Robot MARK

Compte Rendu Intermédiaire

KERVICHE BALDE BRITEL
CCSE3

Année 2022 - 2023

POLYTECH NANTES

Pôle Sciences et technologie
Nantes Université

Présentation 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

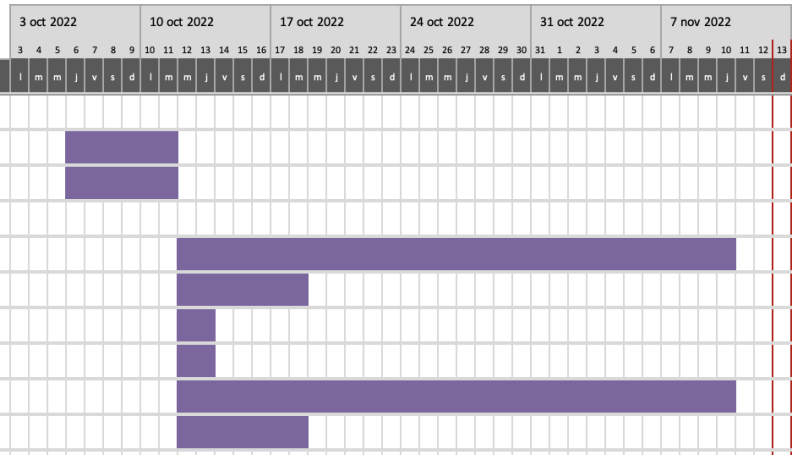
1) Planning

RobotMark

CCSE3
Projet

Début du projet : jeu, 10/6/2022
Semaine d'affichage : 1

TÂCHE	ATTRIBUÉE À	DÉBUT	FIN
Phase 1 : Découverte du projet			
Découverte du projet	KERVICHE, BRITEL, BALDE	6/10/22	11/10/22
Découverte du Github	KERVICHE, BRITEL, BALDE	6/10/22	11/10/22
Phase 2 : Livrable et Test			
Livrable	KERVICHE, BRITEL, BALDE	12/10/22	10/11/22
Analyse Fonctionnelle (FAST)	KERVICHE	12/10/22	18/10/22
Grafcet	BRITEL	12/10/22	13/10/22
Diagramme GANTT	BALDE	12/10/22	13/10/22
Programmation du robot	KERVICHE, BRITEL, BALDE	12/10/22	10/11/22
Procès-verbal	KERVICHE, BALDE	12/10/22	18/10/22



2) Livrable des Caractéristiques

Capteurs, Moteur et Afficheur

Télémètre ultrason Grove (capteur)

Les 3 capteurs permettent de mesurer la distance devant, à droite et à gauche du robot (sans contact) et donc le situer dans un plan 2D.

Alimentation :	Consommation :	Fréquence :	Portée de détection :	Résolution :	Dimensions :
5Vcc	15mA	40kHz	3cm-4m	1cm	43*25*15mm

Capteur de réflectance infrarouge Grove (capteur)

Ce capteur permet de mesurer la distance sous le robot et de détecter la ligne d'arrivée (ligne noire).

Tension de fonctionnement :	Courant de fonctionnement :	Portée effective :	Profondeur minimum détectable :	Temps de réponse :	Phototransistor :	LED infrarouge :
3,3-5 V	14,69 à 15,35 mA	4 à 15 mm	10mm	10 µs	800nm	940nm

Accéléromètre et gyroscope Grove (capteur)

Ce capteur nous permet de mesurer la vitesse à laquelle le robot va et le sens du robot.

Tension de d'alimentation analogique :	Consommation :	Plage de mesure de l'accélération linéaire :	Plage de mesure de la vitesse angulaire :	Temps de réponse :	Phototransistor :	LED infrarouge :
5 V/3,3 Vcc	0,9mA en mode Combo normal 1,25mA en mode Combo haute-performance	±2/±4/±8/±16 g	±125, ±245, ±500, ±1000, ±2000 dps	10 µs	800nm	940nm

Micro-interrupteurs (capteur)

Ces 2 micro-interrupteurs permettent de détecter quand le robot rencontre un obstacle en face avant.

Type d'interrupteurs :	Longueur du bras de levier :	Dimensions :
SPDT	16,3 mm	20*6,4*10,2mm

Servomoteur (moteur)

Le servomoteur permet la rotation du capteur ultrasons frontal.

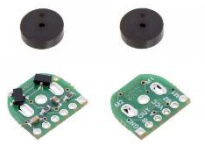
Dimensions :	Vitesse de fonctionnement :	Couple :	Tension de fonctionnement :	Système de contrôle :	Direction :
40,8*20,1*38 mm	0.18 sec/60° (4,8 V) / 0.16 sec/60° (6 V)	9 kg.cm/125.2 oz.in (4.8 V) / 10.2 kg.cm/141.9 oz.in (6 V)	4,8 à 6 V	Analogique	CCW

Micro-moteur (moteur)

Ces micro-moteurs permettent d'actionner les roues et d'en induire une vitesse.

Dimensions :	Vitesse à vide :	Courant à vide :	Courant de décrochage :	Couple de décrochage :	Puissance de sortie maximale :	Type de moteur :
10*12*26 mm	73 tr/min	0,07 Z	0.67 A	2,4 kg/cm	0,44 W	MP 6V

Paire d'encodeurs magnétiques pour micro-moteurs Pololu (capteur)



La paire d’encodeurs magnétiques sert à compter le nombre de tours des roues.

Tension de fonctionnement minimale :	Tension de fonctionnement maximale :	Compatible :	Dimensions :
2.7V	18V	Polulu HPCB	10.6*11.6 mm

Joystick Grove (contrôleur)

Le Joystick permet de se déplacer dans l’afficheur LCD.

Alimentation :	Dimensions :	Température de service :
3,3V et 5V	40*33*20mm	0°C à 40°C

Afficheur LCD RGB Grove (Afficheur)

L’afficheur LCD RGB permet d’afficher des informations sur le robot.

Alimentation :	Consommation :	CGROM :	CGRAM :	Affichage LCD :	Dimensions :
5V	<60 mA	10880 bits	64*8 bits	2 lignes de 16 caractères	84*45*13 mm

3) Procès Verbal

Composants Robot MARK

« Les programmes de tests sont disponibles en Annexe »

Capteur Ultrasonique - Frontal

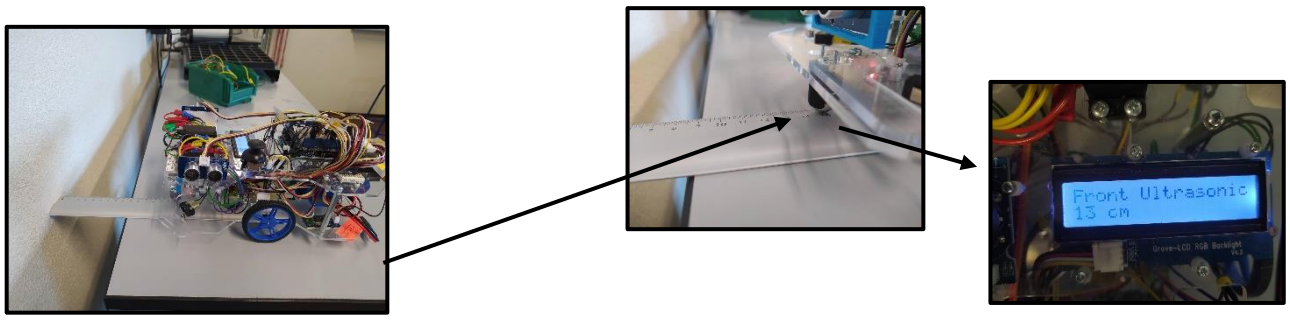
Fonction attendue :

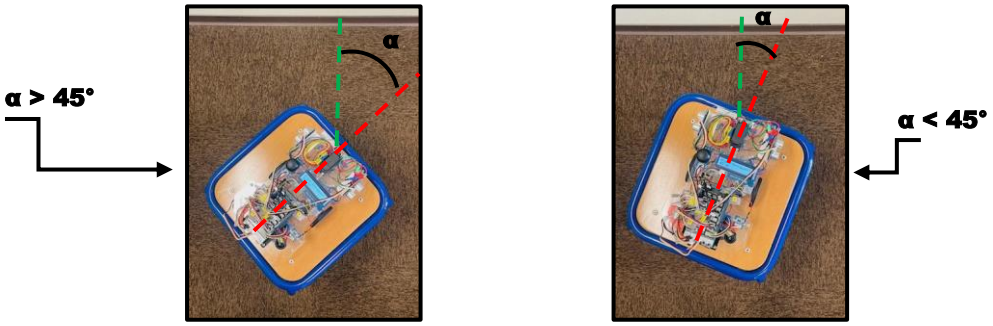
Mesure permanente de la distance entre la face avant du robot et l'obstacle le plus proche.

Indication datasheet :

→ Le capteur fonctionne sur une distance allant de 3 cm à 4m.
→ Le capteur fonctionne avec une résolution de 1 cm
→ Pas indication quand la capacité d'angle de réfection

Réalisation des essais : *(Photos)*





Mesure Impossible **Mesure Possible**

Constat :

→ En plaçant le robot à une distance de 14 cm du mur, on lit une distance mesurée de 15cm. Sachant que nous avons une résolution du capteur de 1 cm, nous pouvons affirmer le bon fonctionnement du capteur.

→ En réalisant plusieurs essais de mesure avec différent angles de réflexion on constate que le capteur fonctionnement approximativement pour un angle de réflexion < 45°

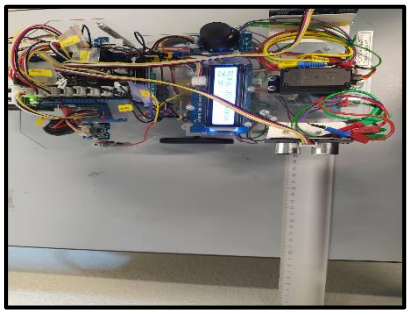
→ Nous avons rencontré sur les mesures de très longue distance (> 2m) des problèmes de mesure. Après réflexion nous avons admis que ces problèmes étaient dû à une légère inclinaison du capteur ce qui perturbe la mesure.

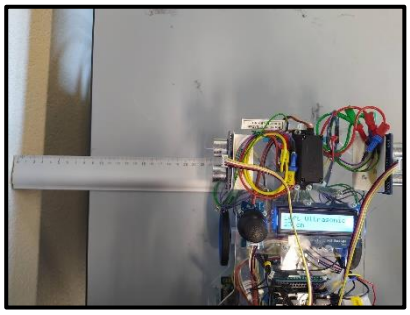
Capteur Ultrasonique - Latéral

Fonction attendue :

Mesure permanente de la distance entre la face (Gauche ou Droite) du robot et l'obstacle le plus proche.

Réalisation des essais : *(Photos)*





Constat :

→ En plaçant le robot à une distance de 23 et 25 cm du mur respectivement pour les capteur gauche et droit, on lit une distance mesurée de 23 et 25cm. Nous pouvons affirmer le bon fonctionnement des capteurs.

Capteur Infrarouge - Ventral

Fonction attendue :

Être capable de savoir si le robot est posé sur le sol ou s'il est porté.

Indication datasheet :

→ Portée effective allant de 4 à 15 mm

Constat :

→ Après avoir programmer un changement de couleur de l'écran LCD en fonction de la distance mesurée par le capteur, on constate la bonne exécution de ce programme et donc le bon fonctionnement du capteur.

Capteur accéléromètre

Fonction attendue :

Mesure de l'accélération sur les axes x, y et z. Mesure de la vitesse de rotation sur les axes x, y et z

Constat :

Pour la réalisation des tests nous avons utilisé une interface graphique sur l'outil développement d'Arduino. On trouvera vitesse, ou bien l'accélération à travers un delta du graphique **(Val_{max}, Val_{min}).**

Moteur roues - Latéral

Fonction attendue :

Rotation, de manière synchrone pour avancer, de manière désynchronisée pour effectuer une rotation.

Constat :

A l'aide de la fonction **MoteurGD(x1, x2)**, avec **x1=x2**, on réaliser une marche avant du robot, avec cette même fonction on obtient une rotation du robot.

Si	**x1 < x2**	Alors	**Rotation vers la Gauche**
Si	**x1 > x2**	Alors	**Rotation vers la Droite**

On constate donc le bon fonctionnent des moteurs de roue

Moteur servomoteur

Fonction attendue :

Permet la rotation du capteur Ultrason Frontal, sur un angle de 130°

Constat :

Bon fonctionnement de la rotation, mouvement fluide

Afficheur – Écran LCD

Fonction attendue :

Affichage texte et valeur. Paramétrage couleur RGB

Constat :

→ Au travers des tests du **capteur Infrarouge** nous avons constaté le bon fonctionnement du **LCD** concernant le paramétrage **couleur**.

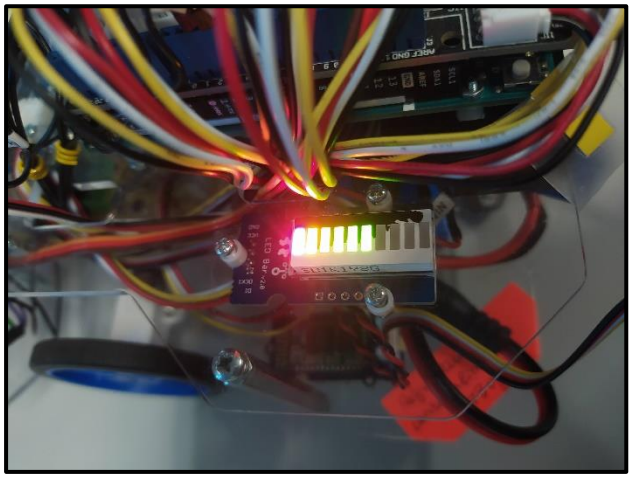
→ Au travers des tests des **capteurs Ultrasons** nous avons constaté le bon fonctionnement du **LCD** concernant l'affichage de **textes** et de **valeurs**.

Afficheur – LED Barre

Fonction attendue :

Affichage de données sous la forme d'une barre de LED, pouvant individuellement prendre plusieurs couleurs.

Réalisation des essais : *(Photos)*



Constat :

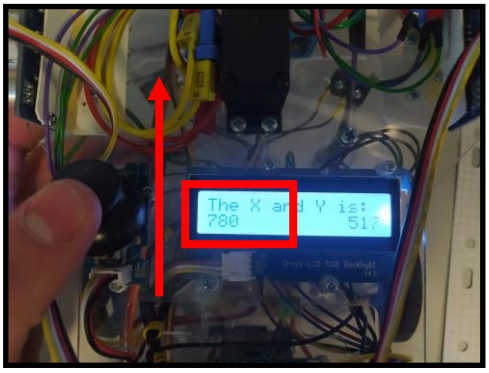
→ Nous avons constaté le bon fonctionnement individuel des LED, ainsi que leur capacité à changer de couleurs.

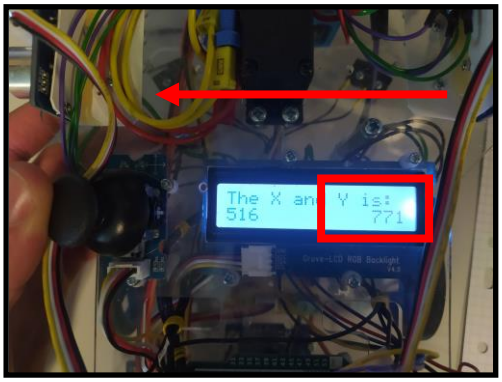
Contrôleur – Joystick

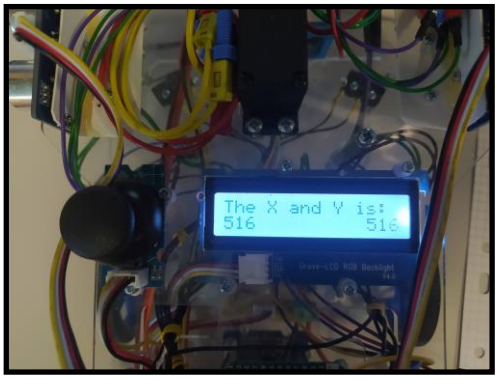
Fonction attendue :

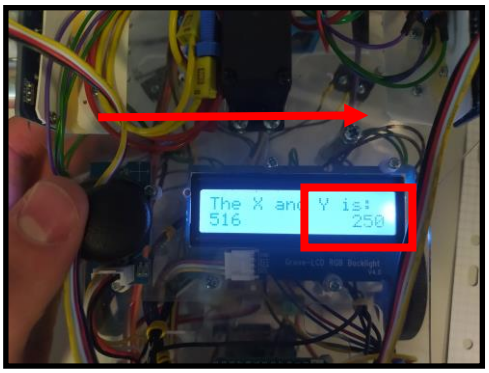
Permettre un contrôle de l'utilisateur directement sur le robot.

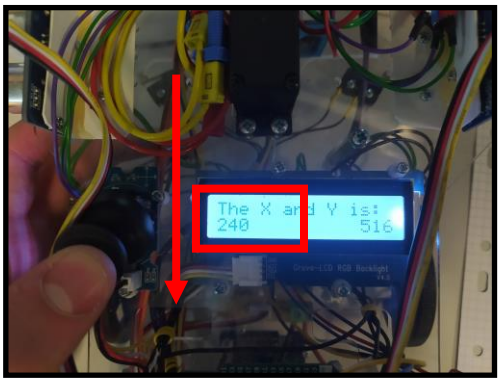
Réalisation des essais : *(Photos)*











Constat :

⟶ Nous pouvons constater à l'aide d'un affichage sur le LCD que le robot comprend correctement les informations transmises par le Joystick.

4) Liste des Fonctions Techniques

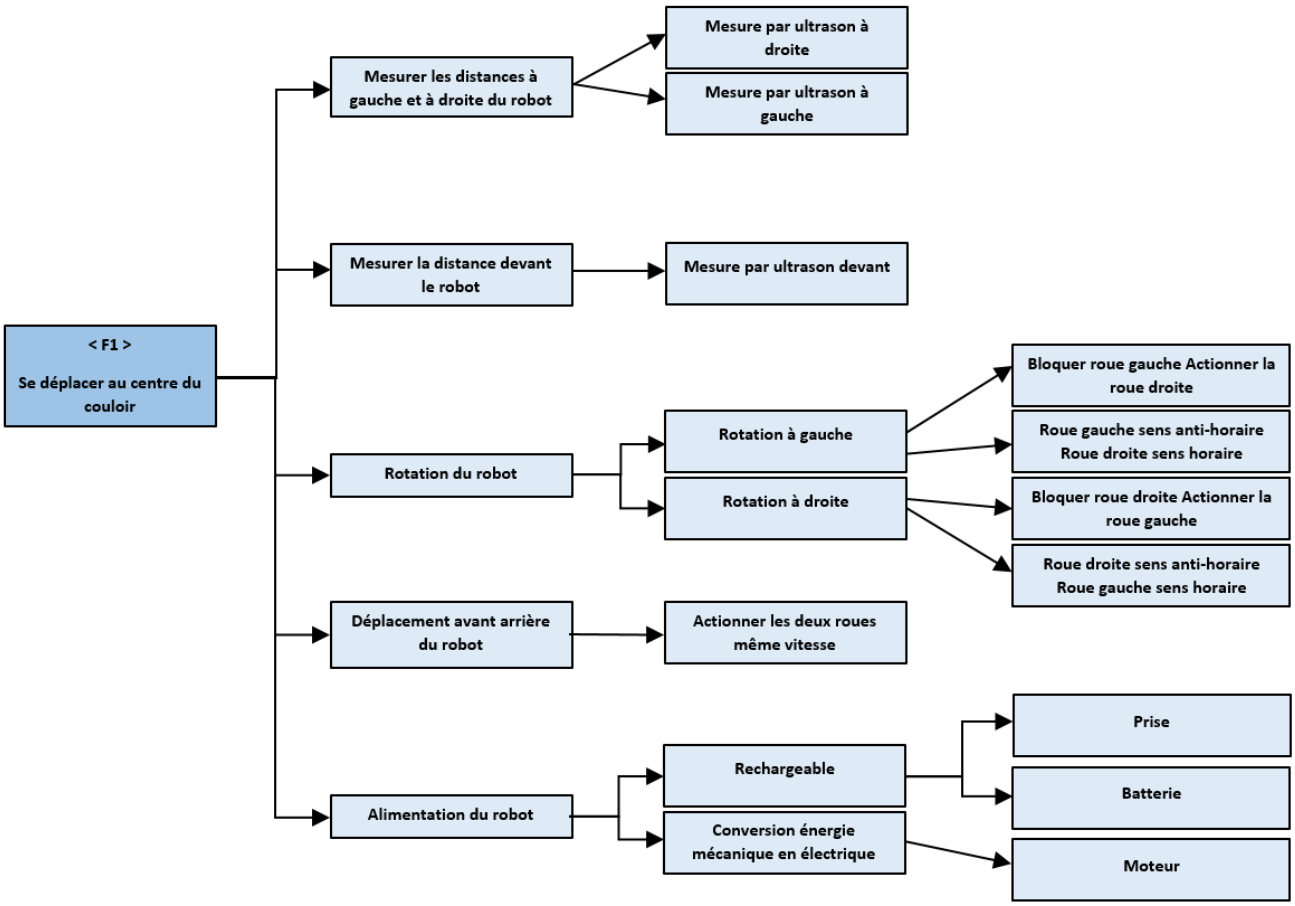
Fonctions

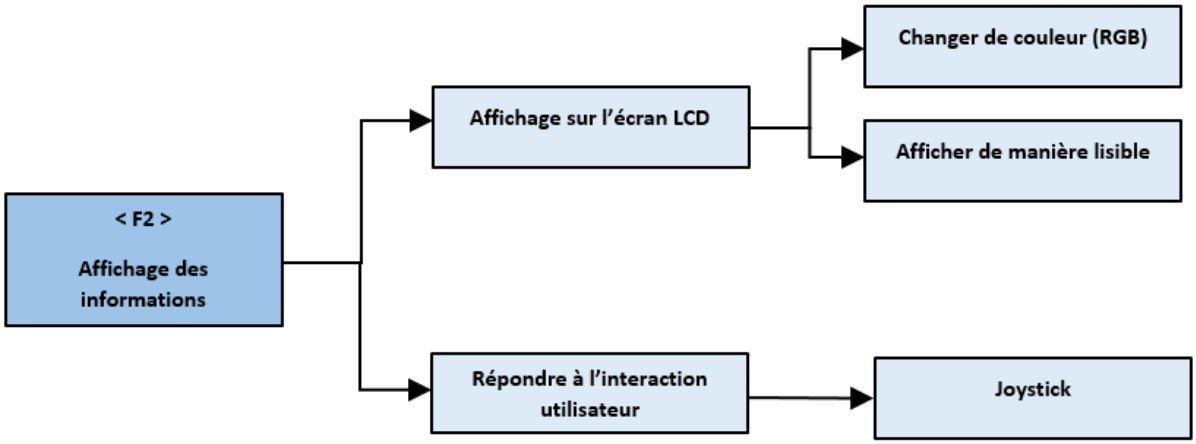
>> F1 : Pouvoir se déplacer au centre du couloir
>> F2 : Pouvoir afficher les informations (Temps, Nb tours, etc...)
>> F3 : Pouvoir atteindre l'objectif
>> F4 : Être resistant au choc et sécurisé
>> F5 : Être commandable à distance
>> F6 : Pouvoir arrêter le robot

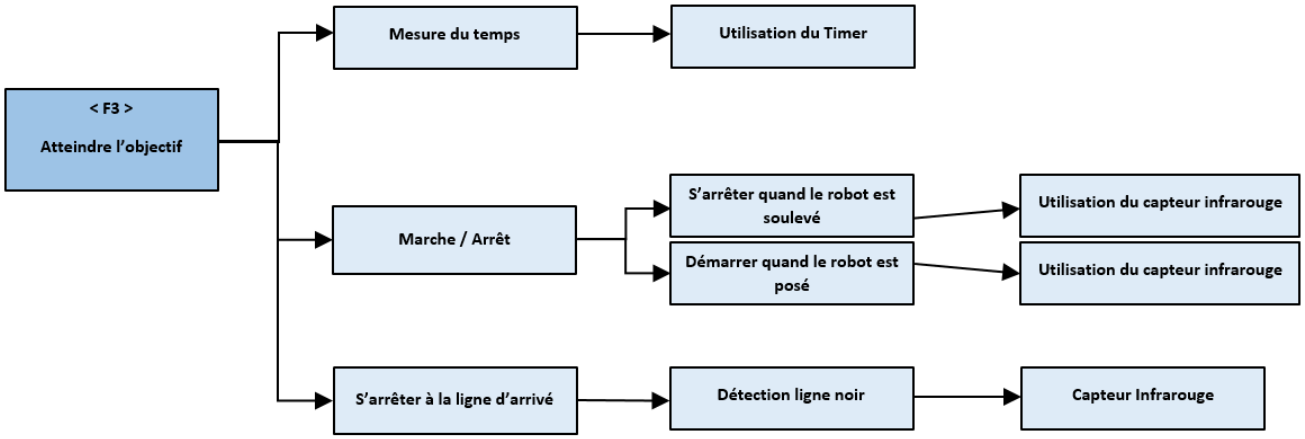
Contraintes

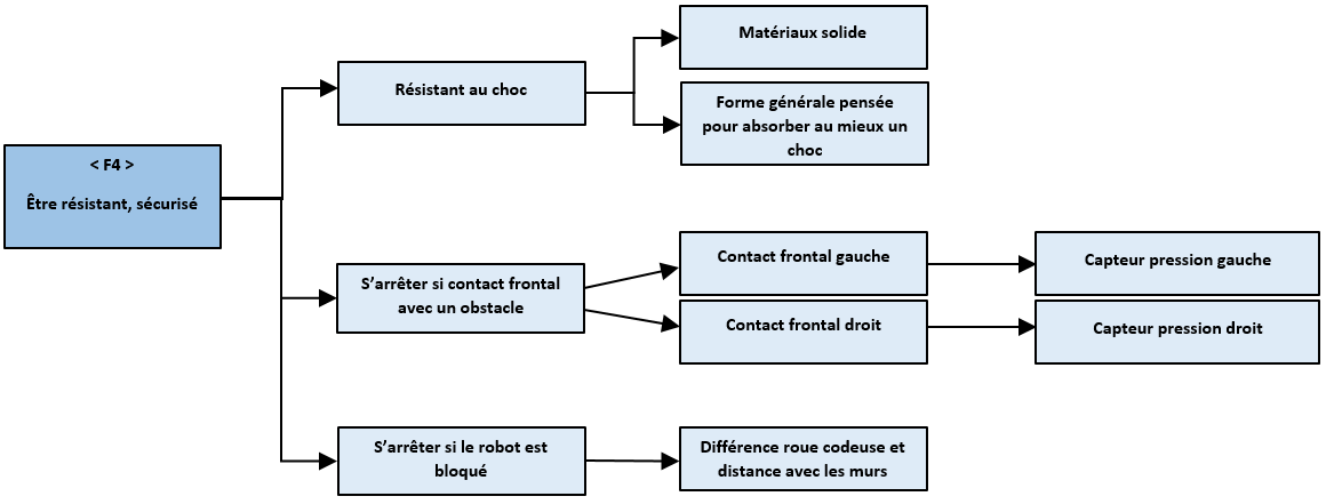
>> C1 : Rester au milieu de la piste
>> C2 : S'arreter 10min après le démarrage si parcours non fini
>> C3 : S'arreter si la distance avec un obstacle est inférieur à 5 cm
>> C4 : S'arreter si on détecte un blocage mécanique des roues
>> C5 : S'arreter si le I Moteur est supérieur au I Nominal
>> C6 : S'arreter si la rotation moteur est supérieur à 200 °/s

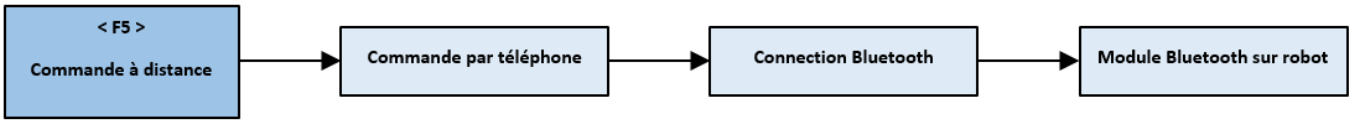
4.1) Analyse Fonctionnelle : Diagramme FAST

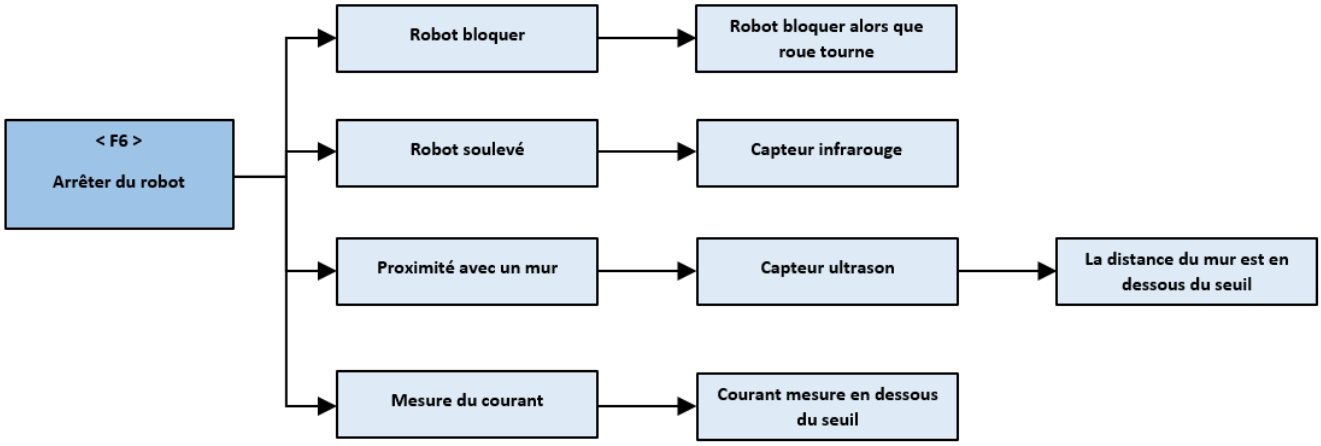




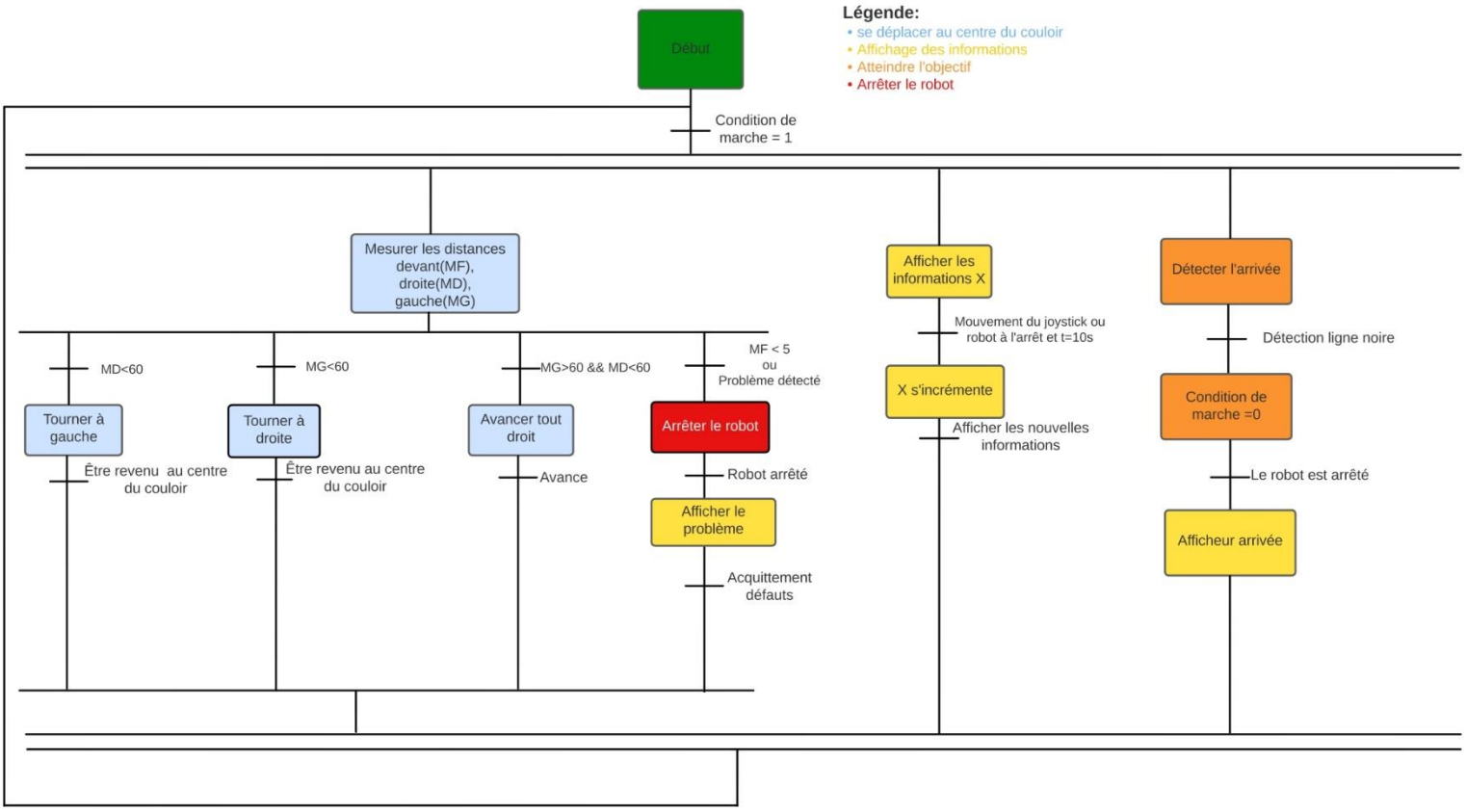

< F3 >
Atteindre l'objectif
Mesure du temps
Utilisation du Timer
Marche / Arrêt
S'arrêter quand le robot est soulevé
Démarrer quand le robot est posé
Utilisation du capteur infrarouge
Utilisation du capteur infrarouge
S'arrêter à la ligne d'arrivé
Détection ligne noir
Capteur Infrarouge


< 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

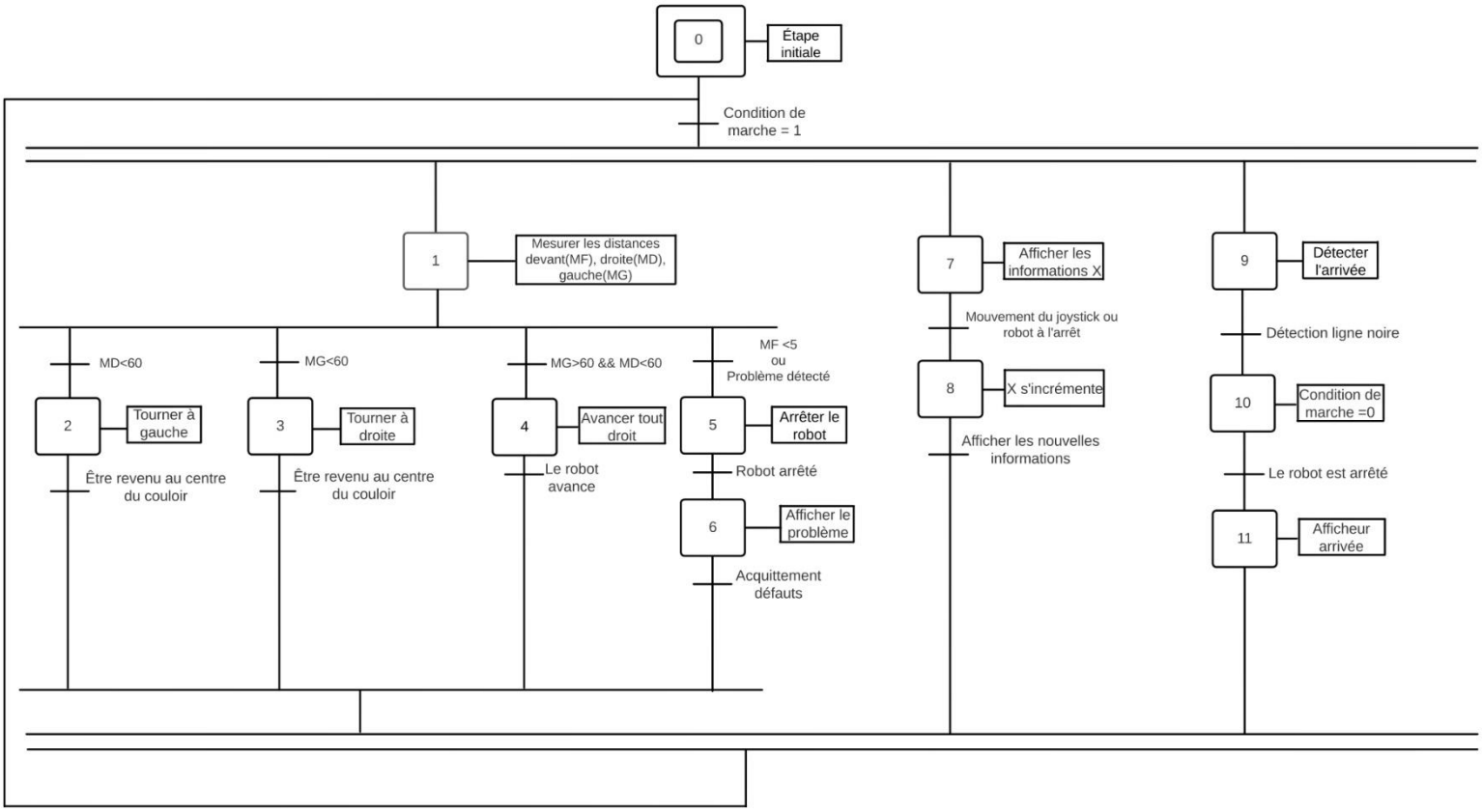

< F5 >
Commande à distance
Commande par téléphone
Connection Bluetooth
Module Bluetooth sur robot


< 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

4.2) Analyse Fonctionnelle : Organigramme



4.3) Analyse Fonctionnelle : Grafcet



5) Code de Protection

5.1) Code Protection : 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 cela, nous avons utilisé l'instruction if, et pour afficher le message d'erreur, nous avons utilisé la librairie de l'afficheur LCD.

```
#define Stop 400 // Initialisation de la valeur d'arrêt du Robot
#include <MARK.h> // librairie pour le Robot
#include "Ultrasonic.h" // librairie pour les ultrasons
#include "rgb_lcd.h" // librairie pour l'afficheur LCD
MARK myrobot;

//Fonction

void ArretDist() { // Début des actions à réaliser

  long DistanceCapteurAvant; // Déclaration d'une variable
  long DistanceCapteurDroite; // Déclaration d'une variable
  long DistanceCapteurGauche; // Déclaration d'une variable
  DistanceCapteurAvant = ultrasonic1.MeasureInCentimeters(); // Attribution d'un ultrason à une variable
  DistanceCapteurGauche = ultrasonic2.MeasureInCentimeters(); // Attribution d'un ultrason à une variable
  DistanceCapteurDroite = ultrasonic3.MeasureInCentimeters(); // Attribution d'un ultrason à une variable

  MoteurGD(600,(606)); //Le robot avance vers l'avant car >400

    if (DistanceCapteurDroite <5 || DistanceCapteurGauche < 5 || DistanceCapteurAvant < 5) { // Si le robot
    //détecte un mur devant, à gauche ou à droite <5, il rentre dans cette instruction
    MoteurGD(400, 400); // Le robot s'arrête avec la valeur d'arrêt (400)

    lcd.print("Distance courte") // On configure l'afficheur LCD pour écrire un message d'erreur
    lcd.setRGB(255, 0, 0) // On configure l'afficheur LCD en couleur rouge
 }
}
```

5.2) Code Protection : 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 utilisé le programme test 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 éviter d'abîmer le moteur.

```
#include <Encoder.h> //Librairie des encodeurs
#include "rgb_lcd.h"
#define Stop 400 //On définit la valeur d'arrêt du robot
MARK myrobot;
Encoder knobLeft(18, 33); //On définit le pin pour l'encodeur Gauche
Encoder knobRight(31, 19 );//On définit le pin pour l'encodeur Droite
long positionLeft  = -999; //On définit une position de base
long positionRight = -999; //On définit une position de base

/Fonction
/Encodeur
void ArretRB(){
    long newLeft, newRight;//On définit des variables pour les nouvelles positions
    newLeft = knobLeft.read(); //Lecture roue gauche
    newRight = knobRight.read();//Lecture roue droite
//Si position actuel est differente de la nouvelle position gauche
    if (newLeft != positionLeft){ //Si la position est différente
      positionLeft = newLeft; //On attribut la nouvelle possition
      millis();
      lcd.print("Le robot avance");
//Sinon
    }
    else{
      MoteurGD(400,400); //robot stop
      lcd.setRGB(255, 0, 0); //Affichage en couleur rouge
      lcd.setCursor(0, 0);
      lcd.print("Roue gauche bloqué"); //messgae d'erreur
    }
//Si position actuel est differente de la nouvelle position gauche
    if (newRight != positionRight){ //Si la position est différente
      positionRight = newRight; //On attribut la nouvelle position de la roue droite
      millis();
      lcd.print("Le robot avance");
//Sinon
    }
    else{
      MoteurGD(400,400); //robot stop
      lcd.setRGB(255, 0, 0); //Affichage en couleur rouge
      lcd.setCursor(0, 0);
      lcd.print("Roue droite bloqué"); //message d'erreur
    }
}
```

5.3) Code Protection : Arrêt Meca

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 est la valeur exacte et pas une valeur approchée.

```
void setup(){
LimEncSecDroite=(LimDegSec/360)*ProprieteEncDroite);//Ici on calcule la limite de l'encodeur droite
LimEncSecGauche=(LimDegreSec/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 il s'arrête car la limite est dépassée.

```
#define ProrieteEncDroite 1180
#define ProprieteEncGauche 1150
#define LimDegSec 200
int DifferentielGauche; //On mesure la différence de position t à t-1
int DifferentielDroite; //On mesure la différence de position t à t-1
void setup(){
LimEncSecDroite=(LimDegSec/360)*ProprieteEncDroite);//Ici on calcule la limite de l'encodeur droite
LimEncSecGauche=(LimDegreSec/360)*ProprieteEncGauche);//Ici on calcule la limite de l'encodeur gauche
}
/Fonction
/Encodeur
void ArretMec(){
    long newLeft, newRight;//On définit des variables pour les nouvelles positions
    newLeft = knobLeft.read(); //Lecture roue gauche
    newRight = knobRight.read();//Lecture roue droite
//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();
      lcd.print("Le robot avance");
    }
    else{//Sinon
      MoteurGD(400,400); //robot stop
      lcd.setRGB(255, 0, 0); //Affichage en couleur rouge
      lcd.setCursor(0, 0);
      lcd.print("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();
      lcd.print("Le robot avance");
    }
    else{ //Sinon
      MoteurGD(400,400); //robot stop
      lcd.setRGB(255, 0, 0); //Affichage en couleur rouge
      lcd.setCursor(0, 0);
      lcd.print("Arret Mec Droite"); //message d'erreur
    }
}
```

5.4) Code Protection : 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.

Pour faire cela, il faut pouvoir récupérer la valeur du courant. La carte arduino est composée d'entrées analogiques qui permettent de récupérer ces valeurs là si le pin est bien branché.

```
//Capteur COURANT
  #define PinCR 0  //Broche du capteur de courant droit
  #define PinCL 13 //Broche du capteur de courant gauche
  float IR;//Variable pour enregistrer le courant du moteur de droite
  float IL;//Variable pour enregistrer le courant du moteur de gauche
```

Ensuite, il faudrait récupérer le I nominal des moteurs et regarder si le courant actuel qu'on récupère est > à I Nominal

```
//Lecture du COURANT
  void LectureCourant(){
    int x=analogRead(PinCR);//On calcule le courant actuel
    IDroite = x;
    int y=analogRead(PinCL);//On calcul le courant actuel
    IGauche =y;
    if(IDroite>INominal || IGauche>INominal){
      MoteurGD(400,400); //robot stop
      lcd.setRGB(255, 0, 0); //Affichage en couleur rouge
      lcd.setCursor(0, 0);
      lcd.print("Arret Elec"); //message d'erreur
      }
    }
```

6) Code Intermédiaire

Le code intermédiaire a pour but de rendre le robot capable de se déplacer au centre d'un couloir et il devra s'arrêter au premier mur rencontré en face.

Pour cela, nous avons utilisé l'instruction if pour que le robot tourne dans le sens opposé si jamais il est trop proche d'un des deux côtés du mur.

```
void loop() {
  long DistanceCapteurAvant;
  long DistanceCapteurDroite;
  long DistanceCapteurGauche;
  DistanceCapteurAvant = ultrasonic1.MeasureInCentimeters();
  DistanceCapteurGauche = ultrasonic2.MeasureInCentimeters();
  DistanceCapteurDroite = ultrasonic3.MeasureInCentimeters();


if(DistanceCapteurGauche > 60  &&  DistanceCapteurDroit < 60){ // Le robot va tout droit s'il est au centre du couloir
        MoteurGD(600, 622) ; //Vitesse normal (la roue droite est plus élevé car elle est plus lente à même vitesse)
        millis();
}
//On tourne vers la droite
if(DistanceCapteurGauche < 60){ //Si la distance du robot avec le mur gauche est trop petite, alors le robot se décale légèrement vers la droite
        MoteurGD(610, 622);
        millis();
}
//On tourne vers la gauche
if(DistanceCapteurDroite < 60){ //Si la distance du robot avec le mur droit est trop petite, alors le robot se décale légèrement vers la gauche
        MoteurGD(600, 632);
        millis();
}

//Sécurité
if(DistanceCapteurAvant < 5 ){ //Le robot s'arrête s'il est trop près du mur devant lui
        MoteurGD(400, 400);
}

 }
```

7) Conclusion

Après avoir eu 5 séances, nous avons pu découvrir le projet Robot Mark et nous avons bien commencé le projet, nous avons les codes de protections et nous pouvons faire avancer le robot en ligne droite.

La prochaine phase du projet consiste à faire la programmation pour que le robot puisse faire le parcours, puis la programmation des différentes informations qu'il devra afficher une fois le parcours fini. Nous allons aussi créer le Github afin de rassembler toutes nos informations sur ce projet.

Il s'agit d'un projet intéressant, reflet d'un travail d'entreprise, allant de l'analyse du Cahier des Charges, à la réalisation d'un produit fini.

Annexes

Annexe : Programme Test

Capteur Ultrasons

```
#include "Ultrasonic.h"

Ultrasonic ultrasonic(8);
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  long RangeInCentimeters;

  Serial.println("The distance to obstacles in front is: ");
  RangeInCentimeters = ultrasonic.MeasureInCentimeters(); // two measurements should keep an interval
  Serial.print(RangeInCentimeters);//0~400cm
  Serial.println(" cm");
  delay(250);
}
```

Capteur Infrarouge

```
#include <MARK.h>

MARK myrobot;

void setup() {
  myrobot.begin();
}

void loop() {
  if ( myrobot.getInfrared()) {
    myrobot.lcdPrint("1");
  }
  else {
    myrobot.lcdPrint("0");
  }
  myrobot.lcdHome();

}
```

Capteur Accéléromètre

```
#include <MARK.h>
MARK myrobot;

void setup() {
  myrobot.begin();
}

void loop() {
  myrobot.setLcdCursor(0, 0);
  myrobot.lcdPrint(myrobot.getAccelX());
  myrobot.setLcdCursor(0, 1);
  myrobot.lcdPrint(myrobot.getGyroX());
  myrobot.setLcdCursor(8, 0);
  myrobot.lcdPrint(myrobot.getAccelY());
  myrobot.setLcdCursor(8, 1);
  myrobot.lcdPrint(myrobot.getGyroY());
  /*other data from IMU :
   * getAccelY()
   * getAccelZ()
   * getGyroX()
   * getGyroY()
   * getGyroZ()
   * getTemp()
   */
  myrobot.lcdPrint("    ");
}
```

Afficheur LCD

```
#include <Wire.h>
#include "rgb_lcd.h"

rgb_lcd lcd;

void setup() {
    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);
    // Print a message to the LCD.
    lcd.print("hello, world!");
}

void loop() {
    // Turn off the display:
    lcd.noDisplay();
    delay(500);
    // Turn on the display:
    lcd.display();
    delay(500);
}
```

Moteur Roues

```
#define Thash 800
#define Stop 400
#define Vmax Thash
// 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)

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 setup() {
  pinMode(43,OUTPUT);
  digitalWrite(43,0);
  initMoteurs();
  sei();
  digitalWrite(43,1);
}

void loop() {
  int i;
  for (i = Stop; i < Vmax ; i++) { // Accelerer les 2 moteurs
    MoteurGD(i,i);
    _delay_ms(5);
  }
  _delay_ms(1000);
  for (i = Vmax; i > Stop; i--) { // Decelerer les 2 moteurs
    MoteurGD(i,i);
    _delay_ms(5);
  }
}
```

Afficheur LED-Bar

```
#include <MARK.h>
MARK myrobot;

void setup() {
  myrobot.begin();
}

void loop() {
  for (int i = 0; i <= 10; i++) {
  myrobot.setLedBarLevel(i);
  delay(1000);
  }
}
```

Contrôleur Joystick

```
#include "rgb_lcd.h"
#include <Wire.h>

rgb_lcd lcd;

void setup()
{
    Serial.begin(9600);
    pinMode(A2,INPUT);
    pinMode(A3,INPUT);
    lcd.begin(16, 2);
    lcd.setRGB(200, 200, 200);
}

void loop()
{
    int sensorValue1 = analogRead(A2);
    int sensorValue2 = analogRead(A3);
    lcd.setCursor(0,0);
    lcd.print("The X and Y is:");
    lcd.setCursor(0,1);
    lcd.print(sensorValue1);
    lcd.setCursor(13,1);
    lcd.println(sensorValue2);
    delay(200);
    lcd.clear();
}
```

Annexe : Programme Intermédiaire

```
#define Thash 800
#define Stop 400
#define Vmax Thash
#include <MARK.h>
#include <Wire.h>  //lib for I2C connection
#include "Ultrasonic.h"
#include "rgb_lcd.h"

Ultrasonic ultrasonic1(8); //Init of ultrasonic snesor on pin 8
Ultrasonic ultrasonic2(10);
Ultrasonic ultrasonic3(12);
const byte infrared = 6;
MARK myrobot;

#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)

void initMoteurs() {  // MoteurG :OC5A=PIN46-PL3, MoteurD : OC5B=PIN45-PL4
  DDRL = 0x18 ; // PL3 et PL4
  DDRB = 0x80 ; // PB7 LedToggle

  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 setup() {
  pinMode(43, OUTPUT);
  digitalWrite(43, 0);
  initMoteurs();
  sei();
  digitalWrite(43, 1);

}
```

```
void loop() {
  long DistanceCapteurAvant;
  long DistanceCapteurDroite;
  long DistanceCapteurGauche;
  DistanceCapteurAvant = ultrasonic1.MeasureInCentimeters();
  DistanceCapteurGauche = ultrasonic2.MeasureInCentimeters();
  DistanceCapteurDroite = ultrasonic3.MeasureInCentimeters();


if(DistanceCapteurGauche > 60  &&  DistanceCapteurDroit < 60){ // Le robot va tout droit s'il est au centre du couloir
          MoteurGD(600, 622) ; //Vitesse normal (la roue droite est plus élevé car elle est plus lente à même vitesse)
          millis();
}
//On tourne vers la droite
if(DistanceCapteurGauche < 60){ //Si la distance du robot avec le mur gauche est trop petite, alors le robot se décale légèrement vers la droite
          MoteurGD(610, 622);
          millis();
}
//On tourne vers la gauche
if(DistanceCapteurDroite < 60){ //Si la distance du robot avec le mur droit est trop petite, alors le robot se décale légèrement vers la gauche
          MoteurGD(600, 632);
          millis();
}

//Sécurité
if(DistanceCapteurAvant < 5 ){ //Le robot s'arrête s'il est trop près du mur devant lui
          MoteurGD(400, 400);
}

 }
```