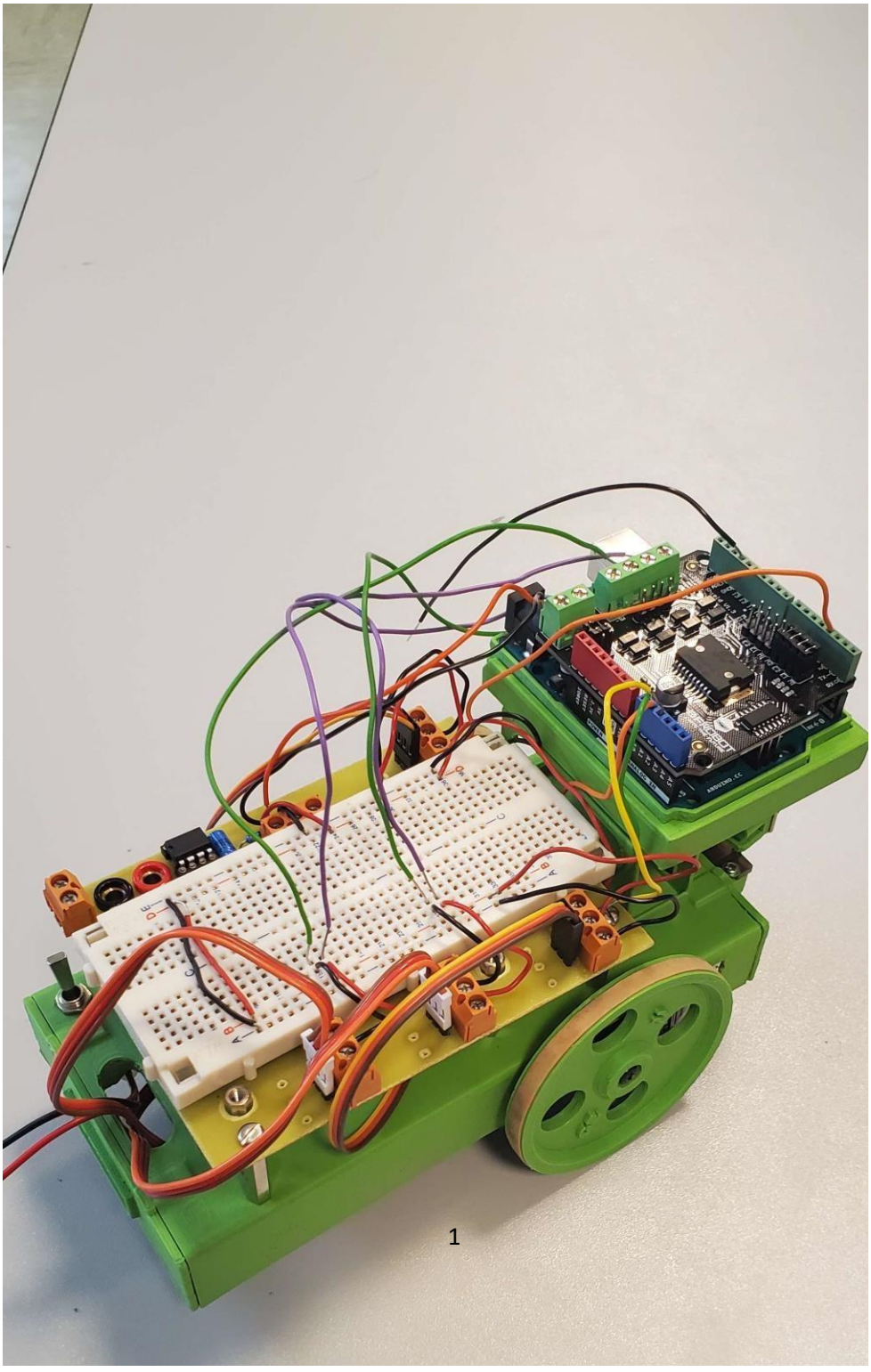
B10

Enzo MARTINS LOPES
Krishneshwar MANICOME

Compte rendu SAE

Robot Suiveur



Sommaire :

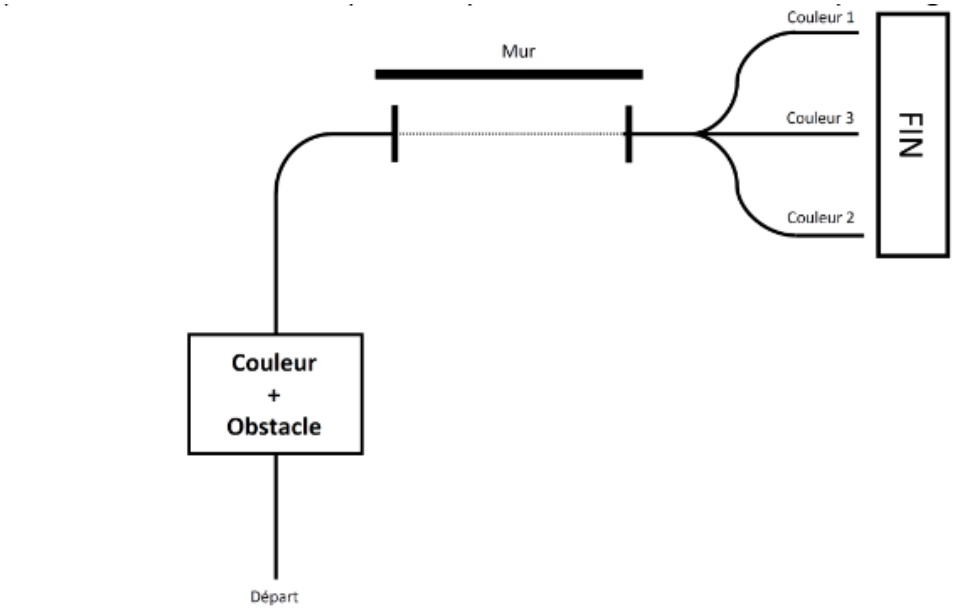
Introduction :

Le projet consiste à concevoir un système intelligent pour un robot mobile, afin qu'il puisse exécuter un scénario spécifique. Le robot sera équipé de capteurs numériques et/ou analogiques, certains étant déjà intégrés au robot, tandis que d'autres devront être ajoutés. Nous devrons donc récupérer les données analogiques ou numériques fournies par les capteurs. Ces informations seront ensuite traitées sur une carte Arduino Uno, afin d'accomplir les différentes tâches requises par le scénario.

Le robot possède les équipements suivants :

- **2 Moteurs PWM pilotant les roues**
- **1 Me RGB Line Follower**
- **1 Télémètre placé à l'avant du robot**
- **1 Servo-moteur permettant de diriger l'orientation du télémètre**
- **1 TCS 34725**
- **1 Shield Moteur**
- **1 Carte Arduino**

Scenario :



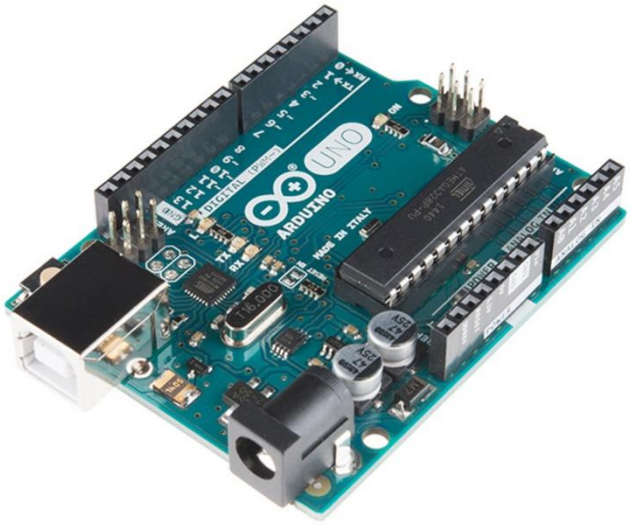
Le parcours est défini par les étapes suivantes :

1. Le robot suit une ligne depuis son point de départ.
2. Le robot détecte un obstacle.
    a. Le robot réduit sa vitesse à partir d'une distance de l'obstacle d'environ 15cm.
    b. Le robot avance doucement jusqu'à la détection de collision avec l'obstacle.
    c. Le robot récupère la couleur de l'obstacle.
    d. Le robot recule jusqu'à une distance convenable pour entamer l'évitement d'obstacle.
    e. Le robot évite l'obstacle et rejoint la ligne du parcours afin de reprendre le suivi de ligne.
3. Le robot suit la ligne.
4. A la détection d'une ligne perpendiculaire, le robot tourne le capteur d'obstacle à gauche pour longer un mur à une distance constante.
5. Le robot récupère le suivi de ligne lorsqu'une ligne perpendiculaire est détectée.
6. Le robot doit ensuite prendre le chemin qui correspond à la couleur de l'obstacle mesurer à l'étape 2.
7. En fonction de la couleur, le robot doit s'arrêter à une distance donnée par rapport à l'obstacle de fin de parcours.

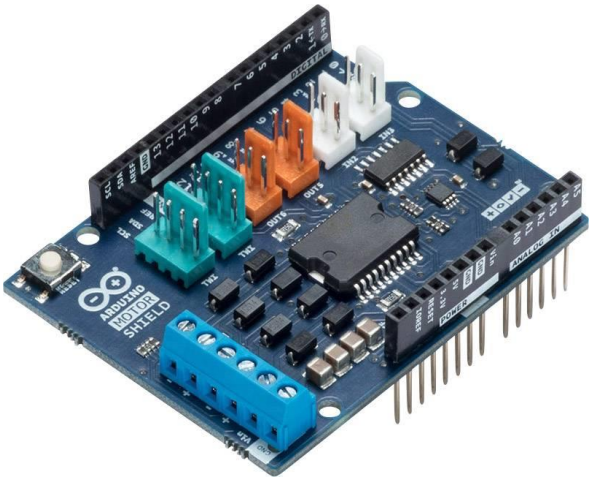
1) Fonction suiveur de ligne :

1-1 Câblage :

Voici notre carte Arduino doit acquérir, traiter et prendre des décisions et contrôler les actions du robot en fonction des informations fournies par les capteurs. Il sert de cerveau central pour orchestrer les différentes fonctionnalités du système intelligent du robot mobile.



Avant même de pouvoir câbler nous mettons notre Shield moteur pour pouvoir manipuler les moteurs avec.



Chaque moteur est équipé de trois câbles, chacun ayant une fonction spécifique :

1) **Le câble violet, qui est utilisé pour la commande.**
2) **Le câble rouge, qui est connecté à une source de 5V.**
3) **Le câble noir, qui est relié à la masse ou à 0V.**

Pour la commande, nous allons l'effectuer en utilisant les registres OCR0A et OCR0B. Ces registres sont associés aux broches D6 et D7,.

En ajustant les signaux envoyés à ces registres, nous pouvons contrôler la vitesse des roues selon nos besoins. Cela nous offre la possibilité de réguler la vitesse des moteurs en utilisant des commandes spécifiques.

En résumé, en reliant les câbles appropriés, en utilisant les registres OCR0A et OCR0B, et en envoyant les signaux de commande adéquats, nous sommes en mesure de régler la vitesse des roues à notre convenance.

1-2 Déplacement :

Avant de parler de mon suivi de ligne j'aimerai parler de comment fonctionne mes fonctions pour le déplacement de mon robot.

Prenons l'exemple de la fonction « avancer » :

```
void avancer()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=110;
  OCR0B=85;
  PORTD&=(1<<4);
  PORTD&=~(1<<7);
}
```

Ici nous pouvons voir au début de ma fonction « TCCR0A » et « TCCR0B ».

Le registre TCCR0A, utilisé dans le code, est configuré de la manière suivante :

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

Pour activer les fonctionnalités qui nous intéressent, les bits COM0A1 et COM0B1 sont mis à 1. Ces bits contrôlent les modes de comparaison pour les broches de sortie A et B du timer0.

Dans notre cas, nous souhaitons utiliser un mode PWM rapide, qui permet de varier le rapport cyclique du signal PWM de manière plus flexible. Pour cela, les deux bits WGM (Waveform Generation Mode) sont également mis à 1. Ces bits déterminent le mode de génération du signal PWM.

Ainsi, en configurant le registre TCCR0A avec la valeur 0xA3, nous activons le mode PWM rapide avec le mode de comparaison au sommet, ce qui nous permet de contrôler la vitesse des moteurs du robot de manière précise.

Le registre TCCR0B, utilisé dans le code, est configuré de la manière suivante :

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

Dans le code fourni, nous utilisons le registre TCCR0B pour effectuer une pré-division de la fréquence, afin d'obtenir la période souhaitée pour le bon fonctionnement des roues du robot. En configurant les bits CS01 et CS00 à 1, nous sélectionnons une pré-division de l'horloge système par un facteur de 64. Ainsi, le registre TCCR0B est défini avec la valeur 0x04.

OCR0A et OCR0B :

Les registres OCR0A et OCR0B sont utilisés pour contrôler les rapports cycliques des signaux PWM appliqués aux roues du robot. OCR0A est associé à la roue droite, tandis que OCR0B est associé à la roue gauche. En ajustant les valeurs de ces registres, nous pouvons varier la vitesse des roues du robot de manière indépendante. (OCR0A est la roue droite et OCR0A est la roue gauche si les deux roues on pas la même valeur c'est qu'il y a une roue plus rapide que l'autre et nous avons ajuster pour que les deux roues aillent a la même vitesse).

PORTD :

En ce qui concerne le sens des roues, nous utilisons les ports PD4 et PD7 de la carte Arduino. Chaque port est associé à une roue spécifique, tout comme les ports OC0A (PD6) pour la roue Droite et OC0B (PD5) pour la roue Gauche qui contrôlent la vitesse. En mettant les bits PD4 et PD7 à 0, nous configurons ces ports pour faire avancer le robot. Ainsi, nous les définissons à 0 dans le code pour indiquer la direction souhaitée. Nous avons décider de mettre dans le code PORTD&=~ (1<<7) car souvent quand la fonction rentrait dans un boucle la valeur du restait a 1 et ne repassais pas a 0 ceci empêchait notre code a passer à la suite.

Donc la fonction « avancer » aura pour but de faire rouler les deux roues à la même vitesse configurant les registres et les ports nécessaires.

Désormais nous allons montrer les différentes fonctions comme la fonction « reculer ».

```
void reculer()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  PORTD&=~(1<<4);
  PORTD|=(1<<7);
  OCR0A=100;
  OCR0B=100;

}
```

« PORTD&=~(1<<4) » ici est utilisé pour mettre le bit correspondant au port PD4 à 0. Cela signifie que la roue associée à ce port tournera dans un sens spécifier qui est en arrière. Car la roue est a l'envers par rapport a l'autre c'est pour cela que nous mettons a 1 pour une roue et l'autre 0.

"PORTD|=(1<<7);" est utilisé pour mettre le bit correspondant au port PD7 à 1.

Desormais nous irons voir la fonction « arrêt ».

```
void arret()
{
  TCCR0A=0;
}
```

Ici nous pouvons voir que nous avons uniquement mit a 0 la vitesse des roues du moteur.

Nous allons maintenant regarder les fonctions qui ont pour but de d'aller à droite ou à gauche.

```
void tourGauche()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=90;
  OCR0B=0;
  PORTD|=(1<<4);
  PORTD&=~(1<<7);
}
```

```
void tourDroite()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=0;
  OCR0B=100;s
  PORTD|=(1<<4);
  PORTD&=~(1<<7);
}
```

Ici nous pouvons voir que je demande uniquement a mes roue de tourner a droite ou a gauche en jouant avec OCR0A et OCR0B et d'avancer.

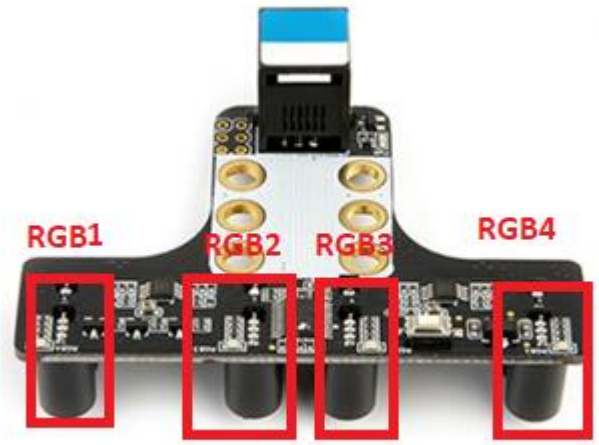
```
void tourGaucherapide()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=100;
  OCR0B=0;
  PORTD|=(1<<4);
  PORTD&=~(1<<7);
}
```

```
void tourDroiterapide()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=0;
  OCR0B=110;
  PORTD|=(1<<4);
  PORTD&=~(1<<7);
}
```

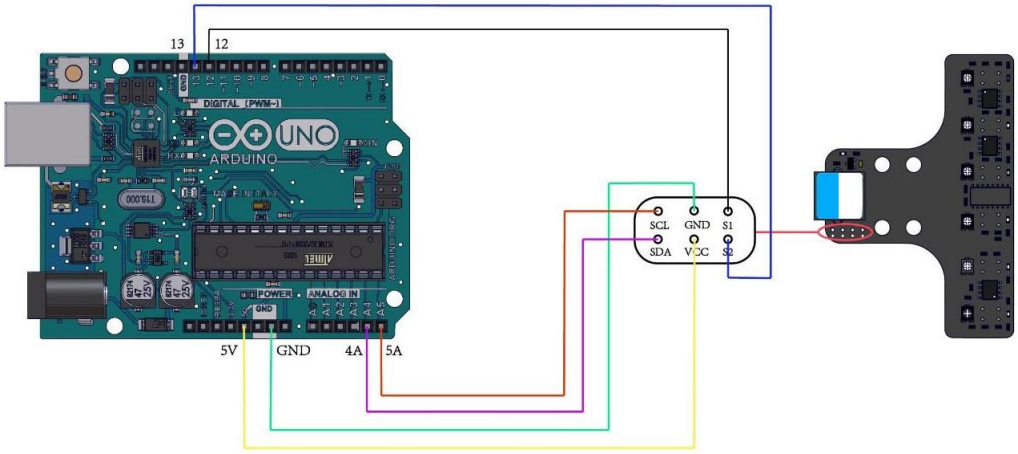
Ici nous faisons exactement la même chose mais j'augmente légèrement la vitesse des roues en n'augmentant la valeur de 10 pour chaque roue.

1-3 Suivi de ligne :

Pour pouvoir faire notre suiveur de ligne nous avons un capteur devant qui se nomme Me RGB LINE Follower.



Il y a 4 capteurs devant. Chaque capteur est considéré comme un bit donc la valeur peut être modifier entre 0 et 15



Pour le câblage c'est un capteur qui fonctionne en I2C comme présent avec SDA et SCL. Le capteur devra fonctionner sur une ligne noire avec la quelle nous devons calibrer en appuyant une fois sur le bouton.

Fonctionnement du capteur :

Capteur	Bit/Etat	Fonction
●○○○	13	Tour Gauche
●●●○	14	Tour Gauche
●●○○	12	Tour Gauche Rapide
●○○○	8	Tour Gauche Rapide
○●●●	7	Tour Droite
●○●●	11	Tour Droite
○○●●	3	Tour Droite Rapide
○○○●	1	Tour Droite Rapide
●○○●	9	Avancer
●●●●	15	Avancer
○○○○	0	Avancer

Désormais parlons de la partie informatique

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "MeRGBLineFollower.h"
#include "Adafruit_TCS34725.h"
```

Ici nous pouvons voir que j'utilise la bibliothèque appelée "Me RGB Line Follower" qui est le capteur de suivi de ligne. Cette bibliothèque facilite l'utilisation et la programmation du capteur dans le projet Arduino.

```
void state4sensor()
{
  state = RGBLineFollower.getPositionState();
}
```

La méthode "getPositionState()" retourne l'état actuel du capteur de suivi de ligne, qui est une valeur représentant les capteurs vu précédemment du robot par rapport à la ligne qu'il suit. La valeur est stockée est dans la variable « state » .

Voici notre code :

```
void suivi()
{
  state4sensor();

   if ((state == 13) || (state == 14))
  {
  tourGauche();
  }
  if ((state == 12) || (state == 8))
  {
   tourGaucherapide();
  }
   else if ((state == 7) || (state == 11))
  {
    tourDroite();
  }
  else if ((state == 3) || (state == 1))
  {

    tourDroiterapide();
  }
   else if ((state == 9) || (state == 15))
  {
    avancer();
  }
  else if (state == 0)
  {
  avancer();
    }

}
```

Ceci a donc pour but de suivre une ligne noire précisément et de tourner lorsqu'il le faut.

2) Contact avec l'obstacle :

Lorsque le capteur détecte un contact, le robot amorcera immédiatement une manœuvre de recul afin de commencer l'évitement de l'obstacle puis devra revenir sur la ligne noir en continuant le suivi de ligne.

Nous allons maintenant créer la fonction contact() qui nous permettra de détecter la couleur et de s'arrêter lorsque le contact est établi. Pour cela, nous allons brancher le commutateur et relier sa commande à la broche A0.

```
void contact_state()
{
  contact=analogRead(A0);

}
```

Par la suite il nous suffira de crée la variable « contact » et de la mettre a 0 et de même pour une variable qui se nomme « obstacle » et qui aura pour but d'empêcher la possibilité que le code reste dans une boucle donc lorsque obstacle change de valeur il pourra donc entrer dans une autre boucle comme ci-dessous.

```
int obstacle = 0, T=1, z=0, a=0;  int contact=0;
```

Ici nous pouvons voir que je mets la valeur de l'obstacle =0.

```
  while ((contact > 400)&&(obstacle == 0))
{

  contact_state();
  colorstate();
  arret();
  delay(200);
  reculer();
  delay(300);
  tourDroiterapide();
  delay(300);
  obstacle=1;
  }
```

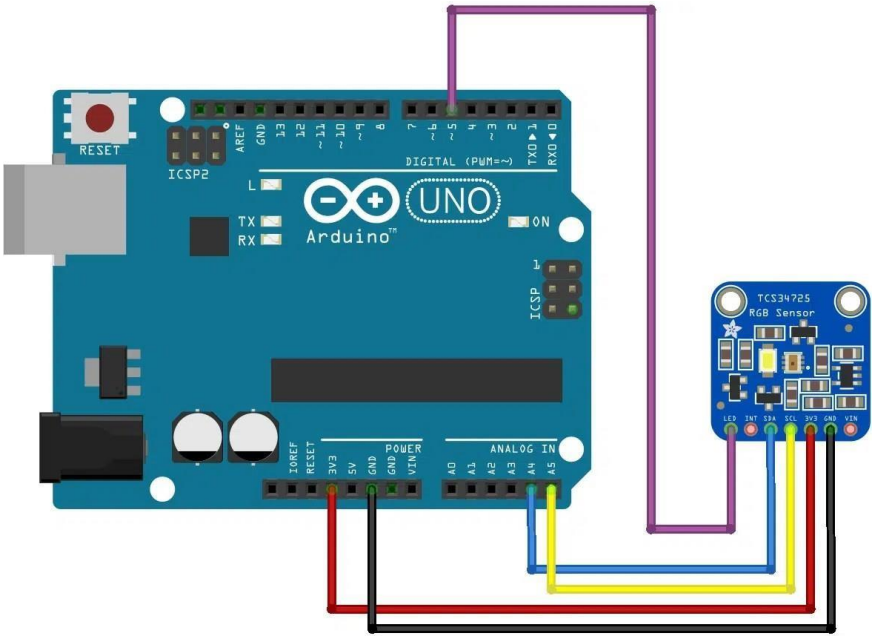
Ici si la valeur de contact est plus grande que 400 (en sachant que lorsqu'il touche l'obstacle la valeur monte à 1024) et qu'obstacle à la même valeur que 0 alors il pourra commencer l'esquivement de l'obstacle.

Donc il commence par un arrêt qui a une durée de 200 millisecondes on le demande ensuite de faire une action qui a pour but de reculer qui a une durée de 300 millisecondes puis on lui demande vite tourner à droite pour se retrouver à la droite de l'obstacle puis nous mettons la valeur de l'obstacle a 1.

3) Fonction de la détection de couleur :

Pour détecter et différencier les trois couleurs potentielles présentes sur l'obstacle, nous utiliserons le capteur de couleur TCS34725. Ce capteur nous permettra de reconnaître précisément les couleurs et de prendre des décisions en fonction de celles-ci. Avant d'atteindre cette étape, il est crucial que le robot soit capable d'identifier et de distinguer les trois couleurs spécifiques qu'il pourrait détecter lors du contact avec l'obstacle.

Voici le câblage de notre capteur TCS34725 :



Après avoir réalisé le câblage requis, nous allons explorer les exemples de la bibliothèque Arduino pour le capteur de couleur. En incluant la bibliothèque appropriée, nous pourrons accéder au programme de test de couleur fourni. En utilisant ce programme, nous avons pu relever les valeurs RGB associées à chaque couleur. Par la suite nous avons personnellement mesurer chaque couleur.

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

  if (tcs.begin()) {

  } else {
    Serial.println("No TCS34725 found ... check your connections");
    while (1); // halt!
  }

  // use these three pins to drive an LED
#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;
    }
    //Serial.println(gammatable[i]);
  }


  RGBLineFollower.begin();
  RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);
  Serial.begin(9600);
  DDRD=0xff;
  DDRB=0xff;
  PORTD=PORTD|(1<<0);
  PORTD=PORTD|(1<<1);
  TCCR1A=0xA2;
  TCCR1B=0x1B;
  OCR1A=375;
  ICR1=5000;

  OCR0A=100;
  OCR0B=100;

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

 Serial.begin(9600);
}
```

```
  while ((contact > 400)&&(obstacle == 0))
{

  contact_state();
  colorstate();
  arret();
  delay(200);
  reculer();
  delay(300);
  tourDroiterapide();
  delay(300);
  obstacle=1;
  }
```

Ici nous pouvons voir comme dans le code précèdent que nous avons mis une fonction « colorstate » celle-ci a pour but de prendre la couleur et de lui donner une valeur pour pouvoir faire le dernière obstacle et choisir la bonne direction en fonction de la bonne couleur.

Voici mon code :

```
void colorstate()
{
  arret();
  delay(100);
if (a==0)
{
  tcs.getRawData(&r, &g, &b, &c); // Relevé des couleurs
  a=1;
}
  if ((r>g)&& (r>b) && (r>200))      // Si on détecte + de rouge
  {
    Serial.println("C'est rouge");
    couleur = 1;

  }

   if ((g>r)&& (g>b) && (g>100)) // Si on détecte + de vert
  {
    Serial.println("C'est vert");
    couleur = 2;

  }

   if ((b>r)&& (b>g) && (b>100)) // Si on détecte + de bleu
  {
    Serial.println("C'est bleu");
    couleur = 3;

  }


}
```

Nous pouvons voir que nous utilisons la variable tcs.getRawData qui a pour but de garder constamment la couleur qui a été vu lors du contact lors de tout le circuit.

Par la suite nous faisons des boucles avec des « if » qui ont pour but de devoir dire si la couleur rouge bleu ou vert est plus grande qu'une autre alors celle-ci doit afficher dans le moniteur série la couleur et doit mettre la variable « couleur » qui a une valeur qui est entre 1 à 3.

4) Evitement d'obstacle :

L'évitement d'obstacle s'activera lorsque notre robot ne verra pas la ligne noire (donc après avoir fait un reculement et un tour droite) et que la variable 'obstacle' sera incrémentée à 1 après le contact.

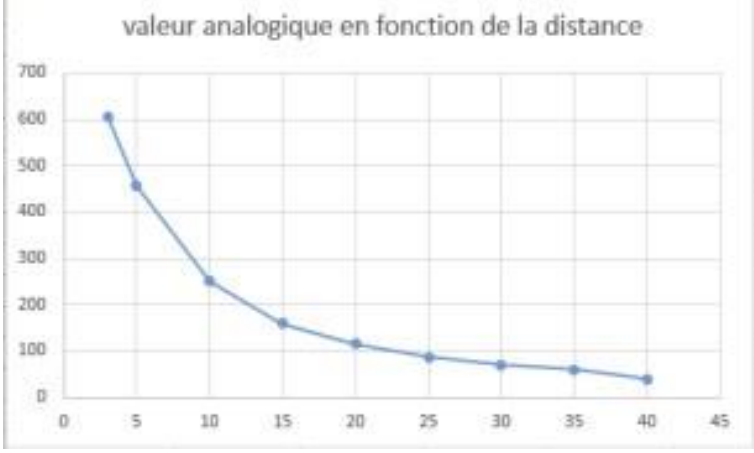
```
if((state == 15) && (obstacle == 1))
  {
    dodge();
    T=2;
  }
```

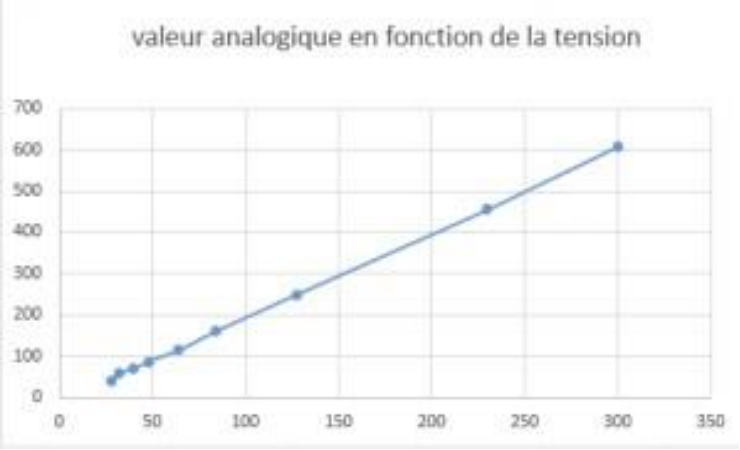
Donc ici si notre capteur voit toute la ligne noir (état a 15) et obstacle est a 1 alors il fait l'évitement. Nous allons ensuite parler de notre télémètre pour pouvoir l'utiliser lors de notre esquivement de l'obstacle

Caractéristiques du télémètre :

Après avoir câblé le télémètre et effectué des mesures on observe que la valeur analogique évolue de manière non-linéaire par rapportà la distance (en cm dans le tableau), plus la distance est grande plusla variation entre deux paliers est faible rendant la mesure moins précise (à l'infini on obtient 2 en Analog et 4mV de tension).
La courbe en fonction de la tension est linéaire avec un rapport proche de 2mV pour 1 unité Analogique.

Distance	3	5	10	15	20	25	30	35	40
mV	300	230	128	84	64	48	40	32	28
Analog	607	456	250	160	115	87	70	60	40





Pour notre fonction évitement :

```
void dodge()
{

  Tel();
  telemetre_gauche();

  while((Telemetre < 170))
  {
    tourGauche();
    Tel();
  }
    tourDroite();
    Tel();


}
```

Pour le contournement on met notre télémètre a gauche pour pouvoir voir l'obstacle avec puis nous avons mesurer avant la valeur pour esquiver parfaitement l'obstacle qui est donc 170, si notre télémètre est proche de l'obstacle alors il ferra un tour à droite donc évitera l'obstacle et s'il est trop loin alors se rapprochera de notre obstacle.

5) Reprise du suiveur de ligne :

Ici la difficulté était de devoir faire en sorte que notre robot puisse reprendre la ligne après notre évitement d'obstacle pour ceci nous avons eu l'idée de pouvoir faire en sorte que lorsque notre robot détecte la ligne peut importer l'état alors celui-ci doit faire un tour droite pour se retrouver face à la ligne et continuer celle-ci.

```
  else if((state != 15) && (obstacle == 1)) {
    Telemetre=0;
    telemetre_normal();
    if(state == 0)
    {
      tourDroiterapide();
      delay(220);
      suivi();
      obstacle=0;
    }
    suivi();

    //obstacle=0;
 }
```

Ici nous pouvons voir que si mon état voit du noir je force mon servo moteur du télémètre à être au milieu pour le mettre en face, puis je demande à mon robot de faire un tour a droite avec un temp indiqué pour qu'il puisse être correctement face a la ligne et pour finir faire mon suivi. Je décide de mettre obstacle a 0 pour qu'il ne rentre plus dans aucune boucle.

6) Détection d'une ligne perpendiculaire :

Pour le mur nous faisons le même principe que pour notre obstacle lorsque obstacle est a 0 et que la totalité de mes capteurs ne pas voient la lignes alors il décide de faire un esquivement d'obstacle avec la même fonction.

```
if((state == 15) && (obstacle == 0))
  {
    //d=1;
    dodge();
    T=3;
  }
```

Ici nous faisons le même principe mais lorsque l'« obstacle » égale à 0 pour qu'il ne puisse pas retourner dans l'ancienne boucle et continuer le circuit. Mais nous mettons une variable T=3 pour la suite.

Nous utilisons la même fonction d'évitement de l'obstacle pour notre mur :

```
void dodge()
{

  Tel();
  telemetre_gauche();

  while((Telemetre < 170))
  {
    tourGauche();
    Tel();
  }
    tourDroite();
    Tel();


}
```

Reprise de la ligne:

Pour notre reprise de la ligne, nous maintenons le même concept que celui utilisé dans notre code initial. Lorsque notre capteur détecte une ligne noire, le robot entre dans l'état de suivi de ligne. Cependant, pour cela, notre variable "T" doit être égale à 3 ; sinon, il ne passe pas dans cet état. Cette variable permet d'apporter une logique hiérarchique à notre code, garantissant que le robot puisse parcourir l'ensemble des obstacles de son circuit sans sauter d'étapes ni rester bloqué dans une boucle.

```
  else if((state != 15)&&(T==3)) {
    Telemetre=0;
    telemetre_normal();
    suivi();
    T=5;
}
```

Désormais nous mettons T=5.

7) Choix du chemin en fonction de la couleur :

Pour la sélection du chemin final, notre objectif était de choisir un chemin soit à gauche, soit au milieu, soit à droite, en fonction de la couleur initiale détectée.

Lorsque le robot a un contact, il analyse la couleur de la ligne qu'il rencontrera a la fin. Si la couleur est interprétée comme indiquant un chemin à gauche, le robot est programmé pour suivre la ligne correspondante à gauche. Si la couleur indique un chemin au milieu, il suit la ligne du milieu. Et si la couleur suggère un chemin à droite, il suit la ligne correspondante à droite.

Voici notre code :

```
while (T == 5) {
 RGBLineFollower.loop();
 state = RGBLineFollower.getPositionState();
  if ((state==0)&&(a == 3)) //kafka
  {
    Telemetre=0;
    telemetre_normal();
    Serial.println(couleur);
    if (couleur == 1)
    {
      RGBLineFollower.setRGBColour(RGB_COLOUR_RED);
      Serial.println("rouge");
      avancer();
      delay(170);
      suivi();
      couleur=5;
    }
    if (couleur == 2) // ou if
    {
      RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);
      Serial.println("vert");
      avancer();
      delay(50);
      tourGauche();
      delay(170);
      suivi();
      couleur=5;
    }
    if (couleur == 3)  //ou if
    {
      RGBLineFollower.setRGBColour(RGB_COLOUR_BLUE);
      avancer();
      delay(50);
      tourDroite();
      delay(120);
      suivi();
      couleur=5;
      Serial.println(couleur);
    }

  }
```

Désormais que T=5 lorsque nos 4 capteurs voient uniquement du noir et que notre variable a = 3.

Alors pour chaque valeur de notre variable « couleur » il prend un chemin précis. En effet nous pouvons voir tout d'abord que lorsqu'il est censé avoir vu une couleur il l'affiche dans le moniteur série et modifie la couleur du Me RGB LineFollower en fonction de la couleur qui a été vu. Ensuite pour chaque couleur il y a une action avec un DeLay comme par exemple « tourdroite » avec un DeLay de 170 milliseconde pour prendre le chemin de droite.

Puis pour chaque chemin nous mettons la variable « couleur » a 5 pour pouvoir sortir de la boucle.

Désormais nous allons rentrer en détaille sur comment la variable couleur prendre c'est 3 valeurs

Choix de la couleur :

```
void colorstate()
{
  arret();
  delay(100);
if (a==0)
{
  tcs.getRawData(&r, &g, &b, &c); // Relevé des couleurs
  a=1;
}
  if ((r>g)&& (r>b) && (r>200))      // Si on détecte + de rouge
  {
    Serial.println("C'est rouge");
    couleur = 1;

  }

   if ((g>r)&& (g>b) && (g>100)) // Si on détecte + de vert
  {
    Serial.println("C'est vert");
    couleur = 2;

  }

   if ((b>r)&& (b>g) && (b>100)) // Si on détecte + de bleu
  {
    Serial.println("C'est bleu");
    couleur = 3;

  }


}
```

Comme dit précédemment ce code choisi une couleur et la garde tout le long du circuit mais pour chaque couleur qui est lu la valeur est prise. Par exemple si notre capteur voit du bleu alors la valeur de la variable de couleur sera de 3. Et pourra donc la stocker tout au long du circuit et exercer l'action a la toute fin donc ici qui est « tourDroite ».

8) Arrêt :

Pour la fin de notre circuit on nous demande de faire arrêter notre robot juste avant l'obstacle pour ne pas abimer le switch. Pour cela nous avons uniquement fait ce code :

```
Tel();
a =2;
if((Telemetre>500)&&(a==2))
{
  Tel();
  Serial.println(Telemetre);
  arret();
  }
  else suivi();
 }
}
```

Ici nous pouvons voir que nous forçons notre valeur a=2 pour rentrer dans la boucle et quand notre télémètre ce trouve très proche d'un obstacle en sachant que notre télémètre est au milieu alors ils s'arrêtent quand il voit un obstacle très proche sinon il continue a faire sont suivi de ligne.

9) Conclusion :

Dans le cadre de ce projet de robot suiveur de ligne, j'ai acquis une meilleure compréhension du fonctionnement de l'I2C. Grâce à cette connaissance que nous avons pu approfondie, nous avons pu mettre en pratique nos compétences en programmation Arduino. Et continuer de travailler dans ce robot suiveur de ligne.

L'un des aspects les plus intéressants de ce projet a été la participation à un concours, où nous avons eu l'opportunité de mettre en compétition notre robot suiveur de ligne contre les mêmes robots avec un code diffèrent. Grâce à nos efforts nous avons optimiser les performances de notre robot et nous avons réussi à avoir la première place pour ce concours.

Au-delà des résultats du concours, ce projet nous a permis d'améliorer nos compétences en matière de résolution et de problèmes, de logique et de pensée créative.

Ce projet de robot suiveur de ligne nous a permis de mettre en théorie et en pratique nos connaissances en Arduino, d'explorer les différentes capacités de l'I2C. Je suis fier des résultats que nous avons obtenus et de notre engagement à optimiser notre robot.

Annexe :

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include "MeRGBLineFollower.h"
#include "Adafruit_TCS34725.h"
int Telemetre;
int obstacle = 0, T=1, z=0, a=0;
//int mure,d=0;
int AZ=0;

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

#define commonAnode true

// our RGB -> eye-recognized gamma color
byte gammatable[256];


Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);

MeRGBLineFollower RGBLineFollower(13,12, ADDRESS2);
int contact=0;
int16_t turnoffset = 0;
int16_t state = 0;
uint16_t r, g, b, c, colorTemp, couleur=0; // Initialisation variables couleur
uint8_t sensorstate;
int color = 0;

void state4sensor()
{
  state = RGBLineFollower.getPositionState();
}

void avancer()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=110;
  OCR0B=85;
  PORTD&=(1<<4);
  PORTD&=~(1<<7);
}

void avancer_etourdissement()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=100;
  OCR0B=100;
  PORTD|=(1<<4);
  PORTD&=~(1<<7);
}

void reculer()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  PORTD&=~(1<<4);
  PORTD|=(1<<7);
  OCR0A=100;
  OCR0B=100;

}

void tourGauche()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=90;
  OCR0B=0;
  PORTD|=(1<<4);
  PORTD&=~(1<<7);
}

void tourDroite()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=0;
  OCR0B=100;
  PORTD|=(1<<4);
  PORTD&=~(1<<7);
}
```

```
void tourGaucherapide()
{
  TCCR0A=0xA3;
  TCCR0B=0x04;
  OCR0A=100;
  OCR0B=0;
  PORTD|=(1<<4);
  PORTD&=~(1<<7);
}

void arret()
{
  TCCR0A=0;
}

void suivi()
{
  state4sensor();

   if ((state == 13) || (state == 14))
  {
  tourGauche();
  }
  if ((state == 12) || (state == 8))
  {
   tourGaucherapide();
  }
   else if ((state == 7) || (state == 11))
  {
    tourDroite();
  }
  else if ((state == 3) || (state == 1))
  {

    tourDroiterapide();
  }
   else if ((state == 9) || (state == 15))
  {
    avancer();
  }
  else if (state == 0)
  {
  avancer();
    }

}

void contact_state()
{
  contact=analogRead(A0);

}
void telemetre_gauche()
{
  OCR1A = 550;
}
void telemetre_normal()
{
  OCR1A = 375;
}
void telemetre_droite()
{
  OCR1A = 190;
}
void Tel()
{
  Telemetre=analogRead(A3);
  //Serial.println(Telemetre);
}



void dodge()
{

  Tel();
  telemetre_gauche();

  while((Telemetre < 170))
  {
    tourGauche();
    Tel();
  }
    tourDroite();
    Tel();


}
```

```
void douge()
{

  Tel();
  telemetre_gauche();

  while((Telemetre < 170))
  {
    tourGauche();
    Tel();
  }
    tourDroite();
    Tel();


}


void colorstate()
{
  arret();
  delay(100);
if (a==0)
{
  tcs.getRawData(&r, &g, &b, &c); // Relevé des couleurs
  a=1;
}
  if ((r>g)&& (r>b) && (r>200))      // Si on détecte + de rouge
  {
    Serial.println("C'est rouge");
    couleur = 1;

  }

   if ((g>r)&& (g>b) && (g>100)) // Si on détecte + de vert
  {
    Serial.println("C'est vert");
    couleur = 2;

  }

   if ((b>r)&& (b>g) && (b>100)) // Si on détecte + de bleu
  {
    Serial.println("C'est bleu");
    couleur = 3;

  }


}


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

  if (tcs.begin()) {

  } else {
    Serial.println("No TCS34725 found ... check your connections");
    while (1); // halt!
  }

  // use these three pins to drive an LED
#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;
    }
    //Serial.println(gammatable[i]);
  }
```

```
  RGBLineFollower.begin();
  RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);
  Serial.begin(9600);
  DDRD=0xff;
  DDRB=0xff;
  PORTD=PORTD|(1<<0);
  PORTD=PORTD|(1<<1);
  TCCR1A=0xA2;
  TCCR1B=0x1B;
  OCR1A=375;
  ICR1=5000;

  OCR0A=100;
  OCR0B=100;

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

 Serial.begin(9600);
}

void loop()
{

  RGBLineFollower.loop();
  suivi();
   contact_state();
   state4sensor();
  if (state !=15)
  {
    telemetre_normal();
    }

    while ((contact > 400)&&(obstacle == 0))
  {

    contact_state();
    colorstate();
    arret();
    delay(200);
    reculer();
    delay(300);
    tourDroiterapide();
    delay(300);
    obstacle=1;
    }

  RGBLineFollower.loop();
  state = RGBLineFollower.getPositionState();


  if((state == 15) && (obstacle == 1))
    {
      dodge();
      T=2;
    }

  else if((state != 15) && (obstacle == 1)) {
    Telemetre=0;
    telemetre_normal();
    if(state == 0)
    {
      tourDroiterapide();
      delay(220);
      suivi();
      obstacle=0;
    }
    suivi();

    //obstacle=0;
 }

  RGBLineFollower.loop();
  state = RGBLineFollower.getPositionState();


  if((state == 15) && (obstacle == 0))
    {
      //d=1;
      dodge();
      T=3;
    }

  else if((state != 15)&&(T==3)) {
    Telemetre=0;
    telemetre_normal();
    suivi();
    T=5;
 }
   RGBLineFollower.loop();
  state = RGBLineFollower.getPositionState();
```

```
 if((state == 15) && (obstacle == 1))
   {
     dodge();
     T=2;
   }

 else if((state != 15) && (obstacle == 1)) {
   Telemetre=0;
   telemetre_normal();
   if(state == 0)
   {
     tourDroiterapide();
     delay(220);
     suivi();
     obstacle=0;
   }
   suivi();

   //obstacle=0;
}

 RGBLineFollower.loop();
 state = RGBLineFollower.getPositionState();


 if((state == 15) && (obstacle == 0))
   {
     //d=1;
     dodge();
     T=3;
   }

 else if((state != 15)&&(T==3)) {
   Telemetre=0;
   telemetre_normal();
   suivi();
   T=5;
}
  RGBLineFollower.loop();
 state = RGBLineFollower.getPositionState();

 while (T == 5) {
  RGBLineFollower.loop();
  state = RGBLineFollower.getPositionState();
   if ((state==0)&&(a == 3)) //kafka
   {
     Telemetre=0;
     telemetre_normal();
     Serial.println(couleur);
     if (couleur == 1)
     {
       RGBLineFollower.setRGBColour(RGB_COLOUR_RED);
       Serial.println("rouge");
       avancer();
       delay(170);
       suivi();
       couleur=5;
     }
     if (couleur == 2) // ou if
     {
       RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN);
       Serial.println("vert");
       avancer();
       delay(50);
       tourGauche();
       delay(170);
       suivi();
       couleur=5;
     }
     if (couleur == 3)  //ou if
     {
       RGBLineFollower.setRGBColour(RGB_COLOUR_BLUE);
       avancer();
       delay(50);
       tourDroite();
       delay(120);
       suivi();
       couleur=5;
       Serial.println(couleur);
     }

   }

 Tel();
 a =2;
 if((Telemetre>500)&&(a==2))
 {
   Tel();
   Serial.println(Telemetre);
   arret();
   }
   else suivi();
 }
}
```