Robot MARK

Rapport de projet

KERVICHE BALDE BRITEL
CCSE3

Année 2022 - 2023

POLYTECH NANTES

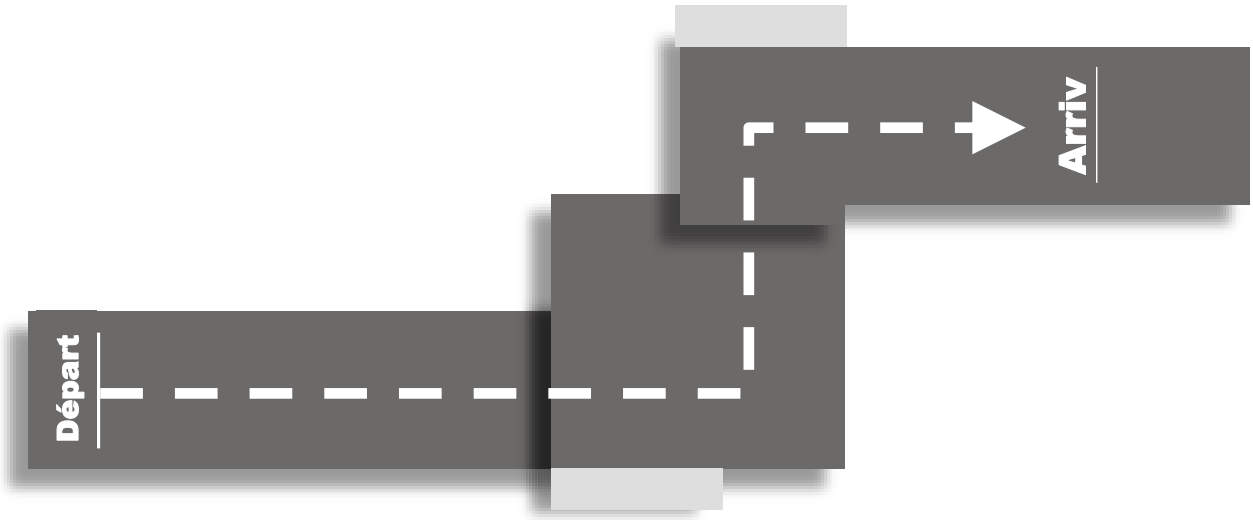
Pôle Sciences et technologie
Nantes Université

Contexte du Projet

Dans le cadre de notre formation Contrôle Commande des Systèmes Électriques nous nous sommes vu confier un projet avec le robot “robot MARK” basé sur une carte Arduino.

L’objectif du projet est la programmation d’un déplacement autonome d’un robot entre deux points A et B. Ce déplacement doit être en partie adaptatif et être capable d’éviter un obstacle imprévu placé aléatoirement sur le parcours.

Nous avons en notre possession le cahier des charges qui va nous permettre de remplir le fonctionnement et les contraintes du robot.



Sommaire

>> Les Objectifs à atteindre ---------- p. 4

>> Organisation et Planification ---------- p. 5

> Découverte et prise en main ---------- p. 5
> Programmation ---------- p. 5
> Finitions et Compte-Rendu ---------- p. 5
> Planning Gantt ---------- p. 6

>> Solutions et Résultats ---------- p. 7

> Programmation et techniques ---------- p. 7
> Tracé réel du robot ---------- p. 16
> Analyse et Critique ---------- p. 16
> Autres solutions ---------- p. 17
> Grafcet ---------- p. 18

>> Bilan et Analyse ---------- p. 19

> Bilan Technique ---------- p. 19
> Difficultés rencontrées ---------- p. 19
> REX Technique et humain ---------- p. 19

>> Conclusion ---------- p. 20

Annexes ---------- p. 21

- GitHub ---------- p. 21
- FAST---------- p. 22
- Code Complet---------- p. 25

>> Objectifs à atteindre

Avant toutes choses, l'appropriation du projet se fait par la compréhension des objectifs qui le composent, objectifs qui sont définis par le cahier des charges :

> Se lancer sur un ordre de marche fourni par l'utilisateur
> Ne pas toucher les murs
> Rester à une distance de 20 cm des murs
> Se déplacer au centre du couloir
> Atteindre la cible en moins de 10 min
> Fournir à la fin du parcours les données : Consommation, Nb tours (Roue Gauche et Roue Droite), distance parcourus, temps de parcours, le nombre de fois où le robot est <20 cm des murs

Une fois les objectifs du cahier des charges, dégagés, nous pouvons réfléchir aux objectifs sous-jacents, c'est-à-dire les objectifs qui ne sont pas forcément écrits mais qui permettent l'optimisation ou l'amélioration du robot par exemple :

> Suivre le parcours le plus court
> Finir le trajet en un minimum de temps

Finalement il reste une dernière catégorie d'objectifs à remplir, et c'est surement la plus importante, ce sont les objectifs de sécurité. Également définit par le cahier des charges ils évitent au robot les situations dangereuses, même si notre code venait à être dysfonctionnel :

> Arrêt si proximité trop importante avec un mur (5cm)
> Arrêt si contact avec un mur
> Arrêt si l'on détecte un blocage mécanique des roues
> Arrêt si on détecte un courant supérieur à la normal
> Arrêt si la vitesse du robot dépasse la limite fixer de 200°/s

>> Organisation et Planification

Découverte et prise en main

Les premières séances projets ont été consacrées à la prise en main du projet, nous avons fait le choix de travailler ensemble durant cette phase. En effet l'ensemble du groupe étant issu de formation disparate nous n'avons pas le même bagage technique notamment en ce qui concerne le Langage C. Cette phase de travail en commun a permis un échange de nos connaissances afin de rééquilibrer un peu la balance.

Cette période de prise en main du robot a été l'occasion d'établir les composantes majeures de celui-ci, capteur, actionneur, afficheur. Dans le même temps, nous pouvions déjà échanger sur notre vision du projet et de l'utilisation de tels ou tels capteurs pour répondre aux exigences du cahier des charges.

Et justement en évoquant le cahier des charges, ce dernier a été décortiqué, analysé pendant cette période d'introduction. Nous avons pu déterminer les différents objectifs (Lister à la page précédente) définis dans celui-ci.

Programmation

Le cahier des charges analysé, les objectifs et contraintes clairement définis, nous avons pu nous atteler aux premières phases de programmation.

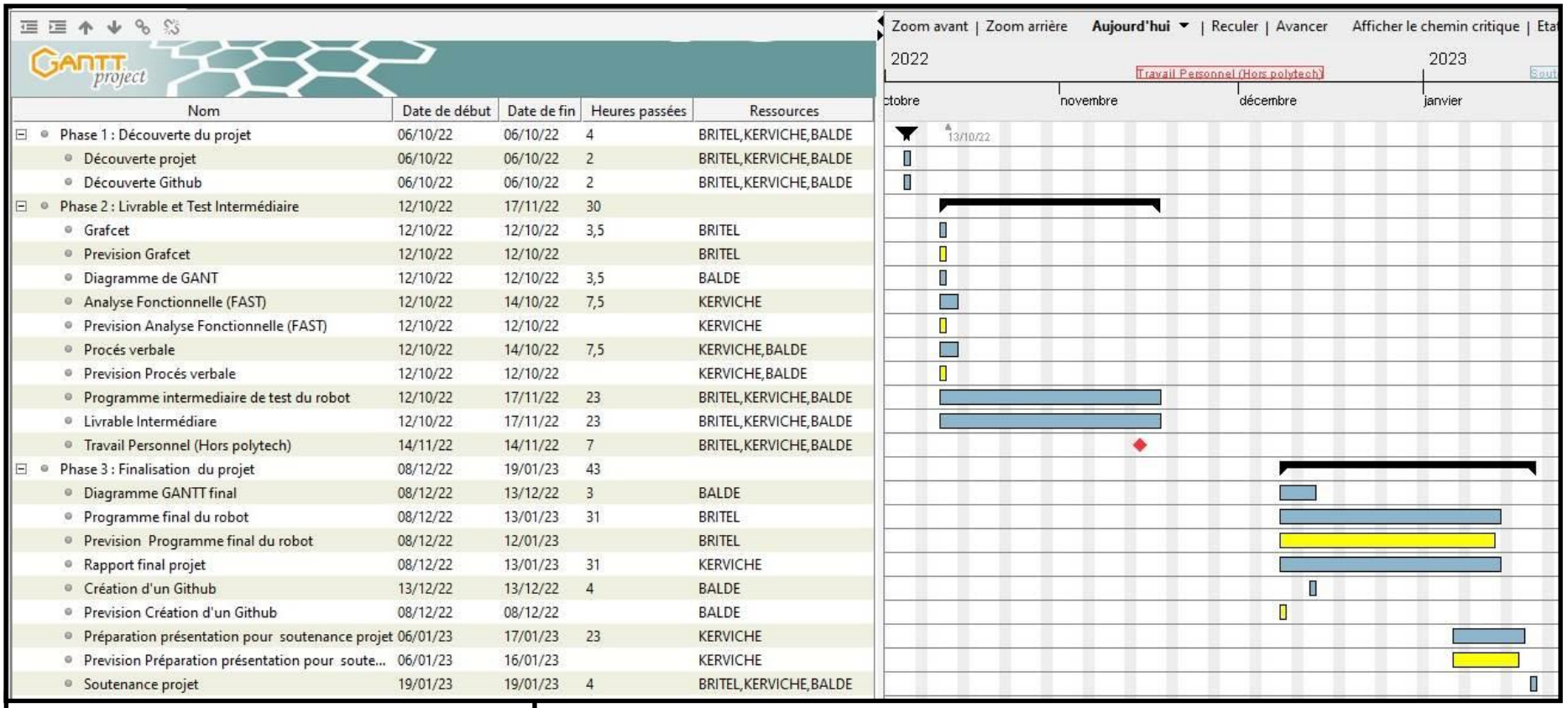
Comme évoquer plutôt la programmation en langage C++, est quelque chose de familier à 2 des membres de notre trinôme, c'est pourquoi nous avons entamé la programmation du robot en restant sur un fonctionnement de groupe unique afin de pouvoir transmettre des bases essentielles de programmation au troisième membre. Cette période a été l'occasion d'utiliser, de tester les différents capteurs, afficheurs afin de s'approprier leur fonctionnement et leurs règles de programmation.

Une fois la prise en main de la programmation en C++ par l'ensemble des membres du trinôme, nous avons commencé à nous répartir les tâches. Pour commencer deux personnes on a poursuivi la partie programmation du robot en mettant en pratique l'utilisation des capteurs pour diriger les premiers pas du robot. Pendant ce temps la troisième personne quant à elle a commencé à élaborer un planning prévisionnel du projet.

Finitions et Compte-Rendu

A mi-parcours, notre robot était capable de suivre l'ensemble du parcours défini. Restait alors à gérer le fonctionnement de l'arrêt sur la ligne d'arrivée (Gestion du capteur InfraRed), gestion de l'affichage des formations de fin de parcours (Nb tour, temps, etc…), et finalement gestion du programme d'arrêt d'urgence sur détection d'un défaut courant, qui n'avait pas été réussi précédemment. La fin du projet approchant, nous avons réorganisé les tâches, ainsi une personne est restée en charge de la programmation C++ avec de l'aide occasionnelle des deux autres membres du trinôme. Une deuxième personne à commencer la rédaction ainsi que la mise en forme du rapport et de la soutenance. La troisième personne s'occupa de son côté de la création et de la gestion du GITHUB, ainsi que du suivi du planning Gantt à la suite des modifications de planning.

Planning Gantt



Nom	Date de début	Date de fin	Heures passées	Ressources
Phase 1 : Découverte du projet	06/10/22	06/10/22	4	BRITEL,KERVICHE,BALDE
Découverte projet	06/10/22	06/10/22	2	BRITEL,KERVICHE,BALDE
Découverte Github	06/10/22	06/10/22	2	BRITEL,KERVICHE,BALDE
Phase 2 : Livrable et Test Intermédiaire	12/10/22	17/11/22	30	
Grafcet	12/10/22	12/10/22	3,5	BRITEL
Prevision Grafcet	12/10/22	12/10/22		BRITEL
Diagramme de GANT	12/10/22	12/10/22	3,5	BALDE
Analyse Fonctionnelle (FAST)	12/10/22	14/10/22	7,5	KERVICHE
Prevision Analyse Fonctionnelle (FAST)	12/10/22	12/10/22		KERVICHE
Procés verbale	12/10/22	14/10/22	7,5	KERVICHE,BALDE
Prevision Procés verbale	12/10/22	12/10/22		KERVICHE,BALDE
Programme intermediaire de test du robot	12/10/22	17/11/22	23	BRITEL,KERVICHE,BALDE
Livrable Intermédiare	12/10/22	17/11/22	23	BRITEL,KERVICHE,BALDE
Travail Personnel (Hors polytech)	14/11/22	14/11/22	7	BRITEL,KERVICHE,BALDE
Phase 3 : Finalisation du projet	08/12/22	19/01/23	43	
Diagramme GANTT final	08/12/22	13/12/22	3	BALDE
Programme final du robot	08/12/22	13/01/23	31	BRITEL
Prevision Programme final du robot	08/12/22	12/01/23		BRITEL
Rapport final projet	08/12/22	13/01/23	31	KERVICHE
Création d'un Github	13/12/22	13/12/22	4	BALDE
Prevision Création d'un Github	08/12/22	08/12/22		BALDE
Préparation présentation pour soutenance projet	06/01/23	17/01/23	23	KERVICHE
Prevision Préparation présentation pour soute...	06/01/23	16/01/23		KERVICHE
Soutenance projet	19/01/23	19/01/23	4	BRITEL,KERVICHE,BALDE

: Planning Réel

: Planning Prévisionnel

>> Solutions et résultats

Programmation et techniques

Comme évoqué précédemment, une seule personne avait déjà programmé sur Arduino dans notre groupe, ce qui a rendu difficile le début du projet.

Nous avons séparé notre programme en 3 grandes parties afin que ce soit plus compréhensible pour celui qui regarde le programme.

Dans un premier temps la définition des constantes, des variables ainsi que des bibliothèques utilisées.

```
//Valeurs Prédéfinies
#define Thash 800
#define Stop 400
#define Vmax Thash
#define VmoyG 700
#define VmoyD 730
#define Angle90 420
#define Imoy 490

// Macros
#define LedToggle digitalWrite(13, !digitalRead(13))
#define MoteurG(Vg) OCR5A=Vg // Vg in [0... 1999]
#define MoteurD(Vd) OCR5B=Vd // VD in [0... 1999]
#define MoteurGD(Vg,Vd) MoteurG(Vg);MoteurD(Vd)
#define StopMoteurGD MoteurGD(Stop,Stop)

//Bibliothèques
#include <MARK.h> //librairie du robot MARK
#include <Wire.h>  //lib for I2C connection
#include "Ultrasonic.h" //Pour les Ultrasons
#include "rgb_lcd.h" //Pour le LCD
#include <Encoder.h> //Librairie des encodeurs
MARK myrobot;

//Déclaration des capteurs ultrasons
Ultrasonic ultrasonicF(8); //Init of ultrasonic sensor on pin 8
Ultrasonic ultrasonicG(10);//Init of ultrasonic sensor on pin 10
Ultrasonic ultrasonicD(12);//Init of ultrasonic sensor on pin 12
```

L'intérêt de ne pas mettre les constantes ou bien certaines variables ici, est le fait de pouvoir les appeler dans n'importe quelle fonction.
En effet, certaines variables sont utilisées dans plusieurs fonctions.

La première partie du programme est sûrement la plus importante, c'est elle qui permet de faire avancer le robot, elle répond à la première et troisième fonction technique de notre FAST.

Nous avons utilisé la fonction *void_loop()* (une fonction qui permet de répéter de manière infini son contenu), dans laquelle nous avons défini une lecture des capteurs ultrasons, de cette manière à chaque répétition de la boucle, nous vérifions la position du robot dans son environnement.

Pour avancer, nous avons utilisé plusieurs boucles *if.* Grâce à elles, nous pouvons corriger la trajectoire du robot s'il est trop à gauche ou trop à droite.

De plus, le parcours que le robot doit faire se coupe en 3 parties :

-La partie où le robot doit aller tout droit
-La partie où il doit aller à gauche
-La partie où il doit aller à droite

Nous avons donc créé une condition pour chaque état.

Lorsqu'il va tout droit, nous avons 3 conditions :

Celle pour aller tout droit

```
if (DistanceCapteurDroite > 75 && DistanceCapteurGauche > 75) { // Si le robot est au centre du couloir il va tout droit
  MoteurGD(VmoyG, VmoyD+2);
  millis(); // La fonction millis() permet l'execution rapide de chaque boucle if et donc une presque exécution simultané des ces mêmes boucles
  }
```

Nous savons que le mur du couloir du parcours fait environ 1m60, de ce fait, nous avons mis la distance des capteurs d'ultrasons à 75 cm à droite et à gauche, ainsi, quand il est approximativement au centre du couloir il a pour ordre d'aller droit devant lui.

Si le robot vient à aller trop à droite, nous avons alors une correction pour que la roue droite aille plus vite que la roue gauche, afin de se recentrer dans le couloir et inversement quand il sera à gauche.

```
if (DistanceCapteurDroite < 80){ // Si le robot est proche du mur droite il tourne légèrement à gauche
    MoteurGD(VmoyG, VmoyD+27);
    millis();
    }

if (DistanceCapteurGauche < 80){ // Si le robot est proche du mur gauche il tourne légèrement à droite
    MoteurGD(VmoyG+2, VmoyD);
    millis();
    }
```

On peut constater que la roue droite a besoin d'une plus grande valeur pour pouvoir se remettre vers le centre. Cela vient du fait que l'encodeur à gauche est plus puissant que celui de droite.
De plus, nous avons un problème avec la roue droite qui affaiblit sa puissance, donc nous devons compenser.
Pour connaître la bonne valeur appropriée, nous avons dû faire de l'asservissement afin de trouver une solution correcte.

Une fois que le robot est allé tout droit dans le premier couloir, il doit aller à gauche. Pour cela, nous avons regardé la distance du mur qui était en face à partir du moment où dans le couloir, le côté gauche augmente. De ce fait, si le robot est inférieur à une valeur que nous avons mesurée et que le capteur gauche est supérieur à une valeur mesurée, alors la roue droite tournera plus vite afin de tourner à gauche.

```
if (DistanceCapteurAvant < 329 && DistanceCapteurGauche > 90) {
  MoteurGD(VmoyG, VmoyD+25);
  millis();
 }
```

Ensuite, pour qu'il tourne à droite, nous reprenons le même procédé, seulement là nous regarderons le capteur gauche et le capteur droite et nous augmentons la roue gauche.

```
if (DistanceCapteurGauche < 110 && DistanceCapteurDroite > 100) {
 MoteurGD(VmoyG+60, VmoyD);
 millis();
 }
```

Une fois arrivé sur la ligne d'arrivée, il doit s'arrêter, nous avons mis toutes nos conditions de parcours dans une autre condition *if* qui va regarder si une variable booléenne est vrai ou fausse (cette variable s'appelle « arret » et est mise en valeur fausse de base.

```
boolean arret = false;
if(arret == false){
```

Pour constater le passage sur la ligne noir nous utilisons le capteur infrarouge, celui-ci basé sur une mesure de la réflexion d'onde lumineuse est alors inefficace en face d'une couleur noire qui rend impossible tout réflexion de lumière. Le robot traite donc l'information comme une distance avec le sol trop élevée pour son capteur. Nous pouvons alors utiliser cette information comme condition de déclenchement de notre boucle arrêt.

C'est ainsi que lorsque notre robot passera sur la ligne noir, il ira dans une condition qui est vraie si le capteur infrarouge est activé et dedans, nous changerons la valeur de la variable en vraie.

```
if (digitalRead(infrared)) { // Le robot s'arrête si on le soulève ou passe sur la ligne d'arrivée
  myrobot.setLcdRGB(0,255, 0);
  arret = true;
  Temps_stop_ms=millis();
  Vf_L = (myrobot.getEncoder("left")); //On récupère le nombre d'incrémentation de la roue gauche
  Vf_R = (myrobot.getEncoder("right")); //On récupère le nombre d'incrémentation de la roue droite

  MoteurGD(Stop, Stop); //Arret du robot
  Duree_ms= Temps_stop_ms - Temps_start_ms; //Calcul du temps du parcours
  affichage(); //Appel de la fonction affichage
  }
```

Une fois le robot arrêter, nous allons appeler la fonction affichage () qui est la grande deuxième partie du programme qui répond à la deuxième et troisième fonction technique de notre FAST

Cette partie permet d'afficher toutes les informations qui sont demandées selon le cahier des charges :

-Temps de parcours
-Energie Électrique Consommée
-Nombre de fois où le robot est <20 cm des murs
-Nombre de tours de la roue droite et gauche
-Distance parcourue

Pour les différents affichages, nous avons créé un switch case qui s'incrémentera avec le joystick que le robot possède. Lorsque le joystick sera poussé à droite dans l'axe des Y, alors il s'incrémentera de 1 et si l'on va à gauche, il se décrémentera.

```
while(1){
  if((myrobot.getJoystickY())<358){
    gps++;
    while((myrobot.getJoystickY())<358);
  }

  else if ((myrobot.getJoystickY())>671){
    gps--;
    while((myrobot.getJoystickY())>671);
  }
  switch(gps) {
    case 0 :
    myrobot.setLcdCursor(0, 0);
    myrobot.lcdPrint("Fin de course");
    myrobot.setLcdCursor(0, 1);
    myrobot.lcdPrint("                ");
    break;
```

Temps de parcours

Le temps de parcours a été plutôt facile, nous avons utilisé la fonction millis() qui, en plus de faire des interruptions, elle permet de calculer un temps. Nous avons donc calculé le temps au départ dans le void_setup (). Cette fonction se répète qu'une seule fois au début du programme)

```
Temps_start_ms=millis(); //Calcul le début de la course parcours
```

Ensuite, une fois le robot arrivé à la ligne d'arrivée, il reprend la valeur de millis() :

```
Temps_stop_ms=millis();
```

On fait par la suite, la soustraction des deux :

```
Duree_ms= Temps_stop_ms - Temps_start_ms; //Calcul du temps du parcours
```

Puis, pour le remettre en seconde, nous faisons la conversion :

```
Duree_s = Duree_ms/1000;
```

La partie dans le programme pour l'affichage est :

```
case 1 :
myrobot.setLcdCursor(0, 0);
myrobot.lcdPrint("Temps parcours        ");
myrobot.setLcdCursor(0, 1);
myrobot.lcdPrint(Duree_s);
myrobot.lcdPrint("  secondes            ");
break;
```

(Les autres affichages reprennent la même méthode pour afficher, seulement le texte change et la variable d'affichage)

Energie Électrique Consommée

Nous avons créé un programme qui calcul la consommation moyenne du robot sur différentes valeurs, grâce à cela, nous avons pu déterminer un courant moyen du robot qui est d'environ 490 mA.

On a calculé que la tension du robot est environ égale à 2 fois le courant : Vout=Imoy*2

De plus, on sait que la puissance P=U*I donc P=Vout*Imoy

Et l'énergie E est égale à P*t où t est le temps de parcours du robot, donc E= Vout*Imoy*t

```
Vout=Imoy*2;
Energie=Vout*Imoy*Duree_h;
```

Nombre de fois où le robot est <20 cm des murs

Pour calculer le nombre de fois où le robot est <20 cm des murs, nous avons ajouté une condition dans la fonction *void_loop()* qui, grâce aux ultrasons, regarde si le robot est à moins de 20 cm. Si le robot est < 20cm, alors une variable s'incrémente.

```
if (DistanceCapteurAvant < 20 || DistanceCapteurDroite < 20 || DistanceCapteurGauche < 20 ){
  millis();
  Nm++; //On incrémente le nombre de fois où le robot est <20cm du mur.
 }
```

Nombre de tours de la roue droite et gauche

Pour calculer la roue droite et gauche, on récupère le nombre d'incrémentations des roues grâce à la fonction *myrobot.getEncoder()* pour la roue droite et gauche. Une fois les valeurs récupérées, et les valeurs mesurées d'un tour pour la roue droite = 1180 incrémentations et = 1150 pour la roue gauche. De ce fait, nous avons divisé le nombre total d'incrémentations par les valeurs d'un tour pour avoir le nombre de tours total.

```
Vf_L = (myrobot.getEncoder("left")); //On récupère le nombre d'incrémentation de la roue gauche
Vf_R = (myrobot.getEncoder("right")); //On récupère le nombre d'incrémentation de la roue droite
long NbtourG = (Vf_L/1150); //Nb de tours roue gauche
long NbtourD = (Vf_R/1180); //Nb de tours roue droite
```

Distance parcourue

Pour calculer la distance parcourue, on a calculé la distance où le robot fait 1 tour de roue qui est égale à 20 cm. Ensuite, on a plus qu'à prendre le nombre de tours pour chaque roue, puis le multiplier par 0.1 pour mettre la valeur en mètres. Nous prenons 10 au lieu de 20 car on additionne le nombre de tour à droite et à gauche, donc si nous ne divisons pas par 2, on aura le double de distance réel.

```
float distance_Robot = (((Vf_R/1180) + (Vf_L/1150))*0.1);
```

Une fois la deuxième partie finie, il reste une grande partie qui est la sécurité du robot afin de protéger les composants. Celui-ci répond à la quatrième et sixième fonction technique de notre FAST.

Comme cité précédemment, selon le cahier des charges, les sécurités que le robot doit avoir sont :

-Arrêt si proximité trop importe avec un mur <5 cm (Arrêt Dist)

-Arrêt si contact avec le mur ou roues bloquées (Arrêt RB)

-Arrêt si l'on détecte un courant supérieur à la normal (Arrêt Elec)

-Arrêt si la vitesse du robot dépasse la limite fixer de 200°/s (Arrêt Mec)

Arrêt Dist

La protection arrêt distance permet de protéger le robot si jamais il détecte un obstacle à moins de 5 cm. Elle permet donc de protéger le robot pour qu'il puisse éviter de se casser en tapant dans un mur ou un autre obstacle.

Pour créer cette sécurité, on a rajouté une condition if dans le *void_loop()* qui va aller dans la fonction *Arret_Dist()* si un des 3 ultrasons du robot est inférieur à 5cm.

```
if (DistanceCapteurAvant < 5 || DistanceCapteurDroite < 5 || DistanceCapteurGauche < 5 ) {  // Le robot s'arrête s'il arrive face à un mur
    millis();
    Arret_Dist(); //On entre dans l'interruption Arret Distance
    }

void Arret_Dist(){
  MoteurGD(Stop, Stop); //Le robot s'arrête
  myrobot.lcdClear();
  myrobot.setLcdRGB(255,0,0); //Rouge
  myrobot.setLcdCursor(0, 0);
  myrobot.lcdPrint("  Arret Distance   ");
  while(not(myrobot.getJoystickClic())); //Attends un appuie sur le joystick
  myrobot.setLcdRGB(255, 255, 255);
}
```

On a rajouté également une boucle *while()* dans la fonction *Arret_Dist()* qui permet de ne pas à avoir redémarrer le programme pour qu'il se poursuive, il suffit d'appuyer sur le joystick après avoir déplacé le robot et il continue le programme.

Arrêt RB

La protection Arrêt RB permet d'arrêter le robot quand il détecte que ses roues ne tournent pas alors que la commande est envoyée. Cela permet d'éviter d'abîmer les moteurs des roues quand elles sont bloquées par quelque chose.

Pour créer cela, nous avons fait une condition pour la roue droite et gauche qui regarde la valeur de l'encodeur et nous avons regardé si à chaque fois, la nouvelle position des roues est différente de l'ancienne position. Si la position est différente, alors le robot continue d'avancer, mais si elle est identique à l'ancienne position, alors cela veut dire que le robot est bloqué et donc il faut arrêter le robot pour protéger le moteur.

On lit les valeurs :

```
newLeft = knobLeft.read(); //Lecture roue gauche
newRight = knobRight.read();//Lecture roue droite
```

Puis on regarde si c'est différent de l'ancienne valeur

```
//Si position actuel est differente de la nouvelle position gauche
if (newLeft != positionLeft) { //Si la position a bougé
  DifferentielGauche=newLeft-positionLeft;//Différence de position
  positionLeft = newLeft; //Nouvelle position
```

Arrêt Elec

La protection Arret_Elec permet d'arrêter le robot si jamais le courant dans un moteur est supérieur au courant nominal pendant un certain temps.

On sait que le courant nominal est 1.25A selon la datasheet.

La tension du robot se lit sur les entrées analogiques A3 et A4 pour le moteur droit et gauche.

Seulement, la valeur donnée est en analogique, nous sommes entre 0-1023 bits.
On a notre tension qui est entre 0-5V, il faut alors calculer la valeur qu'on va récupérer, pour cela, on peut prendre le max qui est 5V, on a 1023.
N représente la valeur analogique qu'on récupère, on sait que la résistance est de 100mOhm(datasheet). Donc N=1023/5*100*10^-3*I

Ce qui veut dire que I= N*5/1023/100*10^-3

On a donc ajouté dans le void_loop(), une mesure de la tension analogique, puis le calcul du courant et on a mis une condition qui, si le courant mesurée est supérieur à 1A (on prend une marge pour ne pas prendre de risques), alors il doit s'arrêter.

```
TensionAnalogique=(analogRead(A3)+analogRead(A4));
CourantMesure=(TensionAnalogique*5/1023/100*10^-3);
```

Puis on rajoute une condition qui vérifie si le courant mesuré est supérieur à 1, si oui, alors il va dans la fonction Arret_Elec et il s'arrête.

```
if(CourantMesure>1){
  millis();
  Arret_Elec();
}
```

```
void Arret_Elec() {
  MoteurGD(Stop, Stop);  //Le robot s'arrête
  myrobot.lcdClear();
  myrobot.setLcdRGB(255, 0, 0);  //Rouge
  myrobot.setLcdCursor(0, 0);
  myrobot.lcdPrint("  Arret Courant  ");
}
```

Arrêt Méca

La protection Arrêt Mec permet de protéger le robot si la vitesse de rotation est supérieure à 200 degrés par seconde.

Pour cela, nous avons dû calculer dans un premier temps, la valeur d'incrémentation lorsque le robot fait un tour grâce à la fonction encodeur.

Roue Gauche	Encodeur= 1150	Angle= 360°
Roue Droite	Encodeur= 1180	Angle= 360°

Maintenant qu'on connaît la valeur des encodeurs pour 360°, nous avons calculé la limite des encodeurs qui est de 200°, précisée par le cahier des charges.

Gauche	Encodeur= 639	Angle= 200°
Droite	Encodeur= 656	Angle= 200°

Dans notre programme, nous avons fait le calcul directement dans le *void_setup()* afin qu'on ait la valeur exacte et pas une valeur approchée.

```
LimEncSecDroite=((LimDegSec/360)*ProprieteEncDroite);//Ici on calcule la limite de l'encodeur droite
LimEncSecGauche=((LimDegSec/360)*ProprieteEncGauche);//Ici on calcule la limite de l'encodeur gauche
```

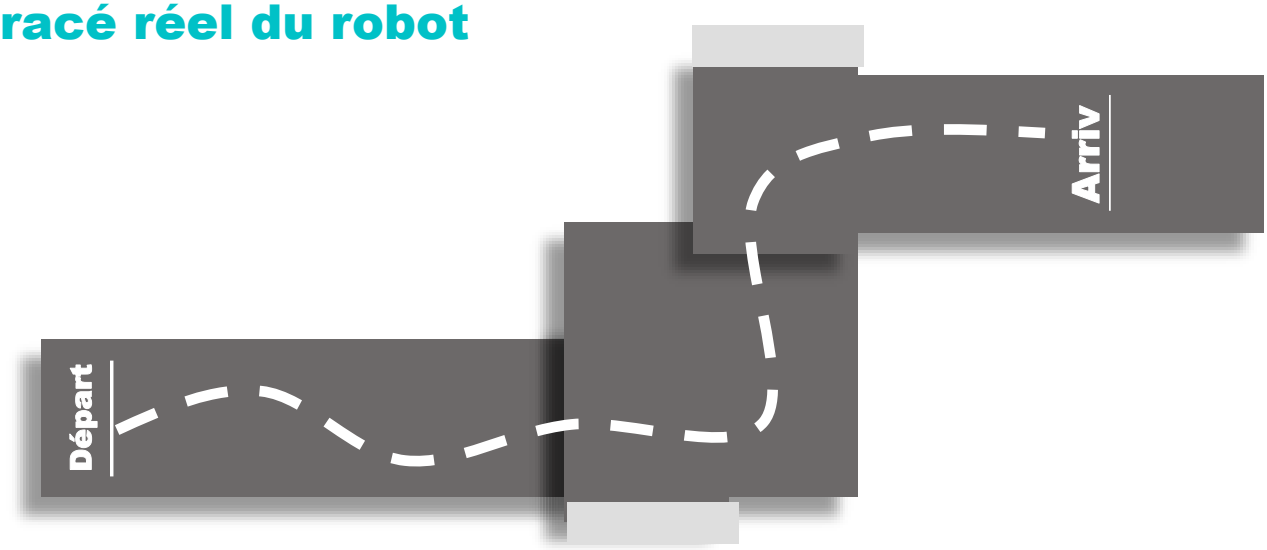
Ensuite, nous avons créé une variable qui va calculer la différence de position entre t et t-1.
Puis, on va écrire que si la différence de position est supérieure à la limite de l'encodeur, alors un ordre d'arrêt sera émis si la limite est dépassée.

```
//Si position actuel est differente de la nouvelle position gauche
if (newLeft != positionLeft) { //Si la position a bougé
  DifferentielGauche=newLeft-positionLeft;//Différence de position
  positionLeft = newLeft; //Nouvelle position

if (DifferentielGauche < LimEncSecGauche){ //Si la position est différente
  millis();
  myrobot.lcdPrint("Le robot avance");
}

else{//Sinon
  MoteurGD(400,400); //robot stop
  myrobot.setLcdRGB(255, 0, 0); //Affichage en couleur rouge
  myrobot.setLcdCursor(0, 0);
  myrobot.lcdPrint("Arret Mec Gauche"); //message d'erreur
  }

if (newRight != positionRight) { //Si la position a bougé
  DifferentielDroite=newRight-positionRight;//Différence de position
  positionLeft = newLeft; //Nouvelle position

/Si position actuel est differente de la nouvelle position droite
if (DifferentielDroite < LimEncSecDroite){ //Si la position est différente
  millis();
  myrobot.lcdPrint("Le robot avance");
}

else{ //Sinon
  MoteurGD(400,400); //robot stop
  myrobot.setLcdRGB(255, 0, 0); //Affichage en couleur rouge
  myrobot.setLcdCursor(0, 0);
  myrobot.lcdPrint("Arret Mec Droite"); //message d'erreur
}
}
```

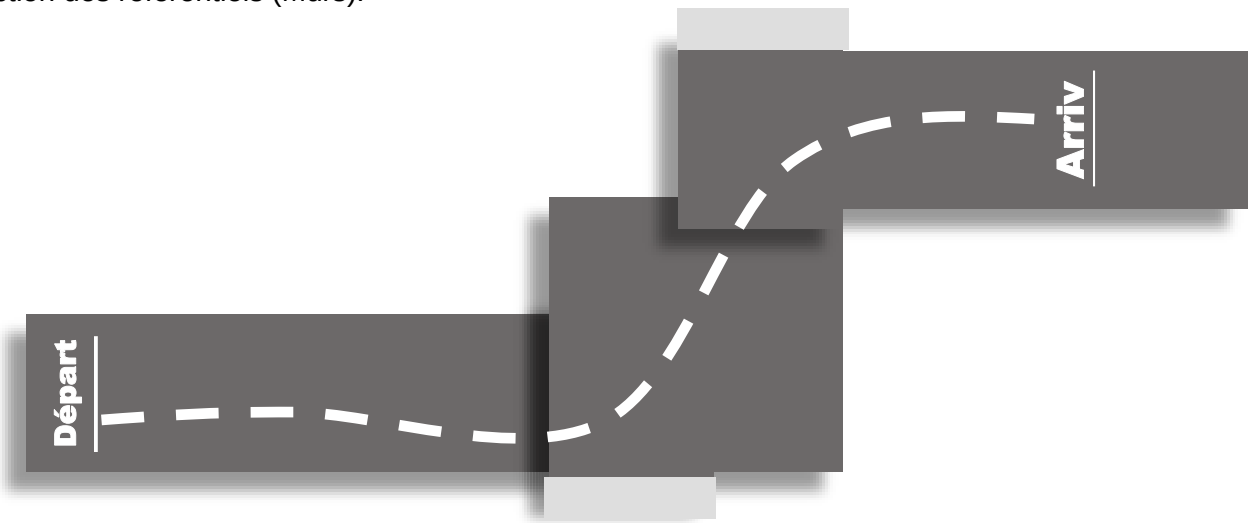
Ce programme est dans la fonction *void_loop()* afin qu'il analyse en continue l'angle.

Finalement, nous nous servons de l'arrêt RB pour faire fonctionner l'arrêt méca

Tracé réel du robot



Le tracé suivi par le robot est inévitablement différent de l'idée première imaginée pour le chemin, notamment parce que notre mode de fonctionnement se base sur une régulation permanente de la position en fonction des référentiels (Murs).

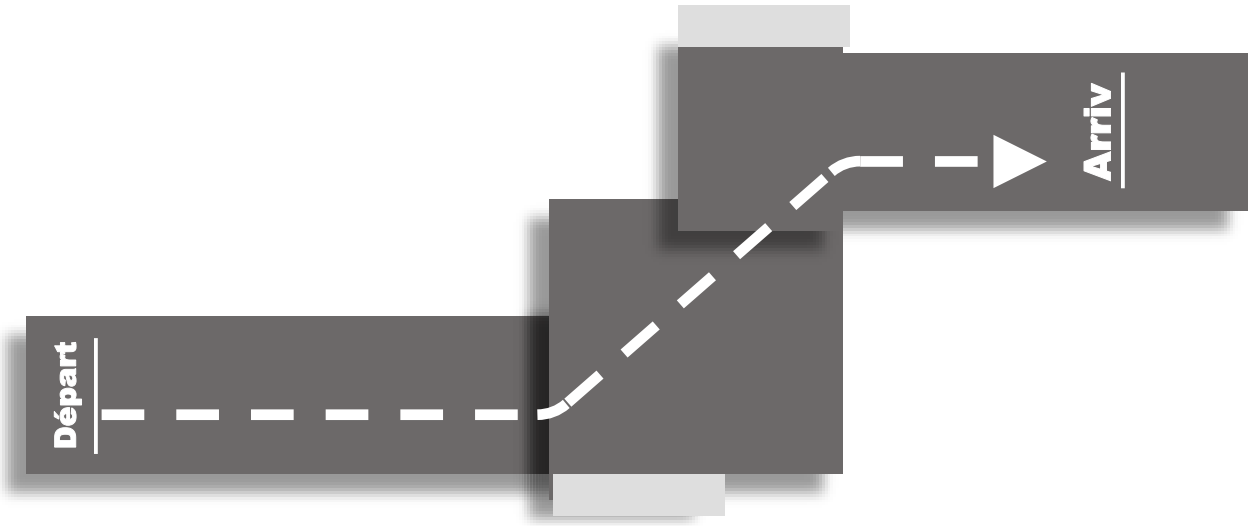


Finalement après plusieurs réglages sur les dernières heures du projet il a été possible d'améliorer significativement le déplacement du robot.

Analyse et Critique

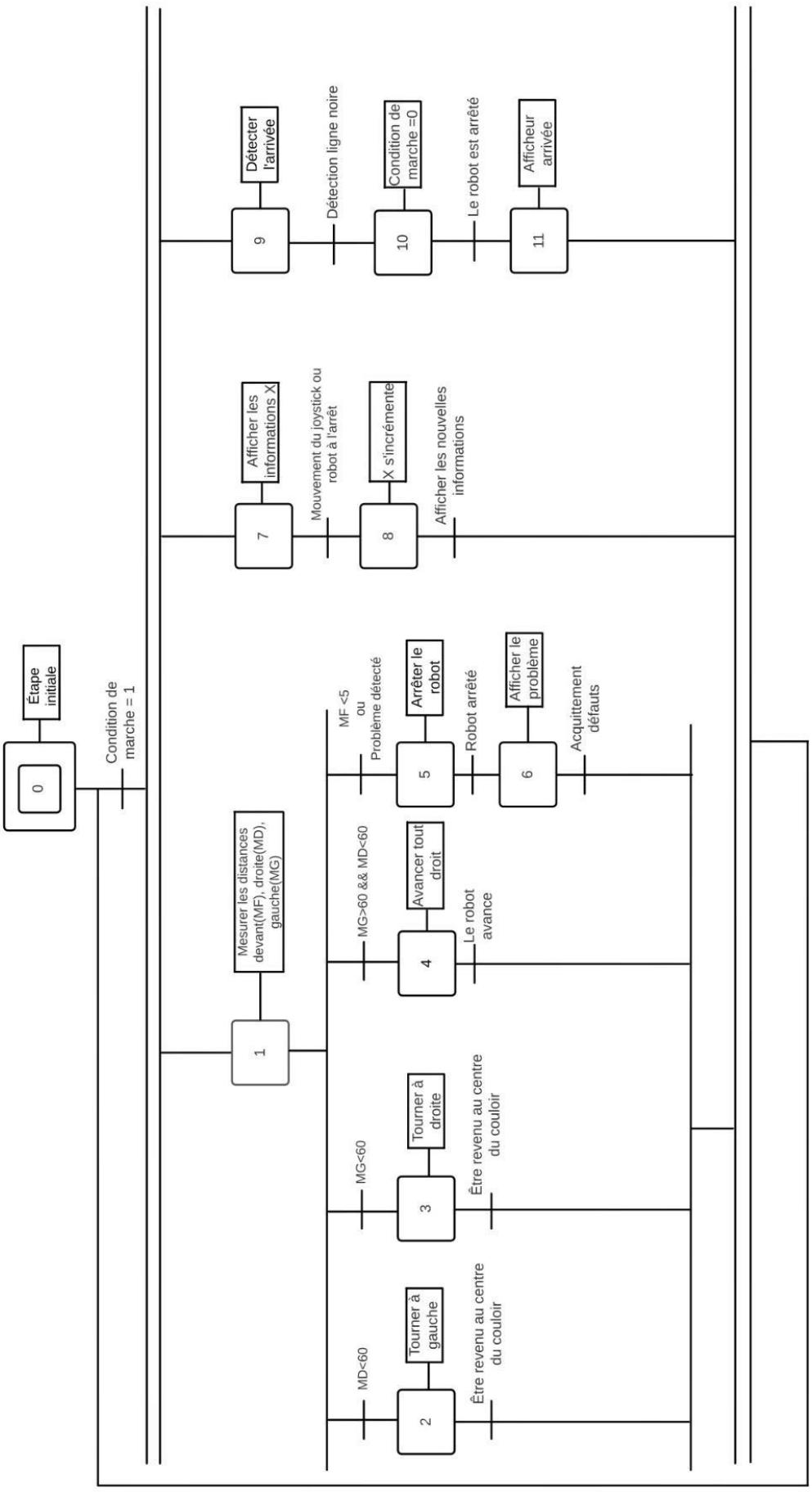
Lorsque l'on observe le trajet du robot la première critique qui se dégage est celle des mouvement « parasite », les écarts que le robot fait en ligne droite est une perte de temps qu'il faut corriger. Après analyse de notre programme nous avons pu constater que ces écarts étaient dû au temps parfois trop important d'exécution des boucles de régulation, en d'autres termes le robot met trop de temps à constater qu'il s'approche d'un mur et donc met tout autant de temps à savoir qu'il doit revenir au centre du couloir. Pour corriger cela nous avons cherché à simplifier les boucles, à réduire leur longueur afin de rendre leur lecture plus rapide. De plus nous avons découvert la fonction millis() qui sert à la base à calculer un temps mais c'est aussi une interruption. Nous avons constaté que si celle-ci était ajoutée en fin de boucle, la vitesse d'exécution de cette dernière était plus rapide.

Autres solutions



Le tracé ci-dessus correspond au trajet idéal que devrait suivre le robot, il s'agit du trajet le plus court et donc le plus rapide. Nous avons à un moment réalisé une tentative de suivre ce tracé, néanmoins après quelques échecs et un manque de temps nous avons fait le choix de la sécurité en somme retourné sur notre idée première qui elle, fonctionnait.

Grafcet


Étape initiale
0
Condition de marche = 1
1
Mesurer les distances devant(MF), droite(MD), gauche(MG)
MD<60
2
Tourner à gauche
Être revenu au centre du couloir
MG<60
3
Tourner à droite
Être revenu au centre du couloir
MG>60 && MD<60
4
Avancer tout droit
Le robot avance
MF <5 ou Problème détecté
5
Arrêter le robot
Robot arrêté
6
Afficher le problème
Acquittement défauts
7
Afficher les informations X
Mouvement du joystick ou robot à l'arrêt
8
X s'incrémente
Afficher les nouvelles informations
9
Détecter l'arrivée
Détection ligne noire
10
Condition de marche =0
Le robot est arrêté
11
Afficheur arrivée

>> Bilan et Analyse

Bilan Technique

Au terme de ce projet nous ne pouvons que constater l'imperfection de notre robot, en effet celui-ci même s'il réponde au cahier des charges n'est pas le résultat que l'on avait escompté, imaginé aux premières séances.

Le principal reproche à faire à notre travail est le déplacement en ligne droite du robot qui n'est pas parfait. Ce déplacement que l'on souhaitait rendre le plus droit possible est en fin de compte perturbé de variation, d'écart dû à un temps de réponse trop long du robot. Pour pallier à ce problème la solution serait passé par l'utilisation de TIMER, cependant ceux-ci entrent en conflit avec l'utilisation des fonctions *myrobot* issues de la bibliothèque <MARK>. Nous avons donc essayé de nous passer de cette dernière cependant par un mélange de manque de temps et de difficulté technique lié au langage de programmation, nous n'avons pas été en mesure de le faire.

Difficultés rencontrées

Au cours des nombreuses séances nous avons rencontré plusieurs contraintes.

Tout d'abord la « barrière de la langue », en effet comme évoqué plus haut dans le rapport seul deux des personnes de notre groupe était familier du langage C++, pour notre troisième camarade une découverte totale du langage de programmation a été handicapante et à nécessairement été un frein au lancement du groupe dans le projet.

Un autre frein qui se dégage du même ensemble de contraintes, est la prise en main de l'environnement de développement Arduino, sur les deux personnes connaissant le C++ seul l'un d'eux avait déjà manipuler l'environnement de développement UNO.

Pour entrer un peu plus dans le robot lui-même un problème auquel nous avons rapidement été confronté a été la gestion de la vitesse. Le premier constat qui nous a étonnés est de ne pas avoir un robot qui se déplace en ligne droite lorsque nous affectons des valeurs de vitesse identique aux deux roues.

Un autre obstacle qui nous est apparu compliqué a été l'utilisation des timers, malgré plusieurs explications, il nous a été difficile de les mettre en application dans notre programme. Cependant leur intérêt est important car ils nous permettraient d'augmenter l'efficacité d'utilisation des capteurs ultrasons et donc la réponse de régulation de déplacement en ligne droite du robot, problème que nous avons évoqué plus tôt dans ce rapport.

REX Technique et humain

Cet exercice de travail sur le robot MARK, pour l'ensemble de notre groupe a été l'occasion de travailler sur notre mode de fonctionnement en groupe projet scolaire. Nous avions tous pu expérimenter cela au cours de nos années lycée et IUT mais jamais sur un projet de cette envergure, étalé sur une si grande durée. Nous avons dû apprendre à nous adapter à l'ensemble des personnes du groupe, à répartir intelligemment les tâches, valoriser les points forts de chacun mais en laissant l'opportunité à toutes les parties de s'exercer sur l'ensemble des disciplines amenées par le projet.

Le langage C++, est probablement la compétence la plus centrale du travail sur le robot, elle est donc le meilleur exemple pour établir un constat de monté en compétence sur la période projet. Car même si l'un de nous n'avions pas tous la même connaissance du langage C, et de l'environnement de développement Arduino nous avons tenu à tous participer à l'élaboration du programme afin de travailler notre compétence dans le domaine.

>> Conclusion

Le projet Robot MARK, a été l'occasion de mettre à profit des connaissances pour un projet commun, d'apprendre à s'organiser dans un groupe en fonction des forces et faiblesses de chacun. Ce projet nous a permis d'apprendre, d'expérimenter, d'améliorer nos compétences en langage C.

Nous avons réussi à répondre au cahier des charges nous ayant été donné. Notre robot réalise le déplacement entre le départ et l'arrivée dans un temps correct et ce en suivant une trajectoire sans trop de perturbations, pas de mouvements parasites, pas de passage trop près des murs.

Cependant nous pouvons encore améliorer notre travail, il est encore possible de perfectionner le déplacement du robot, en suivant sans aucuns écart la trajectoire parfaite. Nous pourrions aussi travailler à augmenter la vitesse du robot, ce qui demanderait d'améliorer le temps de réponse des boucles de régulation de la trajectoire. En somme, notre travail, s'il est fonctionnel et convenable, peut bien sûr être amélioré.

>> Annexes

GitHub

Lien GitHub du projet : **https://github.com/MTK-RobotMark/Projet-RobotMark-MTK-CCSE-3-2022-2023**

GitHub est un site web et un service de cloud qui aide les développeurs à stocker et à gérer leur code, ainsi qu'à suivre et contrôler les modifications qui lui sont apportées.

Dans notre cas, on s'en sert pour pouvoir mettre tous les fichiers qui sont importants pour la compréhension de notre projet. De plus, nous y mettons les diverses versions de nos programmes pour le robot.

Il se découpe en 5 dossiers :

-Programme Arrêt Protection

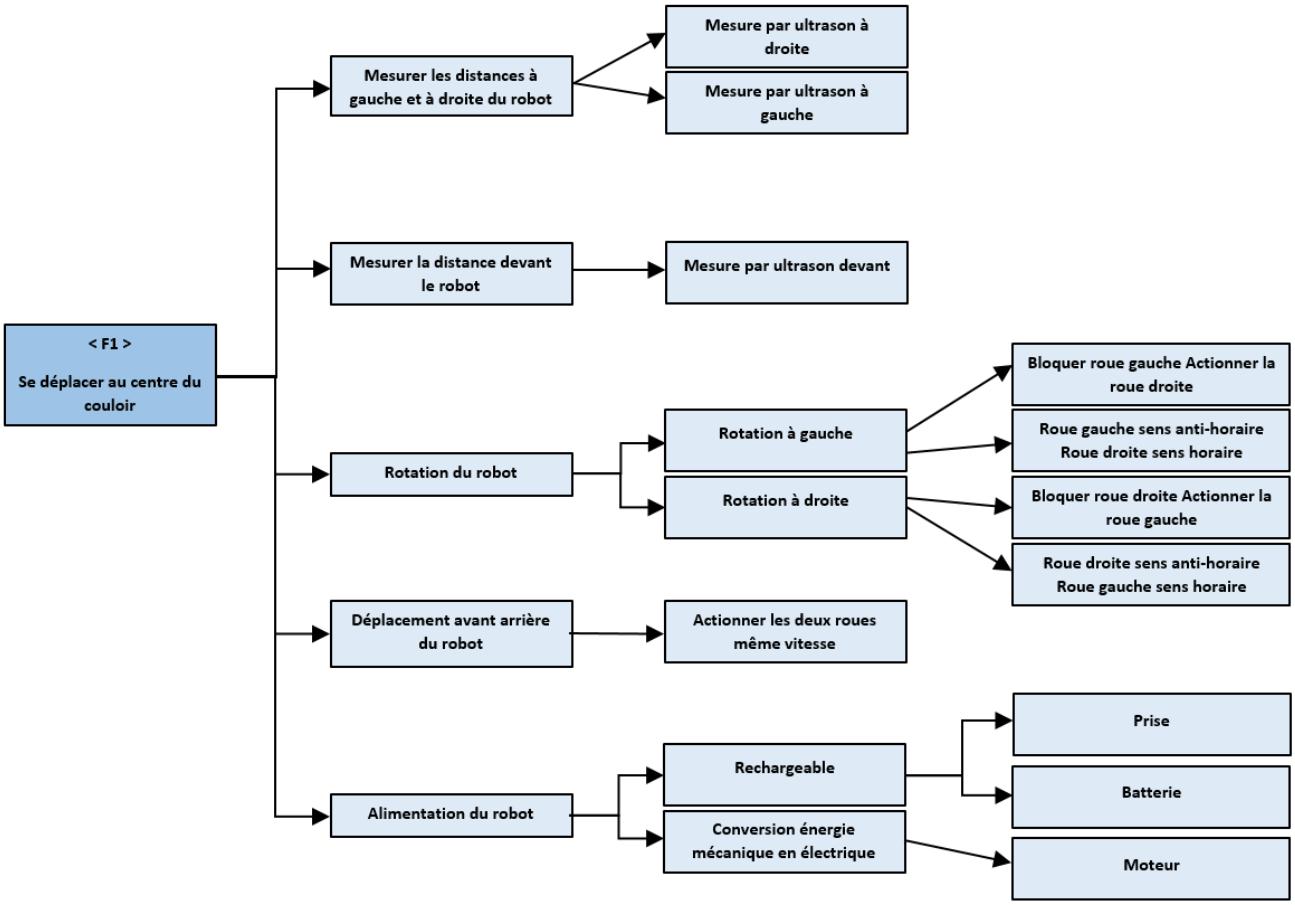
-Code Final

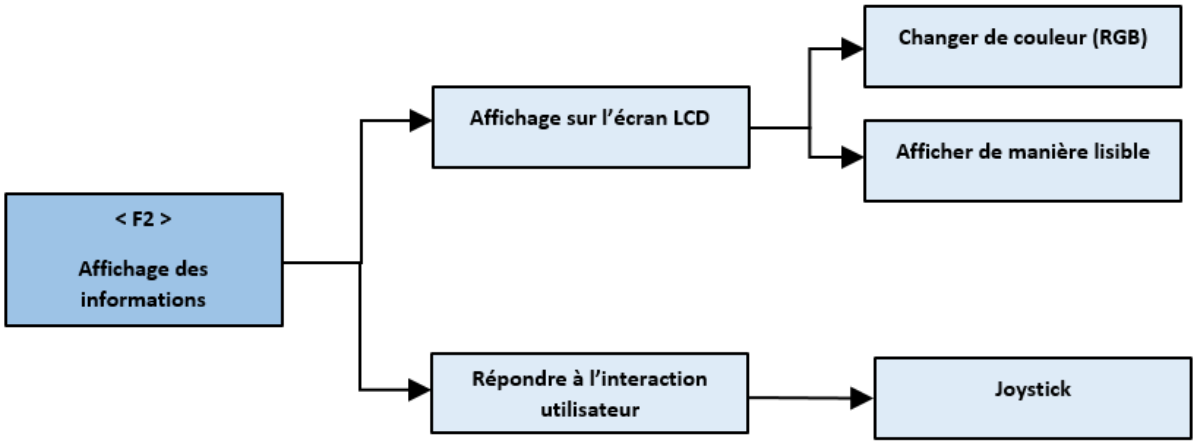
-Documentation

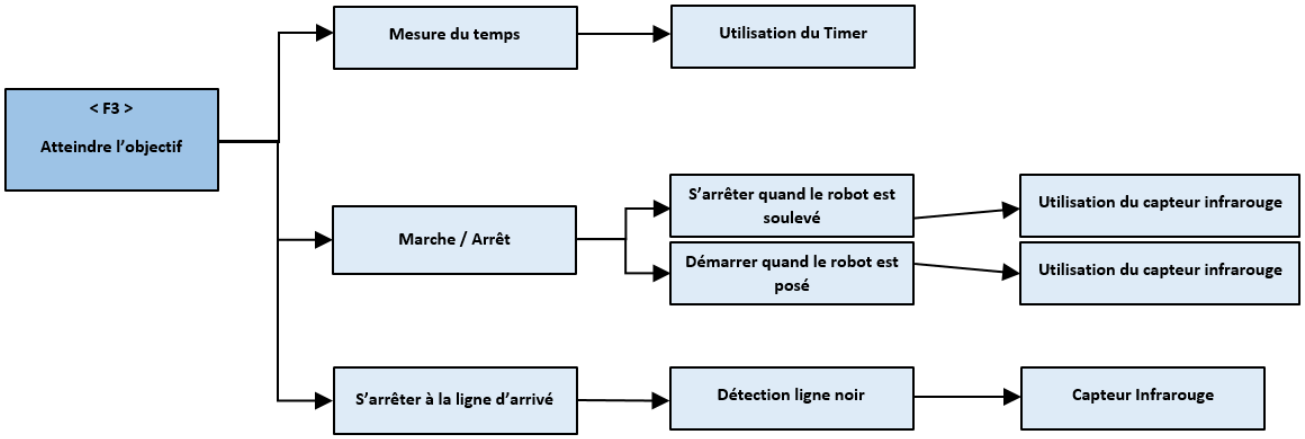
-Rapport & Présentation

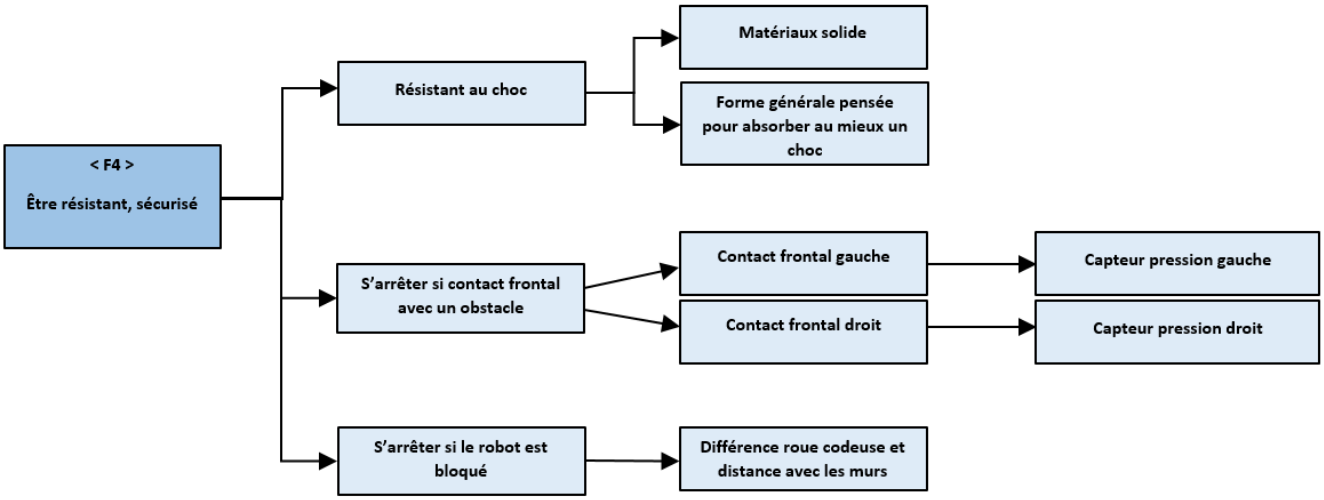
-Version Programme Robot Mark

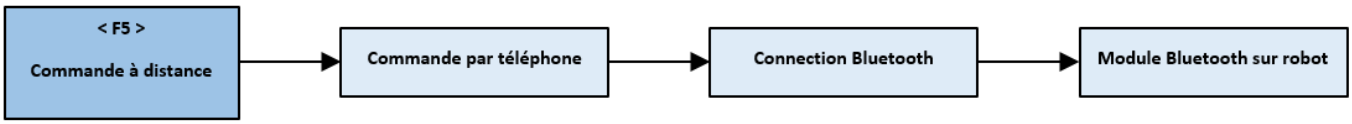
FAST (Fonctions Techniques)

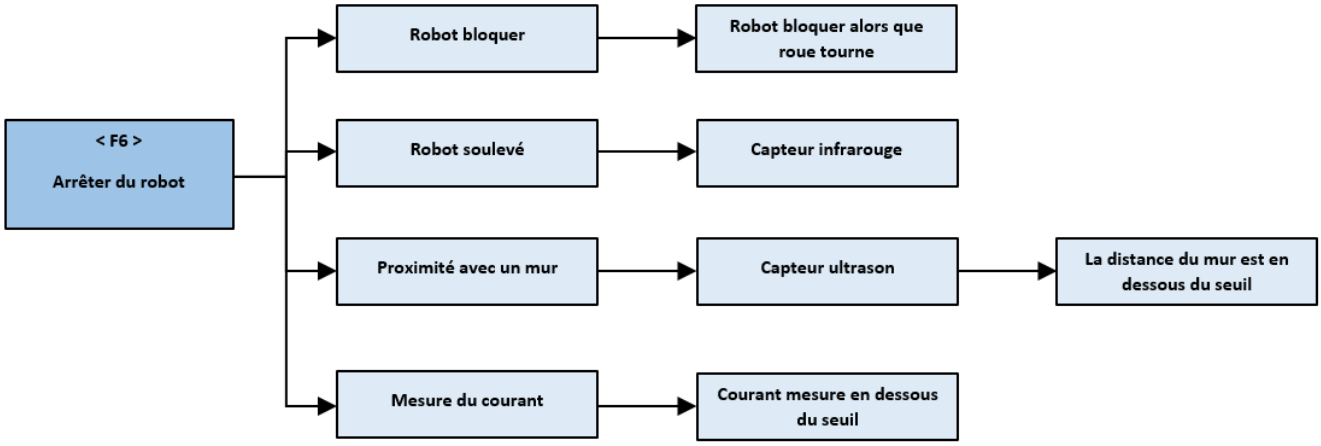





< F3 >
Atteindre l'objectif
Mesure du temps
Utilisation du Timer
Marche / Arrêt
S'arrêter quand le robot est soulevé
Démarrer quand le robot est posé
Utilisation du capteur infrarouge
Utilisation du capteur infrarouge
S'arrêter à la ligne d'arrivé
Détection ligne noir
Capteur Infrarouge


< F4 >
Être résistant, sécurisé
Résistant au choc
Matériaux solide
Forme générale pensée pour absorber au mieux un choc
S'arrêter si contact frontal avec un obstacle
Contact frontal gauche
Contact frontal droit
Capteur pression gauche
Capteur pression droit
S'arrêter si le robot est bloqué
Différence roue codeuse et distance avec les murs


< F5 >
Commande à distance
Commande par téléphone
Connection Bluetooth
Module Bluetooth sur robot


< F6 >
Arrêter du robot
Robot bloquer
Robot bloquer alors que roue tourne
Robot soulevé
Capteur infrarouge
Proximité avec un mur
Capteur ultrason
La distance du mur est en dessous du seuil
Mesure du courant
Courant mesure en dessous du seuil

Code complet

```
//Valeurs Prédéfinies
#define Thash 800
#define Stop 400
#define Vmax Thash
#define VmoyG 700
#define VmoyD 730
#define Angle90 420
#define Imoy 490
#define Inom 1


//Entrées analogiques
pinMode(A2, INPUT);
pinMode(A3, INPUT);

// Macros
#define LedToggle digitalWrite(13, !digitalRead(13))
#define MoteurG(Vg) OCR5A = Vg  // Vg in [0... 1999]
#define MoteurD(Vd) OCR5B = Vd  // VD in [0... 1999]
#define MoteurGD(Vg, Vd) \
  MoteurG(Vg); \
  MoteurD(Vd)
#define StopMoteurGD MoteurGD(Stop, Stop)

//Bibliothèques
#include <MARK.h>         //librairie du robot MARK
#include <Wire.h>         //lib for I2C connection
#include "Ultrasonic.h"   //Pour les Ultrasons
#include "rgb_lcd.h"      //Pour le LCD
#include <Encoder.h>      //Librairie des encodeurs
MARK myrobot;

//Déclaration des capteurs ultrasons
Ultrasonic ultrasonicF(8);   //Init of ultrasonic sensor on pin 8
Ultrasonic ultrasonicG(10);  //Init of ultrasonic sensor on pin 10
Ultrasonic ultrasonicD(12);  //Init of ultrasonic sensor on pin 12
```

```
//Encodeur
unsigned long LimEncSecDroite, LimEncSecGauche, ProprieteEncDroite, ProprieteEncGauche, LimDegSec;
Encoder knobLeft(18, 33);            //On définit le pin pour l'encodeur Gauche
Encoder knobRight(31, 19);           //On définit le pin pour l'encodeur Droite
unsigned long positionLeft = -999;   //On définit une position de base
unsigned long positionRight = -999;  //On définit une position de base
#define ProprieteEncDroite 1180      //Valeur de l'encodeur pour 1 tour de la roue droite
#define ProprieteEncGauche 1150      //Valeur de l'encodeur pour 1 tour de la roue gauche
#define LimDegSec 200                //Valeur limite
int DifferentielGauche;              //On mesure la différence de position t à t-1
int DifferentielDroite;              //On mesure la différence de position t à t-1

//Variables pour l'affichage des diverses informations
unsigned long Temps_start_ms;
unsigned long Temps_stop_ms;
unsigned long Duree_ms;
unsigned long Duree_s;
unsigned long Duree_h;
unsigned long SaveR;
unsigned long SaveL;
unsigned long Vf_L;
unsigned long Vf_R;
unsigned long Nm;
unsigned long courant, a;

//Variables pour les ultrasons
unsigned long DistanceCapteurAvant;
unsigned long DistanceCapteurDroite;
unsigned long DistanceCapteurGauche;

const byte infrared = 6;
boolean arret = false;
float CourantMesure;
float TensionAnalogique;
```

```
void initMoteurs() {  // MoteurG :OC5A=PIN46-PL3, MoteurD : OC5B=PIN45-PL4
  DDRL = 0x18;        // PL3 et PL4
  DDRB = 0x80;        // PB7 LedToggle
  // COM5B_1:0 = 10   -> clear sur egalite++, set sur egalite--
  // WGM5_3:1 = 1000  -> mode 8 => ICR5 defini le TOP
  TCCR5A = (1 << COM5A1) + (1 << COM5B1);
  TCCR5B = (1 << ICNC5) + (1 << WGM53) + (1 << CS50);  // CS_12:10  = 001  -> prediv par 1
  ICR5 = Thash;                                        // 1999 correspond a f = 4khz
  StopMoteurGD;
  // Interrution de débordement du timer
  TIMSK5 = 1 << TOIE5;
}

ISR(TIMER5_OVF_vect) {  // Pour la lecture du courant
  LedToggle;
}

void affichage() {
  int gps = 0;
  int Energie;
  int Vout;
  Duree_s = Duree_ms / 1000;
  Duree_h = Duree_s / 60 / 24;

  float distance_Robot = (((Vf_R / 1180) + (Vf_L / 1150)) * 0.1);
  long NbtourG = (Vf_L / 1150);  //Nb de tours roue gauche
  long NbtourD = (Vf_R / 1180);  //Nb de tours roue droite
  Vout = Imoy * 2;
  Energie = Vout * Imoy * Duree_h;

  while (1) {
    if ((myrobot.getJoystickY()) < 358) {
      gps++;
      while ((myrobot.getJoystickY()) < 358)
        ;
    }

    else if ((myrobot.getJoystickY()) > 671) {
      gps--;
      while ((myrobot.getJoystickY()) > 671)
        ;
    }
```

```
switch (gps) {
  case 0:
    myrobot.setLcdCursor(0, 0);
    myrobot.lcdPrint("Fin de course");
    myrobot.setLcdCursor(0, 1);
    myrobot.lcdPrint("                ");
    break;


  case 1:
    myrobot.setLcdCursor(0, 0);
    myrobot.lcdPrint("Temps parcours        ");
    myrobot.setLcdCursor(0, 1);
    myrobot.lcdPrint(Duree_s);
    myrobot.lcdPrint("  secondes            ");
    break;

  case 2:
    myrobot.setLcdCursor(0, 0);
    myrobot.lcdPrint(" Ener Elec Conso       ");
    myrobot.setLcdCursor(0, 1);
    myrobot.lcdPrint(Energie);
    break;

  case 3:
    myrobot.setLcdCursor(0, 0);
    myrobot.lcdPrint("le nombre Nm      ");
    myrobot.setLcdCursor(0, 1);
    myrobot.lcdPrint(Nm);
    myrobot.lcdPrint("     fois         ");
    break;

  case 4:
    myrobot.setLcdCursor(0, 0);
    myrobot.lcdPrint("Nb tour roue G    ");
    myrobot.setLcdCursor(0, 1);
    myrobot.lcdPrint(NbtourG);
    myrobot.lcdPrint("   tours          ");
    break;
```

```
      case 5:
        myrobot.setLcdCursor(0, 0);
        myrobot.lcdPrint("Nb tour roue R    ");
        myrobot.setLcdCursor(0, 1);
        myrobot.lcdPrint(NbtourD);
        myrobot.lcdPrint("   tours          ");
        break;

      case 6:
        myrobot.setLcdCursor(0, 0);
        myrobot.lcdPrint("    Distance    ");
        myrobot.setLcdCursor(0, 1);
        myrobot.lcdPrint(distance_Robot);
        myrobot.lcdPrint("   m            ");
        break;

      default:
        myrobot.setLcdCursor(0, 0);
        myrobot.lcdPrint(" Retournez en arrière   ");
        myrobot.setLcdCursor(0, 1);
        myrobot.lcdPrint("                        ");
    }
  }
}

//Interruption Arrêt DISTANCE
void Arret_Dist() {
  MoteurGD(Stop, Stop);  //Le robot s'arrête
  myrobot.lcdClear();
  myrobot.setLcdRGB(255, 0, 0);  //Rouge
  myrobot.setLcdCursor(0, 0);
  myrobot.lcdPrint("  Arret Distance   ");
  while (not(myrobot.getJoystickClic()))
    ;  //Attends un appuie sur le joystick
  myrobot.setLcdRGB(255, 255, 255);
}

void Arret_Elec() {
  MoteurGD(Stop, Stop);  //Le robot s'arrête
  myrobot.lcdClear();
  myrobot.setLcdRGB(255, 0, 0);  //Rouge
  myrobot.setLcdCursor(0, 0);
  myrobot.lcdPrint("  Arret Courant  ");
}
```

```
void setup() {
  myrobot.begin();
  pinMode(43, OUTPUT);
  digitalWrite(43, 0);
  initMoteurs();  //Appel de la fonction initMoteurs
  sei();          //On autorise les interruptions
  digitalWrite(43, 1);

  LimEncSecDroite = ((LimDegSec / 360) * ProprieteEncDroite);  //Ici on calcule la limite de l'encodeur droite
  LimEncSecGauche = ((LimDegSec / 360) * ProprieteEncGauche);  //Ici on calcule la limite de l'encodeur gauche

  myrobot.setLedBarLevel(myrobot.getBatteryLevel());

  while (myrobot.getInfrared() || (not(myrobot.getJoystickClic()))) {
    MoteurGD(Stop, Stop);
  }
  delay(1000);
  myrobot.setLcdRGB(0, 255, 0);
  myrobot.setLcdCursor(0, 0);
  myrobot.lcdPrint("Appuyez pour depart");
  myrobot.setLcdRGB(0, 0, 255);

  myrobot.lcdClear();
  Temps_start_ms = millis();  //Calcul le début de la course parcours
}


void loop() {
  DistanceCapteurAvant = ultrasonicF.MeasureInCentimeters();
  DistanceCapteurGauche = ultrasonicG.MeasureInCentimeters();
  DistanceCapteurDroite = ultrasonicD.MeasureInCentimeters();
  newLeft = knobLeft.read();    //Lecture roue gauche
  newRight = knobRight.read();  //Lecture roue droite
  TensionAnalogique = (analogRead(A3) + analogRead(A4));
  CourantMesure = (TensionAnalogique * 5 / 1023 / 100 * 10 ^ -3);
  int angle = 0;

  if (arret == false) {
    millis();

    if (DistanceCapteurDroite > 80 && DistanceCapteurGauche > 80) {  // Si le robot est au centre du couloir il va tout droit
      MoteurGD(VmoyG, VmoyD + 2);
      millis();  // La fonction millis() permet l'execution rapide de chaque boucle if et donc une presque exécution simultané des ces mêmes boucles
    }
```

```
if (DistanceCapteurDroite < 80) {  // Si le robot est proche du mur droite il tourne légèrement à gauche
  MoteurGD(VmoyG, VmoyD + 27);
  millis();
}

if (DistanceCapteurGauche < 80) {  // Si le robot est proche du mur gauche il tourne légèrement à droite
  MoteurGD(VmoyG + 2, VmoyD);
  millis();
}

if (DistanceCapteurAvant < 329 && DistanceCapteurGauche > 90) {
  MoteurGD(VmoyG, VmoyD + 25);
  millis();
}

if (DistanceCapteurGauche < 110 && DistanceCapteurDroite > 100) {
  MoteurGD(VmoyG + 60, VmoyD);
  millis();
}

if (digitalRead(infrared)) {  // Le robot s'arrête si on le soulève ou passe sur la ligne d'arrivée
  myrobot.setLcdRGB(0, 255, 0);
  arret = true;
  Temps_stop_ms = millis();
  Vf_L = (myrobot.getEncoder("left"));   //On récupère le nombre d'incrémentation de la roue gauche
  Vf_R = (myrobot.getEncoder("right"));  //On récupère le nombre d'incrémentation de la roue droite

  MoteurGD(Stop, Stop);                      //Arret du robot
  Duree_ms = Temps_stop_ms - Temps_start_ms;  //Calcul du temps du parcours
  affichage();                               //Appel de la fonction affichage
}

if (DistanceCapteurAvant < 20 || DistanceCapteurDroite < 20 || DistanceCapteurGauche < 20) {
  millis();
  Nm++;  //On incrémente le nombre de fois où le robot est <20cm du mur.
}


if (DistanceCapteurAvant < 5 || DistanceCapteurDroite < 5 || DistanceCapteurGauche < 5) {  // Le robot s'arrête s'il arrive face à un mur
  millis();
  Arret_Dist();  //On entre dans l'interruption Arret Distance
}
```

```
    //-----------------Interruption Distance Méca et RB-----------------//

    //Si position actuel est differente de la nouvelle position gauche
    if (newLeft != positionLeft) {                  //Si la position a bougé
      DifferentielGauche = newLeft - positionLeft;  //Différence de position
      positionLeft = newLeft;                       //Nouvelle position

      if (DifferentielGauche < LimEncSecGauche) {  //Si la position est différente
        millis();
        myrobot.lcdPrint("Le robot avance");
      }
    } else {                          //Sinon
      MoteurGD(400, 400);             //robot stop
      myrobot.setLcdRGB(255, 0, 0);  //Affichage en couleur rouge
      myrobot.setLcdCursor(0, 0);
      myrobot.lcdPrint("Arret Gauche");  //message d'erreur
    }

    if (newRight != positionRight) {                  //Si la position a bougé
      DifferentielDroite = newRight - positionRight;  //Différence de position
      positionLeft = newLeft;                         //Nouvelle position

      //Si position actuel est differente de la nouvelle position droite
      if (DifferentielDroite < LimEncSecDroite) {  //Si la position est différente
        millis();
        myrobot.lcdPrint("Le robot avance");
      }

      else {                            //Sinon
        MoteurGD(400, 400);             //robot stop
        myrobot.setLcdRGB(255, 0, 0);  //Affichage en couleur rouge
        myrobot.setLcdCursor(0, 0);
        myrobot.lcdPrint("Arret Droite");  //message d'erreur
      }
    }
    //---------------Arrêt Elec-------------//
    if (CourantMesure > 1) {
      millis();
      Arret_Elec();
    }
  }
}
```