 2;
 }
}
else if (EvitementObstacle == 2)
{
 if ((Etat == 0x07)|(Etat == 0x03)|(Etat == 0x01)|(Etat == 0x04))
 {
  EvitementObstacle = 3;
 }
 else
 {
  AvancerHigh();
 }
}
else if (EvitementObstacle == 3)
{
 if ((Etat == 0x0D)|(Etat == 0x09))
 {
  SuiviDeLigne() ;
 }
 else
 {
  DriftDroite();
 }
```

```
  }
}

// Fait tourner le télémètre à droite
void TDroite()
{
  OCR1A = 1600;
  OCR1B = 0;
}

// Met le télémètre droit
void TBase()
{
  OCR1A = 2825;
  OCR1B = 0;
}

// Fait tourner le télémètre à gauche
void TGauche()
{
  OCR1A = 4200;
  OCR1B = 0;
}
```

Le robot suis la ligne, ralentis, rentre en contact avec l'obstacle il récupère la couleur recule jusqu'à ne plus détecter l'obstacle, met le télémètre à gauche, avance en tournant à droite avec un delay pour qu'il se rapproche de l'obstacle puis si le télémètre détecte l'obstacle le robot avance en tournant à droite grâce à la commande DriftDroite() et si il ne détecte plus l'obstacle, il fait un DriftGauche() qui lui permet d'avancer tout en tournant à gauche pour se rapprocher de l'obstacle.
Pour récupérer la ligne j'ai utilisé le capteur de ligne.

Lorsque le robot détecte la ligne tout en contournant l'obstacle le robot se met à avancer jusqu'à ce que le second capteur du capteur de ligne en partant de droite s'éteint, ci-dessous le rond noir représente le second capteur.

Lorsque le second capteur s'éteint le robot tourne à droite jusqu'à ce que le premier capteur s'éteigne, celui de droite sur le schéma ci-dessous, à ce moment-là le robot relance le suivi de ligne.

Ces deux schémas représentent les capteurs de face.

Conclusion

Pour conclure notre robot suis la ligne évite un obstacle en récupérant sa couleur, reprend la ligne pour suivre la ligne, lorsqu'il n'y a plus de ligne il suit un mur puis reprend la ligne lorsqu'il la retrouve jusqu'au moment ou le choix du chemin s'impose à lui ou il va choisir le chemin de la couleur qu'il à récupérer, re suivre la ligne sur le bon chemin et s'arrêter à une distance du mur final.

Le concours final entre classe était une bonne expérience avec un grand sentiment de compétition malheureusement nous avons fini 2ème.

Annexes

Robot chemin rouge sur paillasse :

https://youtu.be/Cg61SnHwq0s

Robot chemin vert sur paillasse :

https://youtu.be/bSsag6ugMUA

Robot chemin bleu sur paillasse :

https://youtu.be/GQl6KrALVp8

Finale :

https://youtube.com/clip/UgkxTq-_-nH0ZxBDSooeIrSDHtJx8WNG0Ejc

https://youtube.com/clip/Ugkx9LqtV5UjfZNMwdZu2_9SVgQBb533jt9j