

SAÉ 4

Sommaire

I. Introduction	3
II. Schéma fonctionnel général et calendrier initial	4
III. Description des fonctions de bas niveau	5
III.1. Commande des roues	5
III.2. Capteurs de ligne	7
III.3. Orientation du télémètre	9
IV. Description des fonctions de haut niveau	10
IV.1. Mouvement du robot	10
IV.2. Suivi de ligne	10
IV.3. Détecteur de couleur	12
IV.4. Détection d'obstacle	14
IV.5. Suivi du Mur	20
V. Mise en œuvre du scénario	23
VI. Bilan des références aux AC	24
VII. Conclusion	25
VIII. Annexes	26
VIII.1. Tableau de référence aux compétences	26
VIII.2. Codes	27
<i>VIII.2.1. Code complet</i>	<i>34</i>

I. Introduction

L'objectif de ce projet est système intelligent permettant à un robot mobile de réaliser un scénario donné en reprenant le robot du S3 avec des nouveaux capteurs comme le capteur de suivi de ligne, le capteur de couleur et un bouton poussoir.

Le scénario que le robot doit réaliser est une succession de tâches depuis un point de départ jusqu'au point d'arrivée. Le chemin à parcourir par le robot mobile est donné par la figure ci-dessous.

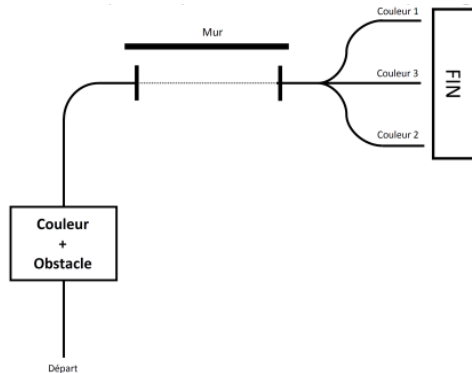


Figure 1. Chemin à réaliser pour le robot

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.

Le Tableau 10 (section VI), récapitule l'utilisation des apprentissages critiques qui ont permis le développement de ce travail.

La section II présente un schéma fonctionnel complet du projet. Il y sera fait référence dans la section III et IV lors de la description de chaque fonction de bas niveau et haut niveau.

II. Schéma fonctionnel général et calendrier initial

Schéma fonctionnel (AC11.01)

Fonction de bas niveau

Fonction de haut niveau

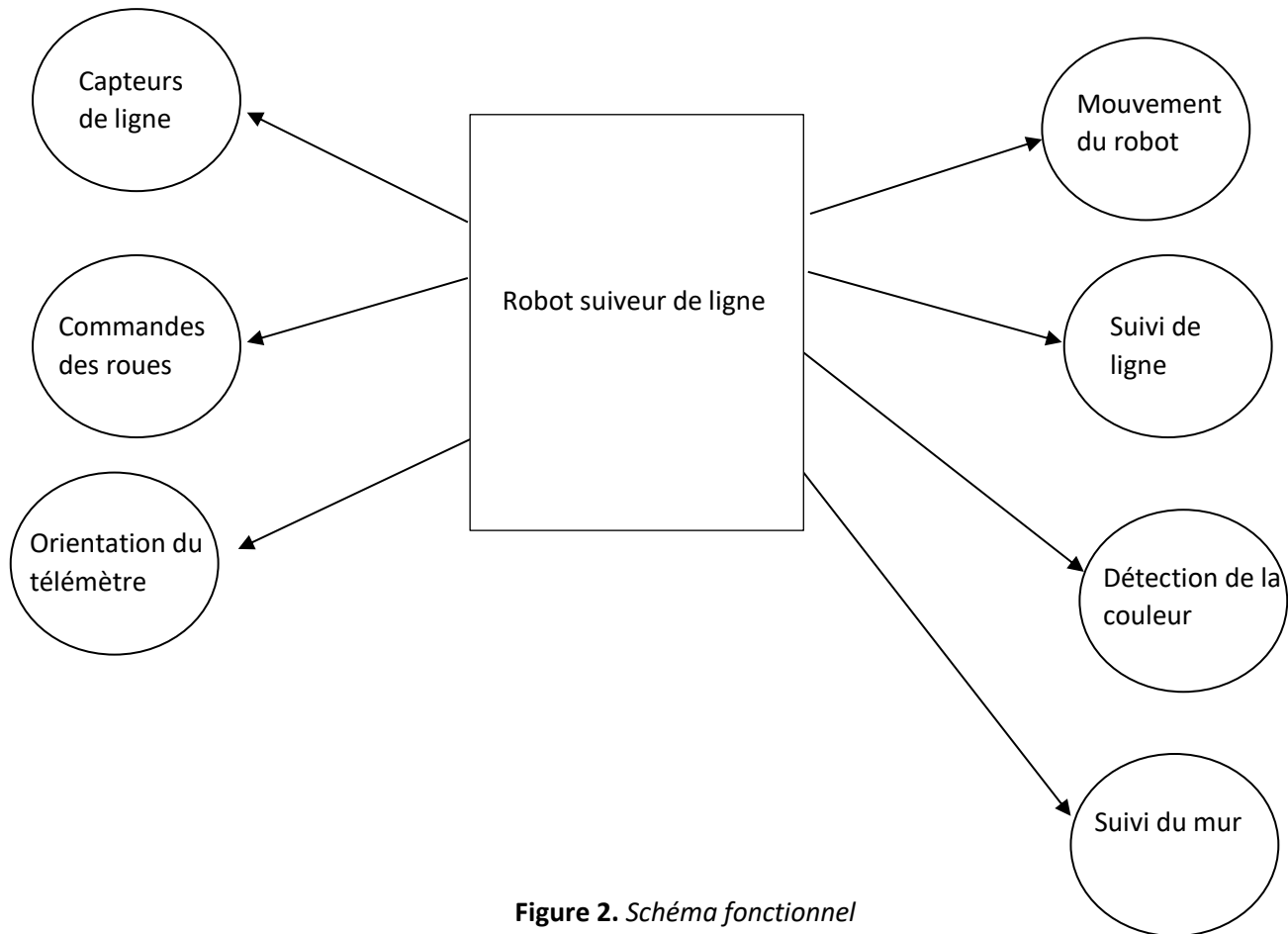


Figure 2. Schéma fonctionnel

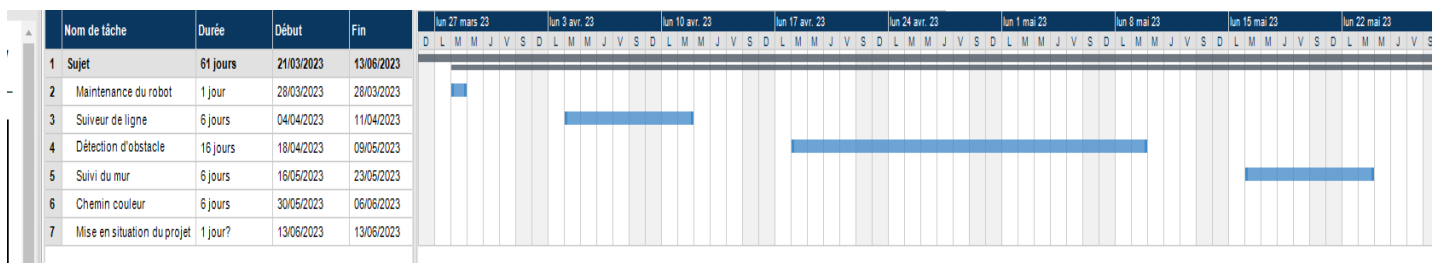


Figure 3. Calendrier initial

III. Description des fonctions de bas niveau

Dans le premier paragraphe (III.1), le mouvement de chaque roue est étudié séparément. Dans le deuxième paragraphe (III.2), on étudie comment fonctionne le capteur de ligne. Puis dans le troisième paragraphe (III.3), on étudie l'orientation du télémètre.

III.1. Commande des roues

AC23.01

Sur le robot suiveur de ligne, nous avons deux moteurs qui commandent chacun une des deux roues. L'objectif de cette partie est de faire avancer les roues du robot sur des vitesses différentes.

Pour la commande des roues on reprend la version du robot suiveur de ligne du S3. Puis sans changer le programme on effectue plusieurs essais afin de savoir si le programme fonctionne toujours.

OCR1A correspond au moteur de la roue droite et OCR1B correspond au moteur de la roue gauche.

On reprend le programme du Timer 1 (**Annexe 1**. Programme Timer1 commande des roues) j'ai repris les mesures avec l'oscilloscope pour voir si ça correspond aux valeurs mesurées dans la version du robot suiveur de ligne du S3. On observe que quand les valeurs de OCR1A et OCR1B se rapprochent de 40000 le signal devient continu en 5V et plus la valeur se rapproche de 0 le signal devient nul.

AC22.01

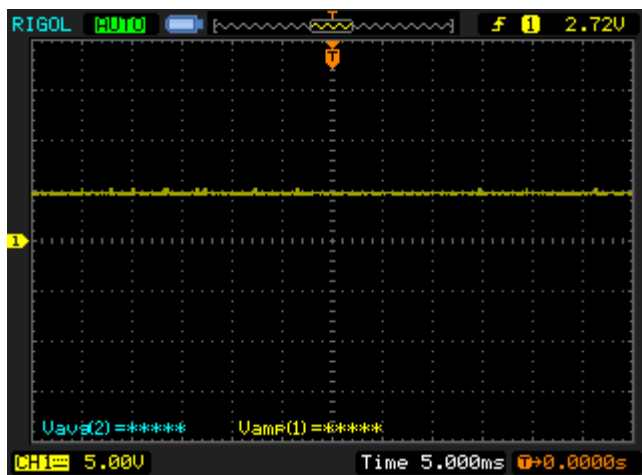


Figure 4. Valeur de OCR1A et OCR1B proche de 40000

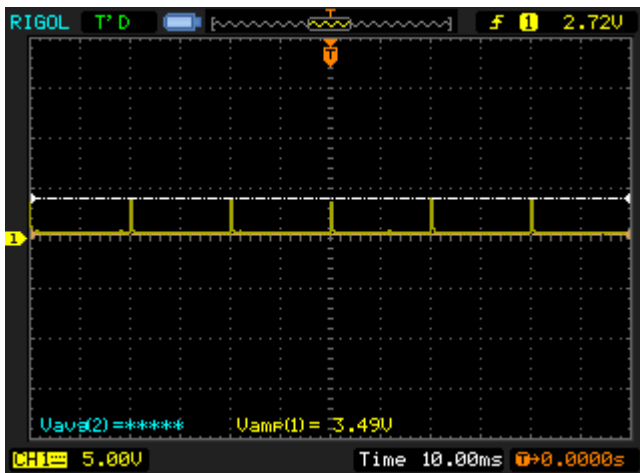


Figure 5. Valeur de OCR1A et OCR1B proche de 0 :

On observe que les roues du robot tournent. Plus la valeur de OCR1A et OCR1B s'approche de 40000 plus la vitesse augmente jusqu'à sa vitesse max et plus la valeur s'approche de 0 plus la vitesse des roues diminue. Ce qui correspond bien au bilan réalisé dans la version du robot suiveur de ligne. On remarque que les roues ne tournent pas du même sens pour les faire tourner dans le même sens, il suffit d'inverser la polarité d'un des moteurs.

Afin que le robot ne va pas trop vite ni trop lentement lors des futurs essais on fixe la valeur de OCR1A et OCR1B à 15000.

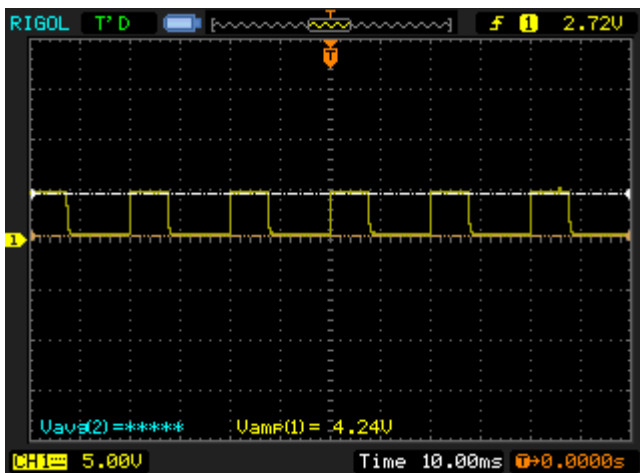


Figure 6. Valeur de OCR1A et OCR1B a la valeur de 15000

III.2. Capteurs de ligne

Dans cette partie on s'intéresse au capteur du suivi de ligne qui a été remplacé par rapport au Robot du S3. Ce capteur ME RGB line follower fonctionne par communication I2C et comprend 4 DEL RVB et 4 récepteurs photosensibles. Il est conçu pour suivre des lignes sur des pistes claires avec un fond sombre ou des pistes sombres sur un fond clair.

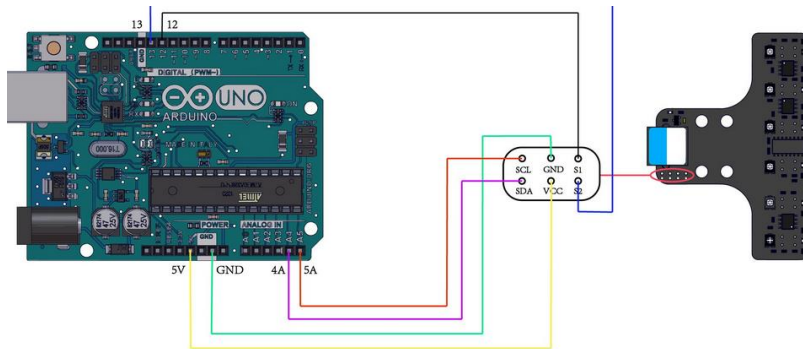


Figure 7. Câblage du capteur suivi de ligne

AC24.01 AII AC24.02 AII

Quand le capteur est câblé les 4 DEL RVB s'allument en bleu. On peut changer la couleur des LED en restant appuyé quelques secondes sur le bouton blanc on peut le faire changer entre bleu vert et rouge. Ce bouton sert aussi d'apprentissage si on appuie une fois les LED vont clignoter et enregistrent la couleur qu'il voit comme couleur de fond (Etat 1). Pour apprendre la couleur de la ligne on positionne le capteur au-dessus de la ligne puis on clique deux fois les LED vont clignoter et enregistrent la couleur qu'il voit comme couleur de ligne (Etat 0). Quand le capteur ne voit pas la ligne les LED s'allument et quand il voit la ligne les LED sont éteintes. Pour commander le capteur suivi de ligne on va prendre le programme MeRGBLineFollower qui nous a été fourni.

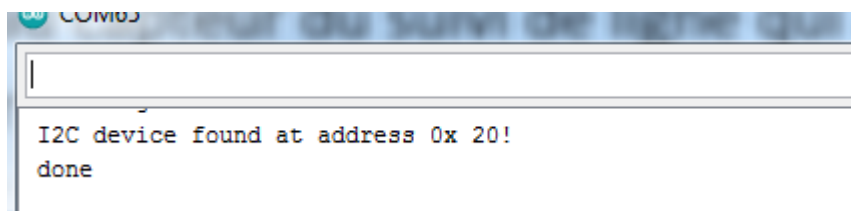


Figure 8. Adresse I2C capteur de suiveur de ligne

Entrées	Sorties	
	Attendues, théoriques	Réelles, mesurées
Capteur sur la ligne noir	Valeur entre 0 et 255	Valeur >100
Capteur sur la ligne blanc	Valeur entre 0 et 255	Valeur <100
Identification des capteur	4 capteurs attendue	i2cData[1,2,3,4]

Tableau 1. Tableau d'identification des 4 capteurs.

AC22.01

Capteur 1 sur le noir :

```
i2cData[0]:250 i2cData[1]:31 i2cData[2]:220 i2cData[3]:163 i2cData[4]:203 i2cData[5]:79 i2cData[6]:254 i2cData[7]:14
```

Capteur 2 sur le noir :

```
i2cData[0]:250 i2cData[1]:126 i2cData[2]:47 i2cData[3]:157 i2cData[4]:202 i2cData[5]:58 i2cData[6]:255 i2cData[7]:13
```

Capteur 3 sur le noir :

```
i2cData[0]:250 i2cData[1]:127 i2cData[2]:195 i2cData[3]:35 i2cData[4]:195 i2cData[5]:173 i2cData[6]:0 i2cData[7]:11
```

Capteur 4 sur le noir :

```
i2cData[0]:250 i2cData[1]:134 i2cData[2]:215 i2cData[3]:166 i2cData[4]:47 i2cData[5]:171 i2cData[6]:1 i2cData[7]:7
```

On remarque le i2cData [0] ne change pas et les capteur i2cData [5,6,7] affiche des valeurs qui ne correspondent à aucune valeur qu'on est censé avoir.

Ses variables correspondent à d'autres valeurs :

```
//RGBLineFollower IIC Register Address
#define RGBLINEFOLLOWER_DEVICE_ID_ADDR      (0x00)
#define RGBLINEFOLLOWER_RGB1_ADDR           (0x01)
#define RGBLINEFOLLOWER_RGB2_ADDR           (0x02)
#define RGBLINEFOLLOWER_RGB3_ADDR           (0x03)
#define RGBLINEFOLLOWER_RGB4_ADDR           (0x04)
#define RGBLINEFOLLOWER_TURNOFFSET_L_ADDR    (0x05)
#define RGBLINEFOLLOWER_TURNOFFSET_H_ADDR    (0x06)
#define RGBLINEFOLLOWER_STATE_ADDR          (0x07)
```

Pour le i2cData [0] ça correspond à l'adresse de l'appareil. Pour le i2cData [5,6] ça correspond à l'adresse du décalage entre la position de référence initiale du capteur. Pour le i2cData [7] ça correspond à l'adresse de l'état du capteur.

Nous allons nous intéresser à cette variable qui reprend la valeur de l'adresse de l'état du capteur sur 4 bits.

```
positionState = i2cData[RGBLINEFOLLOWER_STATE_ADDR] & 0x0F;
```

La variable positionstate correspond à l'état des 4 capteur RGB qui permet de déterminer s'il voit la ligne ou non sur la documentation le « 0 » signifie que la ligne est détectée et le « 1 » signifie que l'arrière-plan est détecté. Cette valeur retourne la valeur des capteurs sous la forme de 4 bits. Exemple : LED (éteint, éteint, allumer, allumer) la valeur qui va être retournée est 3.

```
la position des capteur : 3 i2cData[1]:104 i2cData[2]:115 i2cData[3]:53 i2cData[4]:15
```

On remarque donc qu'on peut aussi commander le robot avec la variable positionState qui peut être plus simple que de comparer les valeurs de 4 variables différentes. Donc Le programme qu'on va mettre dans le programme principal (**Annexe 2**. Programme test du capteur suivi de ligne)

AC22.02

III.3. Orientation du télémètre

AC23.01

L'objectif de cette partie est de contrôler le moteur du télémètre afin qu'il s'oriente à 0° , 90° , -90° . On va brancher le moteur du télémètre sur la broche digital 3.

Pour l'orientation du télémètre on reprend la version du S3. Pour réaliser l'orientation du télémètre, on va reprendre le programme du timer2 à 8 bits en mode PWM rapide mode connecter (**Annexe 3**. Programme Timer2 rotation du télémètre). Avec OCR2A qui sert à modifier la demi-période et OCR2B qui sert pour modifier la durée à l'état haut pour commander la position du moteur du télémètre. Puis sans changer le programme on effectue des tests pour vérifier la fonctionnalité du système.

Orientation du télémètre à 0° :

On obtient une durée à l'état haut de 1.440 ms

Période : 20ms

Orientation du télémètre à 90° :

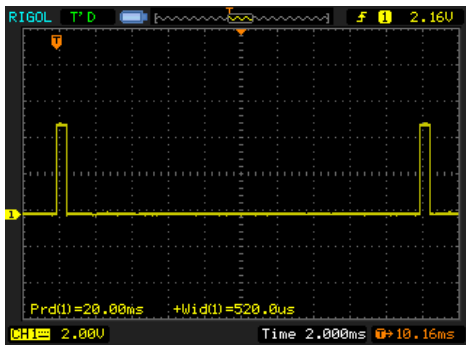


Figure 9. Valeur de OCR2B à l'état haut de 0.52 ms

Orientation du télémètre à -90° :

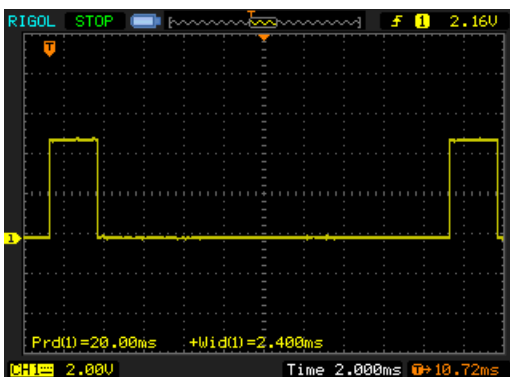


Figure 10. Valeur de OCR2B à l'état haut de 2.4ms

On obtient une durée à l'état haut de 2.4 ms

Période : 20ms

IV. Description des fonctions de haut niveau

IV.1. Mouvement du robot

Dans cette partie on va s'intéresser au différent mouvement du robot qu'on aura besoin pour la partie suivante le suivi de ligne. Pour cela on va reprendre les fonctions que nous avons créées au S3. Avec la fonction avancer (), recule (), droite (), gauche () qui correspond au robot qui avance, recule, tourne à droite, tourne à gauche (**Annexe 4**. Programme fonctions déplacement du robot). Après plusieurs essais toutes les fonctions fonctionnent correctement on peut donc passer à la partie suivante le suivi de ligne (IV.2).

IV.2. Suivi de ligne

Cette partie a pour objectif de créer notre fonction suivie de ligne qui va permettre de commander les moteurs avec les fonctions que nous avons créé dans la partie précédentes (IV.1). On va utiliser la variable positionState pour faire nos conditions pour que le robot suit la ligne. On réalise un tableau où on va mettre toutes les valeurs 4 bits que le capteur peut retourner avec les actions qui devra effectuer.

Capteur : Droite milieu milieu gauche	0 : présence de ligne 1 : couleur de fond
0000 = 0	Ligne perpendiculaire
0001 = 1	Tourne à gauche légèrement
0010 = 2	Pas possible
0011 = 3	Tourne à gauche
0100 = 4	Pas possible
0101 = 5	Pas possible
0110 = 6	Pas possible
0111 = 7	Tourne à gauche
1000 = 8	Tourne à droite légèrement
1001 = 9	Tout droit
1010 = 10	Pas possible
1011 = 11	Tourne à gauche légèrement
1100 = 12	Tourne à droite
1101 = 13	Tourne à droite légèrement
1110 = 14	Tourne à droite
1111 = 15	Tout droit

Tableau 2. Tableau conditions des moteurs en fonction de la valeur retourner de la variable positionState

Pseudocode :

Si positionState <- 9, 15

Alors

Avance ()

Sinon si positionState <- 1,11

Alors

GaucheL()

Sinon si positionState <- 8 ,13

Alors

DroiteL()

Sinon si positionState<- 3,7

Alors

Gauche ()

Sinon Si <- 12,14

Alors

Droite ()

Entrées	Sorties	
	Attendues, théoriques	Réelles, mesurées
Avancer	Positionstate = 9,15	9, 15
Droite	Positionstate = 3,7	3, 7
Gauche	Positionstate = 12,14	12, 14
DroiteL	Positionstate = 1,11	1 , 11
GaucheL	Positionstate = 8 ,13	8, 13

Tableau 3. Tableau de test Suiveur de ligne

AC 11.02, AC 12.01 , AC22.01

On va réaliser un programme test sur Arduino (**Annexe 5.** Programme exemple du suiveur de ligne) avec les conditions et des Serial.print pour vérifier si toutes les conditions sont remplies selon le tableau de test.

Puis une fois le tableau de test remplie on va déplacer les conditions dans le programme principal dans une fonction déplacement () (**Annexe 6.** Programme fonction déplacement suivi de ligne). Afin de le tester en condition réelle avec les moteurs en remplaçant les Serial.print par les fonctions qui commande les moteurs des roues.

Après plusieurs essais le programme suiveur de ligne fonctionne.

IV.3. Détecteur de couleur

Dans cette partie nous nous intéressons au capteur de couleur. On utilise le capteur TCS34725. J'ai câblé ce capteur à l'aide de la Datasheet. Le capteur se câble de cette façon :

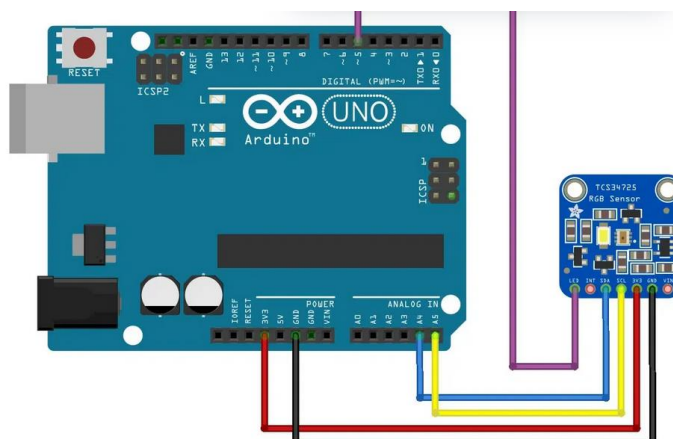


Figure 11. Câblage du capteur TCS34725

AC24.01

Puis j'ai installé les bibliothèques I2C liquid Crystal et Adafruit TCS34725 qui vont servir à avoir toutes les fonctions nécessaires pour pouvoir communiquer avec le capteur. Ensuite pour tester si la communication I2C fonctionne on va utiliser le programme d'exemple « colorview » du TCS34725 (**Annexe 7.** Programme exemple communication I2C TCS34725 capteur de couleur).

Après plusieurs essais la communication I2C ne fonctionne pas. Pour trouver le problème on va effectuer une série de tests pour identifier la cause du problème.

AC12.02, AC12.03

Identification du Dysfonctionnement	Réalisation
Problème de bibliothèques	On a réinstallé toutes les bibliothèques et le problème n'a pas été résolu.
Problème de communication avec la liaison série	On a fait un test avec un Serial.println et la communication de la liaison série fonctionne
Problème lié à l'ordinateur	On a changé d'ordinateur et on a eu le même problème.
Problème dû à la méthode d'installation	On a testé sur un ordinateur d'un autre groupe pour qui ça fonctionner et nous avons eu le même problème.
Problème de câblage	On a vérifié avec un autre groupe et le câblage était correct.

Problème du capteur TCS34725	On a échangé les capteurs avec un autre groupe et le capteur fonctionné avec eux.
Problème de carte Arduino	On a échangé la carte Arduino avec un autre groupe et le capteur fonctionné on peut en déduire que c'est un problème de la carte Arduino qu'on doit changer à la prochaine séance.

Tableau 4. Tableau de test du dysfonctionnement

AC23.03, AC23.04

Après avoir changé de carte Arduino j'ai exécuté le programme « colorview » et on observe dans le moniteur série des nombres correspondant au couleur RGB.

R:	90	G:	89	B:	71
R:	90	G:	89	B:	71
R:	90	G:	89	B:	71
R:	89	G:	89	B:	71
R:	90	G:	89	B:	71
R:	90	G:	89	B:	71
R:	90	G:	89	B:	71
R:	89	G:	89	B:	70
R:	90	G:	89	B:	70
R:	90	G:	89	B:	71
R:	90	G:	89	B:	70
R:	89	G:	89	B:	71
R:	89	G:	89	B:	70
R:	89	G:	89	B:	70
R:	89	G:	89	B:	70

Figure 12. Affichage de la couleur détectée par le capteur TCS34725

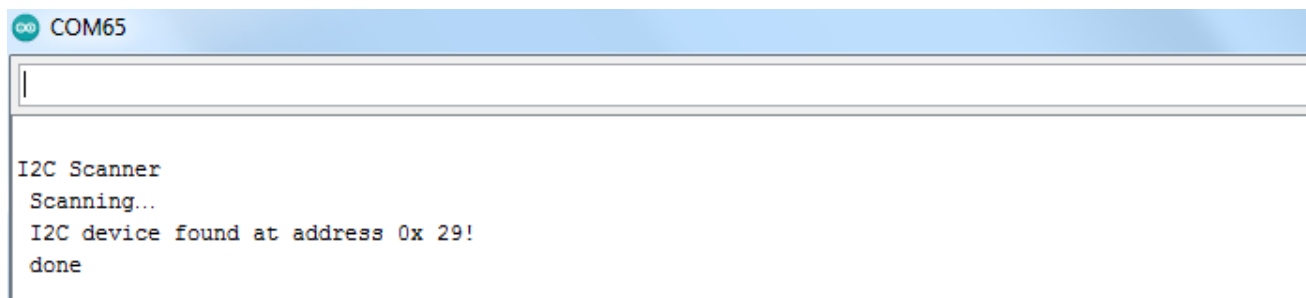
On va essayer plusieurs couleurs pour observer les résultats sur le moniteur série et voir si ça correspond aux valeurs RGB.

Entrées	Sorties	
	Attendues, théoriques	Réelles, mesurées
Couleur noir	R: 0 G:0 B: 0	R:86 G:86 B:76
Couleur rouge	R:255 G: 0 B:0	R:155 G:55 B:55
Couleur Blanc	R:255 G:255 B:255	R:180 G:255 B:211
Couleur Vert	R:0 G: 255 B:0	R:73 G:104 B:62
Couleur bleu	R:0 G:0 B:255	R:38 G:91 B:116

Tableau 5. Tableau de test du capteur de couleur TCS34725

AC22.01, AC22.02

Après plusieurs essais avec différentes couleurs on observe que le capteur détecte de façon approximative la couleur. Les attentes théoriques ne sont pas respectées, car je n'avais pas la couleur parfaitement rouge, bleu et vert. Il détecte la couleur prédominante, mais pas la couleur de façon précise.



```
COM65
I2C Scanner
Scanning...
I2C device found at address 0x 29!
done
```

Figure 13. Adresse I2C capteur de couleur

IV.4. Détection d'obstacle

Dans cette partie on s'intéresse à la détection d'obstacle qui va permettre quand le robot va détecter un obstacle :

- réduire sa vitesse à partir d'une distance de l'obstacle d'environ 15cm.
- avance doucement jusqu'à la détection de collision avec l'obstacle.
- récupère la couleur de l'obstacle.
- recule jusqu'à une distance convenable pour entamer l'évitement d'obstacle.
- évite l'obstacle et rejoint la ligne du parcours afin de reprendre le suivi de ligne

Pour réaliser la détection d'obstacle on utilise un bouton poussoir brancher en tout ou rien et le télémètre. Pour le bouton poussoir on le câble sur la broche A1 pour pouvoir prendre la valeur en sortie du bouton et pouvoir la stocker dans une variable boutonDetect qui va nous servir à faire nos prochaines conditions. Pour le télémètre on va reprendre le programme du télémètre du précédent SAE (**Annexe 8.** Programme CAN télémètre).

On réalise le pseudo code de la fonction « obstacle »

Pseudo-code :

Si tlm <40 et bouton == 0

Alors

Avance <- 5000

Si tlm <40 et bouton == 1023

Alors

Couleur() // fonction permettant de relever la couleur

Arret()

Si 0 < tlm < 40 et priseCouleur == 1

Alors

Reculer()

Si positionState ≠ 15

Alors

deplacement() // reprise du suivi de ligne

Problème : le CAN du télémètre ne fonctionne pas avec le programme repris du S3 même en refaisant le programme avec l'interruption et sans l'interruption le CAN du télémètre sort toujours une valeur de 255 sur tous les ports analogiques empêchant de faire fonctionner l'I2C du suiveur de ligne, du capteur de couleur et l'entrée du bouton poussoir. En mesurant à l'oscilloscope le signal observer est de 0 pour chaque entrée.

AC12.02, AC12.03

Solution : On remarque que le CAN de l'Arduino avec la fonction analogRead fonctionne avec le télémètre donc on conclut que c'est un problème du programme du CAN, mais qu'on ne peut pas trouver. Cependant, les mesures du télémètre ne sont pas les mêmes pour 15 cm on a une valeur de 120. Donc le pseudocode du dessus ne peut fonctionner. On va donc faire modifier le pseudo-code de la fonction obstacle en ajoutant la fonction télémètre avec la fonction analogRead qui va lire sur la broche A1 les valeurs du télémètre.

AC23.03, AC23.04

Pseudo-code de la fonction obstacle :

On change la valeur de la variable color en fonction de la couleur détecter.

Si (boutonDetect<1000 && tlm < 120)

Alors

Deplacement () // fonction du suivi de ligne

Fin si

Si (boutonDetect<1000 && tlm >120)

Alors

Ralenti() // robot doit ralentir quand il est à l'approche de l'obstacle

Fin si

Si (boutonDetect==1023 && tlm >120)

Alors

 Si (red > blue && red>green)

Alors

 color=1 // présence d'une couleur à predominance rouge detecté

 RED // couleur des capteurs rouge

Fin si

 Si (green >red&& green>blue)

Alors

 color = 2 // présence d'une couleur à predominance vert detecté

 GREEN // couleur des capteur vert

Fin si

 Si (blue > red && blue>green)

Alors

 color = 3 // présence d'une couleur à predominance bleu detecté

 BLUE // couleur des capteur bleu

Fin si

Tant que (color>0)

Tour() // fonction qui va permettre de contourner l'obstacle

Fin tant que

Fin si

Après avoir réalisé le code de la fonction obstacle sur Arduino (**Annexe 9**. Programme de la fonction obstacle), on fait un tableau de test pour pouvoir tester nos conditions.

Entrées	Sorties	
	Attendues, théoriques	Réelles, mesurées
pas présence d'obstacle	suivi de ligne	fonctionne
Présence d'obstacle	robot ralenti	fonctionne
Variable couleur	Rouge color = 1 , vert color = 2 bleu color = 3	fonctionne

Tableau 6. Tableau de test de la fonction obstacle

AC12.01, AC22.01

La fonction obstacle fonctionne, le robot exécute le suivi de ligne s'il ne détecte pas d'obstacle puis ralenti quand il le détecte et relève la couleur de l'obstacle au contact. On va donc réaliser la fonction tour qui va servir à contourner l'obstacle

Pseudo-code de la fonction tour :

La variable étape sert pour que le programme s'exécute étape par étape

Tant que (tlm>150 && etape==0)// fais reculer le robot jusqu'à une distance convenable

recule()

Si(tlm<280) //si le robot est à environ 5 cm passe à l'étape suivante

Alors

etape=1

Fin si

Fin tant que

Tant que (etape==1)

tlmd() // passe le telemetre a gauche

advanced() // avance le robot vers la droite

Fin tant que

Si tlm>350) // si le telemetre detecte l'obstacle proche passe à l'etape suivante

Alors

etape=2

Fin si

Tant que (etape==2) //tant que le télémètre détecte l'obstacle contourne jusqu'a qu'il ne détecte plus l'obstacle

Si (tlm >240) // si le télémètre est trop proche de l'obstacle

Alors

gauche()// tourne à gauche pour s'éloigner

Fin si

Sinon si (tlm < 200 && tlm > 45) // si le télémètre est trop éloigné de l'obstacle en le détectant toujours

Alors

tele()

droite() //tourne à droite pour revenir

Fin Si

Sinon si (tlm< 40) // non présence de l'obstacle passe à l'étape suivante

Alors

Tant que (etape==2) // Passe à l'étape suivante

etape=3

Fin tant que

Fin Si

Sinon

avancel() // avance en réduisant sa vitesse

Tant que (etape==3)// reprend le suivi de ligne

deplacement()

Fin tant que

Ensuite on réalise un programme qui va permettre de tester le pseudo code (**Annexe 10**. Programme de la fonction tour), pour valider les différentes conditions dans un tableau de test.

Entrées	Sorties	
	Attendues, théoriques	Réelles, mesurées
tlm >150 et etape = 0	recule	fonctionne
tlm > 280	etape = 1	fonctionne
tant que etape =1	avance vers la droite	fonctionne
tlm > 350	etape = 2	fonctionne
tlm > 240	robot tourne à gauche	fonctionne
45<tlm<200	robot tourne à droite	fonctionne
tlm < 45	etape = 3	fonctionne

Tableau 7. *Tableau de test de la fonction tour*

AC12.01, AC22.01

Le programme fonctionne avec une fiabilité de 60% car quand le télémètre tourne la valeur du télémètre peut faire un pic le faisant passer directement à l'étape 2 donc le robot n'a pas le temps d'avancer vers la droite et reste bloquer sur l'obstacle ou bien quand il avance vers la droite le télémètre ne détecte parfois pas une valeur plus grande de 350, mais si on réduit cette valeur le robot n'avancera pas assez pour pouvoir passer à coter de l'obstacle. On peut rajouter des délais, mais ça rendra le programme moins précis du fait qu'on utilise moins les capteurs et faudra changer les délais pour chaque obstacle différent.

IV.5. Suivi du Mur

Dans cette partie on s'intéresse au suivi du mur

- Détection d'une ligne perpendiculaire le télémètre tourne à gauche
- Le robot continue à suivre la ligne quand une ligne perpendiculaire est détectée
- Le robot prend le chemin qui correspond à la couleur prise par le détecteur de couleur mesurer à l'étape 2
- En fonction de la couleur le robot doit s'arrêter une distance donnée par rapport à l'obstacle de fin de parcours
- Moteur télémètre fonctionne

Pour réaliser cette partie on va commencer par réaliser la fonction du mur qui va nous servir à suivre le mur avec le télémètre quand le suivi de ligne détecte une ligne perpendiculaire. On réalise le pseudocode de la fonction du suivi du mur.

Pseudo-code de la fonction du suivi du mur :

```
Si ( positionState == 0 ) // détecte la ligne
```

```
Alors
```

```
Tant que ( tlm >35) //présence mur
```

```
    angled=1 // change la valeur de la variable angled quand il détecte la ligne
```

```
Fin si
```

```
    Si ( tlm >240) // si télémètre supérieur à 240 tourne à gauche pour s'éloigner du mur
```

```
    Alors
```

```
        gauche()
```

```
    Fin si
```

```
    Sinon si ( tlm > 45 && tlm < 200) // si télémètre est entre 45 et 200 tourne à droite pour revenir vers le mur
```

```
Alors
```

```
    droite()
```

```
Fin si
```

```
Sinon si ((positionState == 0)&& (tlm<40)) // présence d'une ligne et non présence du Mur
```

```
Alors
```

```
    Tant que ( angled ==1 ) // reprend le suivi de ligne
```

```
        etape=4// passe à l'étape suivante
```

```
        déplacement()
```

Fin tant que

Fin si

Sinon

avancel()

Fin Si

Après avoir réalisé ce pseudo-code on va le programmer sur Arduino (**Annexe 11**. Programme de la fonction suivi du mur), puis le tester avec tableau de test pour voir si toutes nos conditions fonctionnent correctement.

Entrées	Sorties	
	Attendues, théoriques	Réelles, mesurées
tIm >240	robot tourne a gauche	fonctionne
45<tIm<200	robot tourne a droite	fonctionne
tIm <45 et positionState =0	reprend suivi de ligne	fonctionne

Tableau 8.Tableau de test de la fonction du suivi du mur.

AC12.01, AC22.01

On observe que le robot suit le mur à la distance donné en condition et corrige sa trajectoire en fonction de la distance du mur. On va donc réaliser la fonction qui va permettre de choisir le chemin de fin de parcours selon la couleur que le capteur de couleur à détecter dans la partie de la fonction obstacle.

Pseudo-code de la fonction chemin :

En fonction de la couleur déterminer et mis dans la variable (« color ») sur la partie détection d'obstacle on va choisir le chemin que va prendre le robot.

//chemin à prendre en fonction de la couleur

Si ((positionState == 0) && (color == 1)) //&& (etape==4)) // Si la couleur détecter à l'etape de l'obstacle est rouge

Alors

avance()

delay(1000) // avance pendant 1 seconde

droite()// tourne à droite

delay(1250) // tourne à droite pendant 1,25 s

Tant que (color==1)

deplacement()

Si (tIm> 180)

```

    Alors
        arret() // arrêt devant l'obstacle de fin de parcours
    Fin si
Fin tant que
Fin si

    Si ((positionState == 0) && (color == 2 ))//&& (etape==4)) // Si la couleur détecter à l'etape de
l'obstacle est vert

    Alors
        avance() // avance
        delay(2500) // Avance pendant 2,5 secondes
        Tant que (color==2)
            deplacement()
            Si (tIm> 200)
                Alors
                    arret() //arrêt devant l'obstacle de fin de parcours
            Fin si
        Fin tant que
    Fin si

    Si ((positionState == 0) && (color == 3)) //&& (etape==4)) // Si la couleur détecter à l'etape de
l'obstacle est bleu

    Alors
        avance()
        delay(1000) // avance pendant 1 secondes
        gauche()// tourne a gauche
        delay(1250) // avance pendant 1,25 secondes
        Tant que (color==3)
            deplacement();
            Si (tIm> 220)
                Alors
                    arret(); // arrêt devant l'obstacle de fin de parcours
            Fin si
        Fin tant que

```

Fin si

Après avoir réalisé ce pseudo-code on va le programmer sur Arduino (**Annexe 12**. Programme de la fonction chemin) puis le tester avec tableau de test pour voir si toutes nos conditions fonctionnent correctement.

Entrées	Sorties	
	Attendues, théoriques	Réelles, mesurées
couleur rouge	tourne a droite	tourne a droite
couleur bleu	tourne a gauche	tourne a gauche
couleur vert	avance	avance

Tableau 9.Tableau de test de la fonction chemin

AC12.01, AC22.01

On observe que le robot selon la valeur qui est dans la variable couleur exécute bien le chemin de fin de parcours et s'arrête bien devant l'obstacle de fin.

V. Mise en œuvre du scénario

Le code complet se trouve dans la section (VIII.2.1). La mise en œuvre du scénario a été réalisée lors du concours. Le robot fonctionne sauf la partie de l'évitement de l'obstacle qui est dû au problème lors que le télémètre tourne. La solution à ce problème serait de mettre du délai et d'arrêter de faire la mesure pendant que le télémètre tourne pour qu'il puisse faire les étapes d'après correctement. La détection de la ligne perpendiculaire fonctionne pour permettre au robot de suivre le mur. Le chemin à prendre en fonction de la couleur fonctionne.

VI. Bilan des références aux AC

Concevoir Mener une conception intégrant une démarche projet	Compétences du PN GEII - BUT 1	Mener une conception partielle intégrant une démarche projet	AC11.01	Produire une analyse fonctionnelle d'un système simple	II.
			AC11.02	Réaliser un prototype pour des solutions techniques matériel et/ou logiciel	IV.2
			AC11.03	Rédiger un dossier de fabrication à partir d'un dossier de conception	Ce document
	Compétences du PN GEII - BUT 2	Concevoir un système en fiabilisant les solutions proposées	AC21.01	Proposer des solutions techniques liées à l'analyse fonctionnelle	
			AC21.02	Dériskuer les solutions techniques retenues	
Vérifier Effectuer les tests et mesures nécessaires à une vérification d'un système	Compétences du PN GEII - BUT 1	Effectuer les tests et mesures nécessaires à une vérification d'un Mettre en place un protocole de tests pour valider le fonctionnement d'un système	AC12.01	Appliquer une procédure d'essais	IV.2, IV.3, IV.5
			AC12.02	Identifier un dysfonctionnement	IV.3, IV.4
			AC12.03	Décrire les effets d'un dysfonctionnement	IV.3, IV.4
	Compétences du PN GEII - BUT 2		AC22.01	Identifier les tests et mesures à mettre en place pour valider le fonctionnement d'un système	IV.3, IV.2, III.2, III.1, IV.4, IV.5
			AC22.02	Certifier le fonctionnement d'un nouvel équipement industriel	III.2, IV.3
Maintenir Assurer le maintien en condition opérationnelle d'un système	Compétences du PN GEII - BUT 2	Intervenir sur un système pour effectuer une opération de maintenance	AC23.01	Exécuter l'entretien et le contrôle d'un système en respectant une procédure	III.1, III.3
			AC23.02	Exécuter une opération de maintenance (corrective, préventive, améliorative)	
			AC23.03	Diagnostiquer un dysfonctionnement dans un système	IV.4, IV.3
			AC23.04	Identifier la cause racine du dysfonctionnement	IV.4, IV.3
Intégrer Intégrer un système de commande et de contrôle	Compétences du PN GEII - BUT 2 (All)	Procéder à une installation ou à une mise en service en suivant un protocole	AC24.01All	Appliquer la procédure d'installation d'un système	III.2, IV.3
			AC24.02All	Exécuter la mise en service d'un système en respectant la procédure	III.2

Tableau 10. Tableau des références aux apprentissages critiques

AC 11.02	11
AC 12.01	11
AC11.01	4
AC12.01	17, 19, 21, 23
AC12.02	12, 15
AC12.03	12, 15
AC22.01	5, 7, 11, 13, 17, 19, 21, 23
AC22.02	8, 13
AC23.01	5, 9
AC23.03	13, 15
AC23.04	13, 15
AC24.01	12
AC24.01 All	7
AC24.02 All	7

Tableau 11. Bilan des Références AC

VII. Conclusion

Pour conclure le robot à la mise en œuvre du scénario à quelques problèmes au niveau de l'évitement de l'obstacle dû au pic du télémètre quand le télémètre tourne après qu'il recule dans la fonction tour la valeur du télémètre dans la condition est trop grande qui cause un problème de fiabilité au niveau de l'évitement de l'obstacle. En remarquant au concours la fonction mur aussi doit être modifiée au niveau de la condition de fin du suivi du mur le mettre uniquement quand il détecte la ligne perpendiculaire et non quand il y a non-présence du mur et la ligne perpendiculaire. Toutes les autres fonctions fonctionnent ensemble. Le scénario de mise en œuvre lors du concours à réaliser tout fonctionne sauf l'évitement d'obstacle le télémètre n'a pas détecté l'obstacle pour reprendre en suivi de l'obstacle pour le contourner et il a avancé tout droit et le chemin de fin de parcours qui est dû à une erreur au niveau de l'emplacement des couleurs. J'ai fini avant-dernier dans le concours, car c'était le premier test du fonctionnement du robot avec le code complet avec plusieurs problèmes qui est dû à mon retard accumulé lors de la réalisation du projet.

	Plan ...	Nom de tâche	Durée	Début	Fin
1	✧	Sujet	64 jours	21/03/2023	16/06/2023
2	✧	Maintenance du robot	1 jour	28/03/2023	28/03/2023
3	✧	Détecteur de couleur	6 jours	04/04/2023	11/04/2023
4	✧	Suiveur de ligne	17 jours	18/04/2023	10/05/2023
5	✧	Détection d'obstacle	21 jours	16/05/2023	13/06/2023
6	✧	Suivi du mur	2 jours	06/06/2023	07/06/2023
7	✧	Mise en situation du projet	1 jour	16/06/2023	16/06/2023

Figure 14. Modification du calendrier

VIII. Annexes

VIII.1. Tableau de référence aux compétences

Comp.	Niveau		Apprentissages critiques	
Concevoir Mener une conception intégrant une démarche projet	Compétences du PN GEII - BUT 1	Mener une conception partielle intégrant une démarche projet	AC11.01	Produire une analyse fonctionnelle d'un système simple
			AC11.02	Réaliser un prototype pour des solutions techniques matériel et/ou logiciel
			AC11.03	Rédiger un dossier de fabrication à partir d'un dossier de conception
	Compétences du PN GEII - BUT 2	Concevoir un système en fiabilisant les solutions proposées	AC21.01	Proposer des solutions techniques liées à l'analyse fonctionnelle
			AC21.02	Dérisquer les solutions techniques retenues
Vérifier Effectuer les tests et mesures nécessaires à une vérification d'un système	Compétences du PN GEII - BUT 1	Effectuer les tests et mesures nécessaires à une vérification d'un système	AC12.01	Appliquer une procédure d'essais
			AC12.02	Identifier un dysfonctionnement
			AC12.03	Décrire les effets d'un dysfonctionnement
	Compétences du PN GEII - BUT 2	Mettre en place un protocole de tests pour valider le fonctionnement d'un système	AC22.01	Identifier les tests et mesures à mettre en place pour valider le fonctionnement d'un système
			AC22.02	Certifier le fonctionnement d'un nouvel équipement industriel
Maintenir Assurer le maintien en condition opérationnelle d'un système	Compétences du PN GEII - BUT 2	Intervenir sur un système pour effectuer une opération de maintenance	AC23.01	Exécuter l'entretien et le contrôle d'un système en respectant une procédure
			AC23.02	Exécuter une opération de maintenance (corrective, préventive, améliorative)
			AC23.03	Diagnostiquer un dysfonctionnement dans un système
			AC23.04	Identifier la cause racine du dysfonctionnement
Intégrer Intégrer un système de commande et	Compétences du PN GEII - BUT 2 (All)	Procéder à une installation ou à une mise en service en suivant un protocole	AC24.01 All	Appliquer la procédure d'installation d'un système
			AC24.02 All	Exécuter la mise en service d'un système en respectant la procédure

Tableau 12. Référentiel de compétences

VIII.2. Codes

```
ISR(TIMER1_OVF_vect)
{
    PORTD = PORTD | (1<<5) | (1<<6); ; // met à 1 le broche 5 et 6 du port D
}

ISR(TIMER1_COMPA_vect) // Signal pwm Moteur 1
{
    PORTD = PORTD & ~(1<<5); // complement de la broche 5
}

ISR(TIMER1_COMPB_vect) // Signal pwm Moteur 2
{
    PORTD = PORTD & ~(1<<6); // complement de la broche 6
}

void setup() {

    // mode connecter

    DDRD = 0xff;
    TCCR1A = 0x02; // pwm non connecter
    TCCR1B = 0x1A; // pre div par 8 en mode 14 pwm fast
    TIFR1 = TIFR1 | (1<<0); // mis a 1 du flag tov
    TIMSK1 = 0x07; // activation des interruption
    ICR1 = 40000; // periode de 20 ms
    sei();
    OCR1A = 10000; // etat haut moteur 1
    OCR1B = 10000; // etat haut moteur 2

}

//char c=0;
void loop()
{

    [ ]
}
```

Annexe 1. Programme Timer1 commande des roues

```

#include "MeRGBLineFollower.h"
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

MeRGBLineFollower RGBLineFollower(13,12, ADDRESS1);
int16_t turnoffset = 0;
uint8_t positionState = 0;

void setup()
{
  RGBLineFollower.begin();
  RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN); // Change la couleur des capteur en vert
  Serial.begin(9600);
}

void loop()
{
  RGBLineFollower.loop();
  turnoffset = RGBLineFollower.getPositionOffset(); // Position de référence initiale des capteur
  positionState= RGBLineFollower.getPositionState(); // Retourne la valeur sous 4 bits de l'etat des capteurs
  Serial.print("position ");
  Serial.println(positionState);
}

```

Annexe 2. Programme test du capteur suivi de ligne

```

void setup()
{
  DDRD = 0xff;
  TCCR2A = 0xF3;
  TCCR2B = 0x0F;
  OCR2A = 200;
  OCR2B = 178; // 165 179 192
}

```

Annexe 3. Programme Timer2 rotation du télémètre

```

void arret ()
{
  OCR1A = 20; // etat haut moteur 1
  OCR1B = 20; // etat haut moteur 2
}

void avance()
{
  PORTD = PORTD&~(1<<4);
  PORTD = PORTD&~(1<<7);
  OCR1A = 15000; // etat haut moteur 1
  OCR1B = 15000; // etat haut moteur 2
}

void recule ()
{
  PORTD = PORTD|(1<<4);
  PORTD = PORTD|(1<<7);

  OCR1A = 10000; // etat haut moteur 1
  OCR1B = 10000; // etat haut moteur 2
}

void gauche()
{
  PORTD = PORTD&~(1<<4);
  PORTD = PORTD&~(1<<7);
  OCR1A = 5; // etat haut moteur 1
  OCR1B = 10000; // etat haut moteur 2
}

void droite()
{
  PORTD = PORTD&~(1<<4);
  PORTD = PORTD&~(1<<7);
  OCR1A = 10000; // etat haut moteur 1
  OCR1B = 5; // etat haut moteur 2
}

```

Annexe 4. Programme fonctions déplacement du robot

```

//Serial.println(positionState);

if ( positionState == 9 || positionState == 15)
{
  Serial.println("avance");
  //avance();
}

if (positionState == 1 || positionState == 11)
{
  Serial.println("droite");
  //droiteL();
}

if (positionState == 8 || positionState == 13)
{
  Serial.println("gauche");
  //gaucheL();
}

if (positionState == 3 || positionState == 7)
{
  Serial.println("droite");
  //droite();
}

if (positionState == 12 || positionState == 14)
{
  Serial.println("gauche");
  //gauche();
}

```

Annexe 5. Programme exemple du suiveur de ligne

```

void deplacement()
{
    suivi(); // fonction capteur suivi de ligne

    if ( positionState == 9 || positionState == 15)
    {
        //Serial.println("avance");
        avance();
    }

    if (positionState == 1 || positionState == 11)
    {
        //Serial.println("droite");
        gaucheL();
    }

    if (positionState == 8 || positionState == 13)
    {
        //Serial.println("gauche");
        droiteL();
    }

    if (positionState == 3 || positionState == 7)
    {
        //Serial.println("droite");
        gauche();
    }

    if (positionState == 12 || positionState == 14)
    {
        //Serial.println("gauche");
        droite();
    }
}

```

Annexe 6. Programme fonction déplacement suivi de ligne

```

#include <Wire.h>
#include "Adafruit_TCS34725.h"

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X); //variable globale tcs

void setup()
{
    Serial.begin(9600);
    //Serial.println("Color View Test!");
    //Lancement de l'I2C
    if (tcs.begin()) {
        //Serial.println("Capteur trouvé");
    } else {
        Serial.println("pas de TCS34725 trouvé ... Vérifiez votre connexion");
        while (1); // stop!
    }
}

// [0-255] valeur RVB
void loop()
{
    float red, green, blue;
    delay(60); // Attente de 60 ms pour lire
    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.print(int(blue));
    Serial.print("\n");
}

```

Annexe 7. Programme exemple communication I2C TCS34725 capteur de couleur

```

char tele=0x02;
ISR(ADC_vect)
{
    ADMUX= (1<<ADLAR)|tele;
    ADCSRA=ADCSRA|(1<<ADSC)|0x07;
    tlm = ADCH;
    Serial.println(tlm);
}

void setup()
{
    Serial.begin(9600);
    DDRC = 0;
    ADCSRA=(1<<ADEN)|(1<<ADIE);
    sei();
    ADCSRA=ADCSRA|(1<<ADSC)|0x07;
}

```

Annexe 8. Programme CAN télémètre

```

void obstacle()
{
    detectionObstacle(); // fonction bouton poussoir
    tele(); // fonction capteur du telemetre
    couleur(); // fonction capteur de couleur

    if ( boutonDetect<1000 && tlm <120 ) // si le bouton n'est pas appuyer et le telemetre ne detecte pas d'obstacle
    {
        deplacement(); // suivi de ligne
    }
    if (boutonDetect<1000 && tlm> 120) // si le bouton n'est pas appuyer et le telemetre detecte un obstacle
    {
        //avance en reduisant sa vitesse
        avance1();
    }

    if (boutonDetect>1000) // si le bouton est appuyer (contact avec l'obstacle)
    {
        // detection de la couleur et change la valeur de la variable couleur en fonction de la valeur du capteur de couleur
        if ((red > blue) && (red>green) )
        {
            color=1; // présence d'une couleur à predominance rouge detecté
            RGBLineFollower.setRGBColour(RGB_COLOUR_RED); // couleur des capteurs rouge
        }

        if ((green >red)&& (green>blue))
        {
            color = 2; // présence d'une couleur à predominance vert detecté
            RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN); // couleur des capteurs vert
        }

        if ((blue > red) && (blue>green))
        {
            color = 3; // présence d'une couleur à predominance bleu detecté
            RGBLineFollower.setRGBColour(RGB_COLOUR_BLUE); // couleur des capteurs bleu
        }

        while(color >=1)// Vérifie qu'une couleur à était prise par le capteur avant de passer à l'etape suivante
        {
            tour(); // Fonction pour contourner l'obstacle
        }
    }
}

```

Annexe 9. Programme de la fonction obstacle

```

void tour()
{
    tele();
    // contournement d'obstacle//////////
    while ( tlm>150 && etape==0) // fais reculer le robot jusqu'a une distance convenable
    {
        tele();
        recule();
        if (tlm<280) //si le robot est a environ 5 cm passe a l'etape suivante
        {
            etape=1;
        }
    }
    while(etape==1)
    {
        tlm(); // passe le telemetre a gauche
        tele();
        avance(); // avance le robot vers la droite

        if(tlm>350) // si le telemetre detecte l'obstacle proche passe à l'etape suivante
        {
            etape=2;
        }
    }
    while(etape==2) //tant que le telemetre detecte l'obstacle contourne jusqu'a qu'il ne detecte plus l'obstacle
    {
        tele();
        if ( tlm >240) // si le telemetre est trop proche de l'obstacle
        {
            tele();
            gauche();// tourne à gauche pour s'eloigner
        }
        else if ((tlm < 200) && (tlm > 45)) // si le telemetre est trop eloigné de l'obstacle en le détectant toujours
        {
            tele();
            droite(); //tourne à droite pour revenir
        }
        else if ( tlm< 40) // non présence de l'obstacle passe à l'etape suivante
        {
            tele();
            while (etape==2) // Passe à l'etape suivante
            {
                etape=3;
            }
        }
    }
}

    while (etape==2) // Passe à l'etape suivante
    {
        etape=3;
    }
    else
    {
        tele();
        avance(); // avance en réduisant sa vitesse
    }
}

while(etape==3)// reprend le suivi de ligne
{
    deplacement();
}
}

```

Annexe 10. Programme de la fonction tour


```

void mur()
{
    suivi();
    tele();
    tlm();

    if ( positionState == 0 ) // detecte la ligne
    {
        while ( tlm > 35 ) //pr sence mur
        {
            angled=1;
            tele();

            if (tlm >240) // si telemetre sup rieur a 240 tourne a gauche pour s'eloigner du mur
            {
                gauche();
            }

            else if ( ( tlm > 45) &&(tlm < 200)) // si telemetre est entre 45 et 200 tourne a droite pour revenir vers le mur
            {
                droite();
            }

            else if ((positionState == 0 )&& (tlm<40)) // pr sence d'une ligne et non pr sence du Mur
            {
                while ( angled ==1 ) // reprend le suivi de ligne
                {
                    etape=4; // passe   l'etape suivante
                    Serial.println(etape);
                    deplacement();
                }
            }
            else
            {
                avancel();
            }
        }
    }
}

```

Annexe 11. Programme de la fonction suivi du mur

```

void chemin()
{
    suivi();
    tlmf(); // remet le telemetre en face
    //chemin a prendre en fonction de la couleur
    if ((positionState == 0) && (color == 1) && (etape==4)) // Si la couleur d tecter   l'etape de l'obstacle est rouge
    {
        avance(); // avance
        delay(2500);
        while(color==1)
        {
            tele();
            deplacement();
        }
        if (tlm> 180)
        {
            arret();
        }
    }

    if ((positionState == 0) && (color == 2) &&(etape==4)) // Si la couleur d tecter   l'etape de l'obstacle est vert
    {
        avance();
        delay(1000);
        gauche(); // tourne   droite
        delay(1250);
        while(color==2)
        {
            tele();
            deplacement();
            if (tlm> 200)
            {
                arret();
            }
        }
    }

    if ((positionState == 0) && (color == 3) &&(etape==4)) // Si la couleur d tecter   l'etape de l'obstacle est bleu
    {
        avance();
        delay(1000);
        droite(); // tourne a gauche
        delay(1250);
        while(color==3)
        {
            tele();
            deplacement();
            if (tlm> 220)
            {
                arret();
            }
        }
    }
}

```

Annexe 12. Programme de la fonction chemin

VIII.2.1. Code complet

```
1 #include <Arduino.h>
2 #include <Wire.h>
3 // bibliotheque capteur suivi de ligne
4 #include "MeRGBLineFollower.h"
5 #include <SoftwareSerial.h>
6 MeRGBLineFollower RGBLineFollower(13,12, ADDRESS1);
7 int16_t turnoffset = 0;
8 uint8_t positionState = 0;
9 // bibliotheque capteur de couleur
10 #include "Adafruit_TCS34725.h"
11 Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);
12 //variable////////////////////////////////////
13 int tlm;
14 int boutonDetect = 0, color =0, etape =0,angled=0;
15 float red, green, blue;
16
17 //Interruption Timer1
18 ISR(TIM1_OVF_vect)
19 {
20   PORTD = PORTD | (1<<5)|(1<<6); // met à 1 le broche 5 et 6 du port D
21   //ADCSRA=ADCSRA|(1<<ADSC)|0x07;
22 }
23
24
25 ISR(TIM1_COMPA_vect) // Signal pwm Moteur 1
26 {
27   PORTD = PORTD & ~(1<<5); // complement de la broche 5
28 }
29
30
31
32 ISR(TIM1_COMPB_vect) // Signal pwm Moteur 2
33 {
34   PORTD = PORTD & ~(1<<6); // complement de la broche 6
35 }
36
37
38
39 void setup()
40 {
41   DDRD = 0xff;
42   TCCR1A = 0x02; // pwm non connecter
43   TCCR1B = 0x1A; // pre div par 8 en mode 14 pwm fast
44   TIFR1 = TIFR1|(1<<0); // mis a 1 du flag tov
45   TIMSK1 = 0x07; // activation des interruption
46   ICR1 = 40000; // periode de 20 ms
47   OCR1A = 20; // initialisation a l'arret du moteur
48   OCR1B = 20; // initialisation a l'arret du moteur
49
50   TCCR2A = 0xf3;
51   TCCR2B = 0x0F;
52   // TIMSK2 = 0x07;
53   OCR2A = 200; // 35 22 8
54   OCR2B = 179; // 90° 165 0° 179 -90° 192
55
56
57   //Line follower//
58   RGBLineFollower.begin();// lancement de l'i2c du capteur suivi de ligne
59   RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN); // couleur des LED des capteur
60
61   // capteur de couleur
62   // lancement i2c du capteur de couleur
63   if (tcs.begin())
64   {
65     //Serial.println("Found sensor");
66   } else {
67     Serial.println("No TCS34725 found ... check your connections");
68     while (1); // stop!
69   }
70
71   Serial.begin(9600);
72 }
73
74 void loop()
75 {
76   obstacle();
77   delay(1);
78 }
79
80
81
```

```

83 void deplacement()
84 {
85     suivi(); // fonction capteur suivi de ligne
86     tele(); // fonction telemetre
87     if ((etape==3) && (angled==0)) // si l'etape de l'obstacle est fini on active la fonction du suivi du mur
88     {
89         mur();
90     }
91     if (etape==4)
92     {
93         chemin();
94     }
95     if ( positionState == 9 || positionState == 15 )
96     {
97         avance();
98     }
99     if (positionState == 1 || positionState == 11)
100    {
101        gauchel();
102    }
103
104    if (positionState == 8 || positionState == 13)
105    {
106        droitel();
107    }
108
109    if (positionState == 3 || positionState == 7)
110    {
111        gauche();
112    }
113
114    if (positionState == 12 || positionState == 14)
115    {
116        droite();
117    }
118 }

```

```

120
121 void obstacle()
122 {
123     detectionObstacle(); // fonction bouton poussoir
124     tele(); // fonction capteur du telemetre
125     couleur(); // fonction capteur de couleur
126     if ( boutonDetect<1000 && tlm <120 ) // si le bouton n'est pas appuyer et le telemetre ne detecte pas d'obstacle
127     {
128         deplacement(); // suivi de ligne
129     }
130     if (boutonDetect<1000 && tlm> 120) // si le bouton n'est pas appuyer et le telemetre detecte un obstacle
131     {
132         //avance en reduisant sa vitesse
133         avancel();
134     }
135     if (boutonDetect>1000) // si le bouton est appuyer (contact avec l'obstacle)
136     {
137         // detection de la couleur et change la valeur de la variable couleur en fonction de la valeur du capteur de couleur
138
139         if ((red > blue) && (red>green) )
140         {
141             color=1; // présence d'une couleur à predominance rouge detecté
142             RGBLineFollower.setRGBColour(RGB_COLOUR_RED); // couleur des capteurs rouge
143         }
144
145         if ((green >red)&& (green>blue))
146         {
147             color = 2; // présence d'une couleur à predominance vert detecté
148             RGBLineFollower.setRGBColour(RGB_COLOUR_GREEN); // couleur des capteurs vert
149         }
150
151         if ((blue > red) && (blue>green))
152         {
153             color = 3; // présence d'une couleur à predominance bleu detecté
154             RGBLineFollower.setRGBColour(RGB_COLOUR_BLUE); // couleur des capteurs bleu
155         }
156
157         while(color >=1)// Vérifie qu'une couleur à était prise par le capteur avant de passer à l'etape suivante
158         {
159             tour(); // Fonction pour contourner l'obstacle
160         }
161     }
162 }

```

```

164
165 void tour()
166 {
167     Serial.println(etape);
168     tele();
169
170     while ( tlm>150 && etape==0)// fais reculer le robot jusqu'a une distance convenable
171     {
172         tele();
173         recule();
174         if (tlm<200) //si le robot est a environ 5 cm passe a l'etape suivante
175         {
176             etape=1;
177             Serial.println(etape);
178         }
179         while(etape==1)
180         {
181             tlmD(); // passe le telemetre a gauche
182             tele();
183             avanceD(); // avance le robot vers la droite
184
185             if(tlm>350) // si le telemetre detecte l'obstacle proche passe à l'etape suivante
186             {
187                 etape=2;
188                 Serial.println(etape);
189             }
190         }
191         while(etape==2) //tant que le telemetre detecte l'obstacle contourne jusqu'a qu'il ne detecte plus l'obstacle
192         {
193             tele();
194             if (tlm >240) // si le telemetre est trop proche de l'obstacle
195             {
196                 tele();
197                 gauche(); // tourne à gauche pour s'eloigner
198             }
199             else if ((tlm < 200) && (tlm > 45)) // si le telemetre est trop éloigné de l'obstacle en le détectant toujours
200             {
201                 tele();
202                 droite(); //tourne à droite pour revenir
203             }
204             else if ( ( tlm< 40)) // non présence de l'obstacle passe à l'etape suivante
205             {
206                 tele();
207                 while (etape==2) // Passe à l'etape suivante
208                 {
209                     etape=3;
210                     Serial.println(etape);

```

```

211         }
212     }
213     else
214     {
215         tele();
216         avanceL(); // avance en réduisant sa vitesse
217     }
218 }
219
220 while(etape==3)// reprend le suivi de ligne
221 {
222     deplacement();
223 }
224 }
225
226
227 void mur()
228 {
229     suivi();
230     tele();
231     tlmD();
232
233     if ( positionState == 0 ) // detecte la ligne
234     {
235         while ( tlm >35) //présence mur
236         {
237             angled=1;
238             tele();
239
240             if (tlm >240) // si telemetre supérieur a 240 tourne a gauche pour s'eloigner du mur
241             {
242                 gauche();
243             }
244
245             else if ( (tlm > 45) &&(tlm < 200)) // si telemetre est entre 45 et 200 tourne a droite pour revenir vers le mur
246             {
247                 droite();
248             }
249
250             else if ((positionState == 0 )&& (tlm<40)) // présence d'une ligne et non présence du Mur
251             {
252                 while ( angled ==1 ) // reprend le suivi de ligne
253                 {
254                     etape=4; // passe à l'etape suivante
255                     Serial.println(etape);
256                     deplacement();
257                 }
258             }

```

```

256         {
257             {
258                 }
259             else
260             {
261                 avance();
262             }
263         }
264     }
265 }
266
267
268 void chemin()
269 {
270     suivi();
271     tlmf(); // remet le telemetre en face
272     //chemin a prendre en fonction de la couleur
273     if ((positionState == 0) && (color == 1) && (etape==4)) // Si la couleur detecter à l'etape de l'obstacle est rouge
274     {
275         avance(); // avance
276         delay(2500);
277         while(color==1)
278         {
279             tele();
280             deplacement();
281         }
282         if (t1m> 180)
283         {
284             arret();
285         }
286     }
287     if ((positionState == 0) && (color == 2) && (etape==4)) // Si la couleur detecter à l'etape de l'obstacle est vert
288     {
289         avance();
290         delay(1000);
291         gauche(); // tourne à droite
292         delay(1250);
293         while(color==2)
294         {
295             tele();
296             deplacement();
297             if (t1m> 200)
298             {
299                 arret();
300             }
301         }
302     }
303 }

```

```

304     if ((positionState == 0) && (color == 3) && (etape==4)) // Si la couleur detecter à l'etape de l'obstacle est bleu
305     {
306         avance();
307         delay(1000);
308         droite(); // tourne a gauche
309         delay(1250);
310         while(color==3)
311         {
312             tele();
313             deplacement();
314             if (t1m> 220)
315             {
316                 arret();
317             }
318         }
319     }
320 }
321
322
323 void t1mg()
324 {
325     OCR2B = 192;
326 }
327
328
329 void t1md()
330 {
331     OCR2B = 166;
332 }
333
334 void t1mf()
335 {
336     OCR2B= 179;
337 }
338 void arret ()
339 {
340     OCR1A = 20; // etat haut moteur 1
341     OCR1B = 20; // etat haut moteur 2
342 }
343
344 void avance()
345 {
346     PORTD = PORTD&~(1<<4);
347     PORTD = PORTD&~(1<<7);
348     OCR1A = 15000; // etat haut moteur 1
349     OCR1B = 17000; // etat haut moteur 2
350 }

```

```

352 ,
353 void avance1()
354 {
355     PORTD = PORTD&~(1<<4);
356     PORTD = PORTD&~(1<<7);
357     OCR1A = 10000; // etat haut moteur 1
358     OCR1B = 10000; // etat haut moteur 2
359 }
360 void recule ()
361 {
362     PORTD = PORTD|(1<<4);
363     PORTD = PORTD|(1<<7);
364     OCR1A = 15000; // etat haut moteur 1
365     OCR1B = 15000; // etat haut moteur 2
366 }
367
368 void gauche()
369 {
370     PORTD = PORTD&~(1<<4);
371     PORTD = PORTD&~(1<<7);
372     OCR1A = 5; // etat haut moteur 1
373     OCR1B = 10000; // etat haut moteur 2
374 }
375
376 void droite()
377 {
378     PORTD = PORTD&~(1<<4);
379     PORTD = PORTD&~(1<<7);
380     OCR1A = 10000; // etat haut moteur 1
381     OCR1B = 5; // etat haut moteur 2
382 }
383
384 void gauchel()
385 {
386     PORTD = PORTD&~(1<<4);
387     PORTD = PORTD&~(1<<7);
388     OCR1A = 5; // etat haut moteur 1
389     OCR1B = 9500; // etat haut moteur 2
390 }
391
392 void avance2()
393 {
394     PORTD = PORTD&~(1<<4);
395     PORTD = PORTD&~(1<<7);
396     OCR1A = 8000; // etat haut moteur 1
397     OCR1B = 15000; // etat haut moteur 2
398 }

```

```

399 ,
400 void droitel()
401 {
402     PORTD = PORTD&~(1<<4);
403     PORTD = PORTD&~(1<<7);
404     OCR1A = 9500; // etat haut moteur 1
405     OCR1B = 5; // etat haut moteur 2
406 }
407 void detectionObstacle()
408 {
409     boutonDetect= analogRead(A0);
410     // Serial.println(boutonDetect);
411 }
412
413 void suivi()
414 {
415     RGBLineFollower.loop();
416     turnoffset = RGBLineFollower.getPositionOffset();
417     positionState= RGBLineFollower.getPositionState();
418     //Serial.print("position ");
419     //Serial.println(positionState);
420 }
421
422 void couleur()
423 {
424     delay(10); // Attente de 60 ms pour lire
425     tcs.getRGB(&red, &green, &blue);
426 }
427
428 void tele()
429 {
430     //telemetre
431     tlm = analogRead(A1);
432     //Serial.println(tlm);
433     delay(60);
434 }
435

```