Département de Génie électrique

Filière d'Ingénieurs

Control, Commande, Systèmes Electriques

(F.I C.C.S.E)

Livrable N°2 du Projet Robot MARK

Tests Unitaires, Algorithmes et Code Intermédiaire du robot

Réalisé par :

- **VEYRIOL Luc**
- **RAOULT Lucas**

Sous l'encadrement de :

- **SALHI Issam**
- **BARAKAT Abdallah**
- **DELFIEU David**

Année universitaire: 2019/2020

Table des matières

I – Introduction

A – État d'avancement du projet

L'état d'avancement du projet à l'amorce d'entamer les tests capteurs / moteurs et la première version du codage est la suivante : l'analyse fonctionnelle vient d'être réalisée. Cela a permis d'éclaircir beaucoup d'aspects du projet :

1. Identifier les fonctions formulées dans le cahier des charges
2. Énumérer les potentielles solutions pour chacune des fonctions de contrainte
3. Évoquer le choix des solutionss

À Noter : Un planning est tenu à jour (cf : Annexe 1)

B – Objectif des tests unitaires et du codage intermédiaire

L'Objectif de cette phase du projet est double :

- Dans un premier temps, il faut tester les équipements séparément les uns des autres. Cela convient d'écrire puis d'exécuter les programmes associés à chacun des équipements (capteurs / moteurs) nécessaires pour en comprendre et en vérifier le fonctionnement.

- À la suite de cette première réalisation, le second objectif intervient : l'idée est de rassembler ces programmes « tests » en un seul et d'en modifier ses instructions / fonctions, de sorte que le robot puisse atteindre le point intermédiaire ('1' sur la figure 1), c'est-à-dire arrêter le robot lorsque le premier mur de face est détecté. Cela permet de valider que le robot corrige bien sa trajectoire en ligne droite. Ensuite, il faut gérer les virages ou les rotations de manière à atteindre l'arrivée ('2') et stopper le robot.

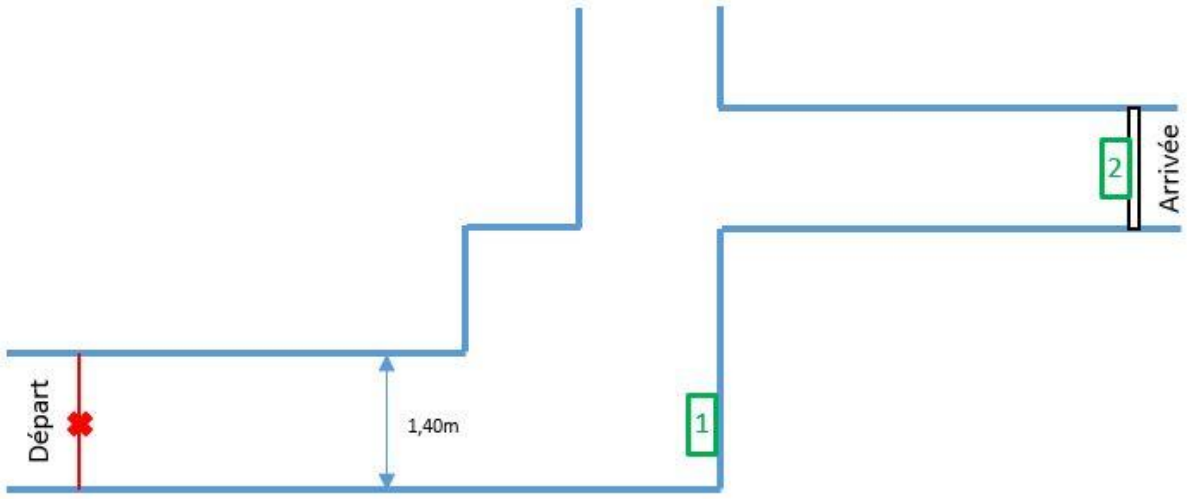


Figure 1 : Plan du terrain d'essai

Aussi, cette seconde mission est l'occasion pour introduire la notion d'interruption. Au-delà du fait qu'elles nous seront utiles dans la gestion dite « multitâche » de notre programme, c'est aussi l'opportunité idéale d'étudier un outil largement utilisé dans les technologies industrielles.

II – Tests unitaires

A – Ultrasons (exemple détaillé)

test_Ultrasons §

```
#include "rgb_lcd.h"
#include "Ultrasonic.h"
//déclaration des structures: 3 Ultasons brochés aux PIN 8, 10, 2 + 1 afficheur LCD
Ultrasonic us_front(8);
Ultrasonic us_right(10);
Ultrasonic us_left(2);
rgb_lcd lcd;

//déclaration de variables globales --> préparation à l'utilisation des interuptions
volatile int dist_us_right, dist_us_left, dist_us_front;

void setup() {
  lcd.begin(16, 2);   //initialisation de l'afficheur LCD 16 colonnes, 2 lignes par défaut à 9600baups = 9600bits/sec
}

void loop() {
  //***Affectations des valeurs mesurées par les récepteurs ultrasons aux variables globales respectives***
  dist_us_front = us_front.MeasureInCentimeters(); // dist_us_front prend la valeur de la mesure en cm de l'US de front
  dist_us_left = us_left.MeasureInCentimeters();
  dist_us_right = us_right.MeasureInCentimeters();
  //***Affichage des données ultrasons***
  lcd.setCursor(9,0);   //fonction propre à rgb_lcd.h --> placement du curseur
  lcd.print(dist_us_front);   //fonction propre à rgb_lcd.h --> afficher la variable "dist_us_front"
  lcd.setCursor(0,1);
  lcd.print(dist_us_left);
  lcd.setCursor(9,1);
  lcd.print(dist_us_right);
  delay(1000);    //temporisation d'une seconde entre deux cycle (loop)
}
```

Figure 2 : Tests capteurs Ultrason (x3) + afficheur LCD

Le test unitaire des capteurs ultrasons consiste à afficher sur l'afficheur LCD les mesures des trois capteurs ultrasons, eux-mêmes définis par leurs directions d'émission/réception. Pour effectuer l'affichage d'une mesure d'ultrason, on peut suivre le protocole suivant :

1. Déclarer une variable globale qui recevra la mesure d'un ultrason
2. Affecter à cette variable globale, la valeur de la mesure renvoyée par la fonction « ***MeasureInCentimeter() » (***propre à l'ultrason) ;
3. Positionner le curseur de l'afficheur LCD avec la fonction ***setCursor(colonne,ligne)*** » (propre à l'afficheur LCD). C'est-à-dire, d'où le premier caractère de la prochaine écriture va débuter sur l'afficheur.
4. Écrire sur l'afficheur avec la fonction « ***print(variable_globale_ultrason)*** » (propre à l'afficheur).

B – Moteurs

Les moteurs associés aux roues sont les éléments cruciaux pour convenir du déplacement du robot. Deux fonctions présentes dans leurs bibliothèques ("Grove_I2C_Motor_Driver.h") seront de grande utilité dans le code final :

- Motor.speed (ID du moteur, vitesse) ;
- Motor.stop (ID du moteur) ;

Le code test exécute une rotation des deux roues à 30% de leurs maximums pendant une seconde avant de les stopper une autre seconde, en Bouclage infinie.

```
void loop() {
  Motor.speed(MOTOR1, 30);  //Moteur Gauche à vitesse 30 (min:-100; max:100)
  Motor.speed(MOTOR2, 30);  //Moteur Droit à vitesse 30
  delay(1000);  //Temporisation d'une seconde
  Motor.stop(MOTOR1); //Moteur Gauche stoppé
  Motor.stop(MOTOR2); //Moteur Droit stoppé
  delay(1000);  //Temporisation d'une seconde
}
```

Figure 3 : Test des moteurs droit et gauche

C – Joystick

Le joystick contribue fortement à l'ergonomie du système. En effet, il commande notre affichage LCD et bien plus encore, il contrôle le Marche/Arrêt de notre robot (différent du bouton ON/OFF d'alimentation).

```
int sensorValue1 = analogRead(A2);
int sensorValue2 = analogRead(A3);
```

Figure 4 : Test du joystick

On peut stocker en temps réel la position du joystick grâce à la fonction **analogRead ().**

Sachant que le pin A2 correspond à l'axe vertical, et le pin A3 à l'axe horizontal, on peut facilement donner une consigne au robot à une certaine position.

Nous avons aussi remarqué qu'une impulsion sur je joystick permettait de faire ressortir une valeur de 1023 sur le pin A2.

D – Capteur infrarouge

Le capteur infrarouge devrait être utilisé pour la détection de ligne au sol. Quand le capteur infrarouge reçoit une couleur différente du blanc (coloris du sol), alors, il devra suivre la ligne jusqu'à détecter le mur du fond.

Le test du code présent ci-dessous fonctionne. Mais la détection du capteur infrarouge s'effectue seulement au contact d'un élément.

```
if (digitalRead(infrared)) {  //SI détection capteur infrarouge différent du blanc
  lcd.setRGB(0, 255, 0);  //couleur de fond du LCD VERT
}
else {
  lcd.setRGB(255, 0, 0);  //couleur de fond du LCD ROUGE
}
```

Figure 5 : code test du capteur infrarouge

III – Code intermédiaire

A – Introduction aux interruptions

- **Qu'est qu'une interruption ?**

Une interruption, comme son nom l'indique, interrompt une boucle principale pour exécuter un sous-programme. Une interruption est toujours associée à un Timer de façon à la cadencer à une fréquence choisie. Elle doit être de courte durée et donc ne contenir qu'un minimum d'instructions linéaires (ni boucle, ni calcul) de manière à donner l'impression de fonctionner en même temps que le programme principal. En réalité le programme principal se « paralyse » quelques microsecondes afin de donner la priorité au sous-programme (l'interruption). De cette association, entre programme et sous-programme qui s'exécutent quasi-parallèlement, naît le concept de fonctionnement multitâche.

À Noter : un « delay() ; » est une forme d'interruption. À l'exception et grande problématique qu'elle stoppe le programme entier et ne peut donc gérer aucune application multitâche.

Deux grands types d'interruptions existent :

1. D'une part, les interruptions externes faisant appel à la fonction « **ISR (nom d'évènement)** », le format suivant est aussi utilisé : « **attachInterrupt (type d'interruption, fonction, mode)** ». Le vecteur d'interruption (ou source d'interruption ou nom d'évènement) affecte directement une broche des entrées / sorties Analogiques de l'Arduino. Dans le cas d'une interruption externe, l'utilisation des noms d'évènement INT0_vect ou INT1_vect est commune (seul le pin Arduino diffère entre ces deux mêmes types d'interruptions externes). Elles interviendront en cas de front montant ou front descendant comme pour un arrêt d'urgence.

2. D'autre part, les interruptions internes utilisant aussi la fonction « ISR (nom du vecteur d'interruption) » mais cette fois-ci afficheront un vecteur d'interruption différent des interruptions externes. Dans notre code, c'est le vecteur d'interruption « *Timer2_OVF_vect* » qui est utilisé, exécutant l'interruption à chaque TOP d'horloge

- **Quelles sont les applications des interruptions ?**

Les applications sont diverses selon si les interruptions sont internes ou externes. Les interruptions internes peuvent gérer les conversions Analogique/Numérique, le seuil d'un compteur, une temporisation tandis que les interruptions externes seront plus utilisées pour détecter un front sur une broche du microcontrôleur (Arrêt d'urgence, Chronomètre, Dépassement d'un seuil analogique).

- **Comment fonctionne une interruption interne?**

Une interruption interne se programme à l'aide de registres systèmes, propres au Timer associé à l'interruption (Timer0, Timer1, Timer2, …etc). Chacun de ces registres contient des bits impactant, le comportement, le déclenchement, la cadence de l'interruption.

- **Quelle interruption utilise-t-on dans notre projet ?**

À ce stade du projet, nous utilisons seulement une interruption interne associée au Timer2 de L'ATmega2560, pour l'actualisation des mesures des capteurs ultrasons. Quatre registres sont nécessaires à la compréhension du Timer2 et de l'interruption interne utilisée : TCCR2A, TCCR2B, TCNT2 et TIMSK.

TCNT2 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
(0xB2)	TCNT2[7:0]								TCNT2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 6 : Registre TCNT2

Le registre TCNT2 est un compteur qui s'incrémente périodiquement.

TCCR2A –Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
(0xB0)	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 7 : Registre TCCR2A

Le Registre TCCR2A possède six bits qui nous intéressent : Les deux bits WGM21 :0 (Waveform Generation Mode) contrôlent la séquence de comptage, c'est-à-dire le mode de comparaison pour lequel l'interruption aura lieu. Les quatre bits *COM2X0 :1* définissent dans quelles conditions OC2 (le signal de sortie) doit commuter ou non (0←→1).

TCCR2B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
(0xB1)	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 8 : Registre TCCR2B

Dans le registre TCCR2B, on reconnaît un bit WGM22. Trois autres bits nommés CS2 :0 impacteront la fréquence du signal d'horloge interne en la divisant. Ces trois bits sont plus communément appelés « *Pré-diviseur* ».

Table 20-9. Clock Select Bit Description

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)

Figure 9 : Affectations du pré-diviseur selon ses bits d'entrées CS2:0

On retrouve les effets produits sur la fréquence d'horloge (Clk_{T2S}) selon les différentes valeurs des bits CS2:0.

TIMSK2 – Timer/Counter2 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x70)	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2	TIMSK2
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 10 : Registre TIMSK

Quand le bit TOIE2 est actif, il autorise l'interruption de débordement.

- **Comment avons-nous programmé l'interruption ?**

```
//*****FONCTIONS D'INTERRUPTION PAR TIMER2*****
ISR(TIMER2_OVF_vect){
  tempo++;  //A CHAQUE TOP ATTEINT --> INCRÉMENTATION DU COMPTEUR = TEMPORISATION ENTRE 2 MESURES
  if(tempo > 30){  // QUAND LE TOP A ÉTÉ ATTEINT 30 fois --> ON AFFECTE LES MESURES DES ULTRASONS À LEUR VARIABLE RESPECTIVE
    //*****CAPTEURS ULTRASONS*****
    dist_us_front = us_front.MeasureInCentimeters(); //déclaration d'une variable globale: prend la valeur de la mesure en cm de l'US de front
    dist_us_left = us_left.MeasureInCentimeters();  //..... de gauche
    dist_us_right = us_right.MeasureInCentimeters(); //..... de droite
    tempo = 0;  //Remise à 0 du compteur
  }
}

void configTimer2(){  //configuration du Timer 2
  TCCR2A = 0; //choix mode: WGM2 = 0, WGM1 = 0, WGM0 = 0 -->  0 = mode normal
  TCCR2B = (1<<CS22)+(1<<CS21); //CHOIX DU PRÉDIVISEUR = 256
  TIMSK2 |= (1<<TOIE2); //TOIE2: "Timer/Counter2 Overflow Interrupt Enable" --> Rend possible l'interrution de débordement
}

void setup() {
  pinMode(A2,INPUT);  //DÉCLARATION DE LA PIN ANALOGIQUE "A2" EN ENTRÉE --> appuie sur JOYSTICK
  pinMode(A3,INPUT);  //DÉCLARATION DE LA PIN ANALOGIQUE "A3" EN ENTRÉE --> mouvement JOYSTICK
  lcd.begin(16, 2);   //INITIALISATION DE LA TAILLE DE L'AFFICHEUR
  configTimer2();   //APPEL DE LA FONCTION "CONFIGTIMER2"
  sei();            //AUTORISE LES INTERRUPTIONS A SE DÉLENCHER
}
```

Figure 11 : Codage d'une interruption associée à un Timer2

L’interruption interne **« ISR(Timer1_OVF_vect) »** de notre programme principal, réceptionne les mesures des trois ultrasons du robot MARK et les affecte à des variables globales précédemment déclarées (volatile int dist_us_front, dist_us_right, dist_us_left). Cela s’effectue tous les trente cycles d’interruption pour éviter au microcontrôleur de saturer en affectant à chaque instant, les mesures. Tout cela s’effectue parallèlement au pilotage des moteurs et de l’affichage LCD.

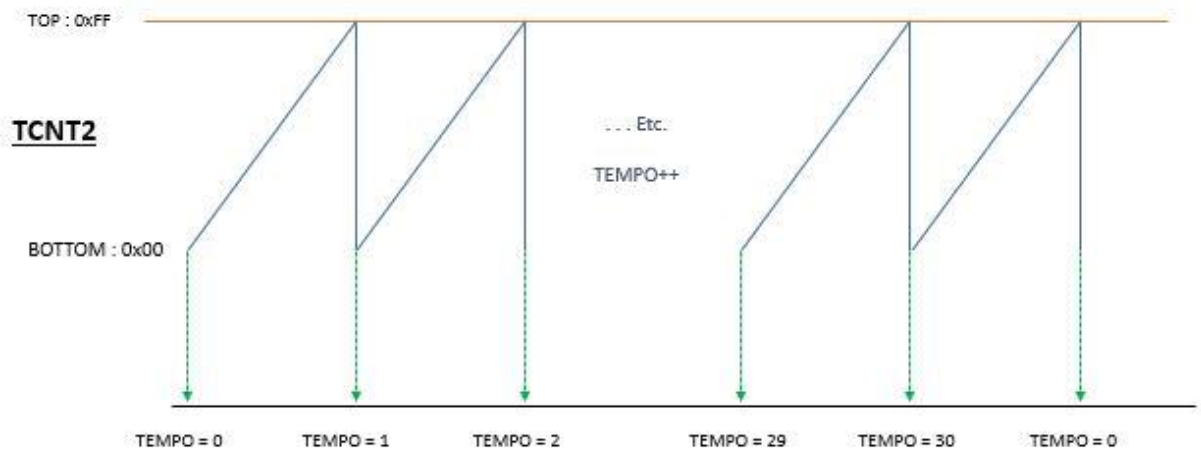


Figure 12 : représentation du signal TCNT2 associé à l'interruption

B – Mode de fonctionnement

Pour la partie fonctionnement des moteurs :

Nous n'avons pas encore prévu de correction proportionnelle, nous avons des conditions avec des valeurs fixes. Pour l'instant nous pouvons résumés notre avancés par ce graphique :

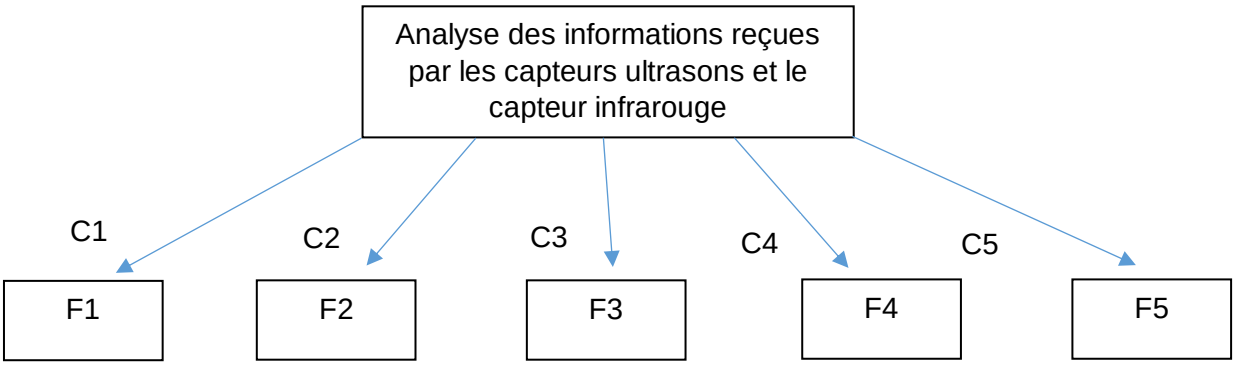


Figure 13 : Schéma d'analyse des données ultrasons

C1 : Si la distance du mur droite est inférieur à 60 et supérieur à 40
F1 : Avance normale

C2 : Si la distance du mur droite est inférieur à 40
F2 : Tourne un peu vers la gauche

C3 : Si la distance du mur droite est supérieur à 60
F3 : Tourne un peu vers la droite

C4 : S'il y a un obstacle en face
F4 : Virage sec vers la gauche

C5 : S'il n'y a rien en face et rien à droite
F5 : Virage sec vers la droite

Nous voudrions dans le futur effectuer une correction proportionnelle pour gagner en efficacité dans les virages. On pourra ainsi adapter la vitesse de chaque roue en fonction de de la correction à apporter.

Pour la partie affichage :

Nous avons choisi un affichage sur l'écran LCD pouvant être gérer avec le joystick, on peut ainsi afficher les informations que l'on veut en défilant sur les différents écrans préenregistrés.

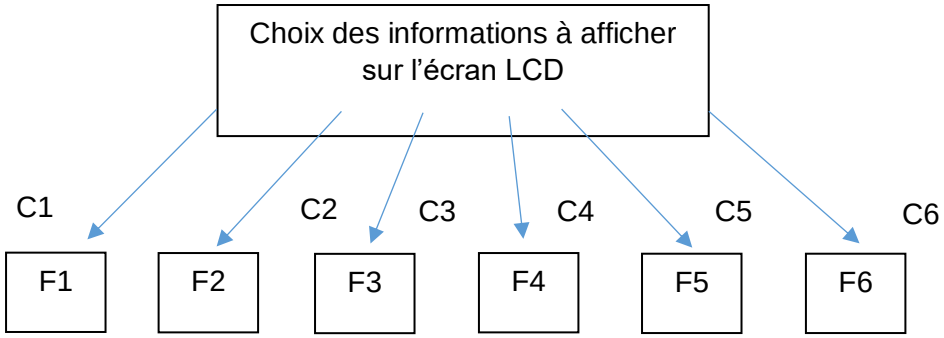


Figure 14 : Schéma de la gestion de l'afficheur LCD

C1 : Ecran 1
F1 : Etat de marche du robot

C2 : Ecran 2
F2 : Vitesse maximum de chaque roue du robot

C3 : Ecran 3
F3 : Nombre de tour de chaque roue du robot

C4 : Ecran 4
F4 : Distance parcourue par le robot

C5 : Ecran 5
F5 : Temps écoulé

C6 : Ecran 6
F6 : Affichage du nombre de fois ou le robot à dépasser la limite des 20 cm de distance avec un mur

C – Détails

Affichage :

La figure 4 représente le programme gestion de l'affichage de l'écran LCD

```
if (analogRead(A3) > 700 ){
  ecran = ecran+1;
  lcd.clear();
  _delay_ms(200);
}

 if (analogRead(A3) < 400 ){
  ecran = ecran-1;
  lcd.clear();
  _delay_ms(200);
}
```

Figure 15 : Programme de la commande écran LCD

Comme dit précédemment, un mouvement du joystick sur l'axe horizontal permet de changer l'écran et donc l'information à afficher.

Ainsi lorsque le joystick se déplace vers la gauche (analogRead(A3) > 700) la variable écran est incrémenté, provoquant ainsi le changement dans le switch (figure 10).

Lorsque le joystick se déplace vers la droite (analogRead(A3) < 400) la variable écran est décrémenté. Cela permet de revenir à l'écran d'avant.

Nous utilisons un délai de 200 millisecondes dans chaque condition afin d'éviter un défilement trop rapide d'écran et de laisser le temps à l'utilisateur de choisir celui qui l'intéresse.

```
switch (ecran){
  case 0:
    lcd.setCursor(0,0);
    lcd.print("ETAT DU ROBOT :");
    lcd.setCursor(0,1);
    if(start == 1) lcd.print("Marche");
    else lcd.print("Arret");
    break;
```

Figure 16 : Switch pour ecran = 0

Pour *ecran = 0*, l'écran informe l'utilisateur sur l'état de marche du robot (Marche ou Arrêt)

Lcd.set.Cursor(0,0) permet de placer le curseur à la position 0 de la ligne 0.

Avec ce programme on obtient donc l'affichage de la figure 6 :

Figure 17 : Affichage pour ecran = 0

Nous avons fait ceci pour chaque information voulue (voir **B - Mode de fonctionnement** et Annexe 2).

Si la variable *ecran* prend une valeur différente que les valeurs programmées dans le swtich, l'écran d'état de marche du robot réapparait, ainsi si l'utilisateur arrive au dernier écran d'information et qu'il continue à incrémenter la variable, il retournera au début créant ainsi une boucle (figure 13).

```
default :
  ecran = 0 ;
  break;
```

Figure 18 : Boucle d'affichage en cas de dépassement

Calculs :

Pour le calcul du nombre de tour de chaque roue, on utilise l'encodeur, il va lire et retourner des valeurs qui seront stockés dans des variables (newLeft et newRight) (figure 14).

La division par 3600 de ces valeurs permet d'obtenir le nombre de tour de chaque roue et les résultats seront stockés dans positionLeft et positionRight.

On pourra ainsi afficher ces variables sur le LCD.

```
//-------------------- Nombre de tour de chaque Roue --------------------

  newLeft = knobLeft.read();
  newRight = knobRight.read();
  if (newLeft != positionLeft || newRight != positionRight) {
    positionLeft = newLeft / 3600;
    positionRight = newRight / 3600;
  }
```

Figure 19 : Calcul du nombre de tours de roue

Nous devons compter le nombre de fois où le robot est situé à moins de 20cm d'un obstacle.

Pour ce faire, lorsqu'un capteur détecte une valeur < 20cm, nous incrémentons une variable (comp_dist) (figure 19).

Test permet d'éviter une incrémentation continue de la variable comp_dist, il faut attendre d'être à une distance supérieure à 20 cm de l'obstacle gênant pour reprendre le comptage.

```
//------------------- Compter distance < 20 cm ----------------------
*/

  if ((dist_us_front < 20 or dist_us_left < 20 or dist_us_right < 20) and test == 0 ){
   compt_dist = compt_dist + 1;
   test = 1;
  }
  else{
    if (dist_us_front > 20 and dist_us_left > 20 and dist_us_right > 20 and test == 1){
     test = 0;
    }
  }
```

Figure 20 : Calcul lorsque distance < 20

Nous n'avons pas encore réalisé les calculs de temps et de vitesse maximale de chaque roue, de plus ces calculs s'effectuent lorsque le robot est alimenté, dans le futur nous feront en sorte qu'ils ne se fassent que lorsque le robot est en état de marche.

IV – Conclusion

Selon le planning, le projet est en bonne voie pour atteindre les objectifs fixés au début du projet. D'un côté, un léger retard est constaté concernant les tests non effectués de certains équipements comme le module wifi, le servomoteur, le diviseur de tension. Cela s'explique par le fait qu'il n'a pas été encore nécessaire de les tester pour répondre efficacement à la fonction principale (qui est d'effectué le trajet de A à B, sans se rapprocher à moins de vingt centimètres des murs). Cependant, l'avancée de notre programme informatique est en légère avance. En effet, en démarrant le robot à la position initiale du parcours, il se déplace en plus ou moins une minute jusqu'à l'arrivée.

La prochaine amélioration informatique devrait concerner la structure du programme, qui pourrait être déclinée en différentes fonctions. Par exemple, rendre possible différentes manières d'atteindre l'arrivée (virage différentiel, rotation du robot par rapport à une roue ou par rapport à son centre) pour comparer les différentes méthodes et mettre en application la meilleure afin de gagner du temps. Enfin, le calcul et l'affichage de la consommation énergétique du robot feront parties des prochaines évolutions à apporter.

Ce second livrable est l'occasion de réunir de nombreuses connaissances acquises au cours des dernières semaines. En effet, un travail autour des microcontrôleurs et de leurs fonctionnements a été effectué. Cela a permis de mieux comprendre les rouages de ces micro-cerveaux électroniques, des interruptions auxquelles sont intiment liés les Timers jusqu'à leurs registres systèmes dont ils sont composés.

Ces nouvelles connaissances théoriques induisent de nouvelles compétences techniques comme la configuration de Timer0, Timer1, Timer2, ou la création d'interruption sous Arduino pour effectuer de la gestion « multitâche ». Cette dernière compétence, indispensable à la compréhension de produits, familiers tels que les Systèmes d'exploitation, ou plus futuristes avec l'utilisation de plus en plus fréquente d'intelligences artificielles toujours plus cognitives.

L'application de ces notions au projet MARK et plus précisément à la programmation comportementale du robot face à un environnement donné, finalise l'apprentissage aux microcontrôleurs.

V – Bibliographie

« Afficheur LCD RGB Grove 104030001 ». https://www.generationrobots.com/fr/402442-afficheur-lcd-rgb-grove.html (17 octobre 2019).

« Batterie Li-Ion 7.4v 2200mAh (2S2P) avec PCM ». https://www.generationrobots.com/fr/403175-batterie-li-ion-74v-2200mah-2s2p-avec-pcm.html (17 octobre 2019).

« Capteur de réflectance infrarouge Grove v1.2 - Génération Robots ». https://www.generationrobots.com/fr/402799-grove-infrared-reflective-sensor-v12.html (17 octobre 2019).

« Generationrobots-Lab/MARK ». *GitHub*. https://github.com/generationrobots-lab/MARK (17 octobre 2019).

« Joystick de pouce Grove COM90133P Seeedstudio ». https://www.generationrobots.com/fr/401881-joystick-de-pouce-grove.html (17 octobre 2019).

« Robot Arduino M.A.R.K. pour l'éducation ». https://www.generationrobots.com/fr/403325-robot-arduino-mark-pour-l-education.html?utm_source=Doofinder&utm_medium=Doofinder&utm_campaign=Doofinder (17 octobre 2019).

« Servomoteur standard 180° FS5109M ». https://www.generationrobots.com/fr/403267-servomoteur-standard-180-fs5109m.html (17 octobre 2019).

« Shield de connexion Grove Mega Shield ». https://www.generationrobots.com/fr/401591-shield-de-connexion-cartes-arduino-mega.html (17 octobre 2019).

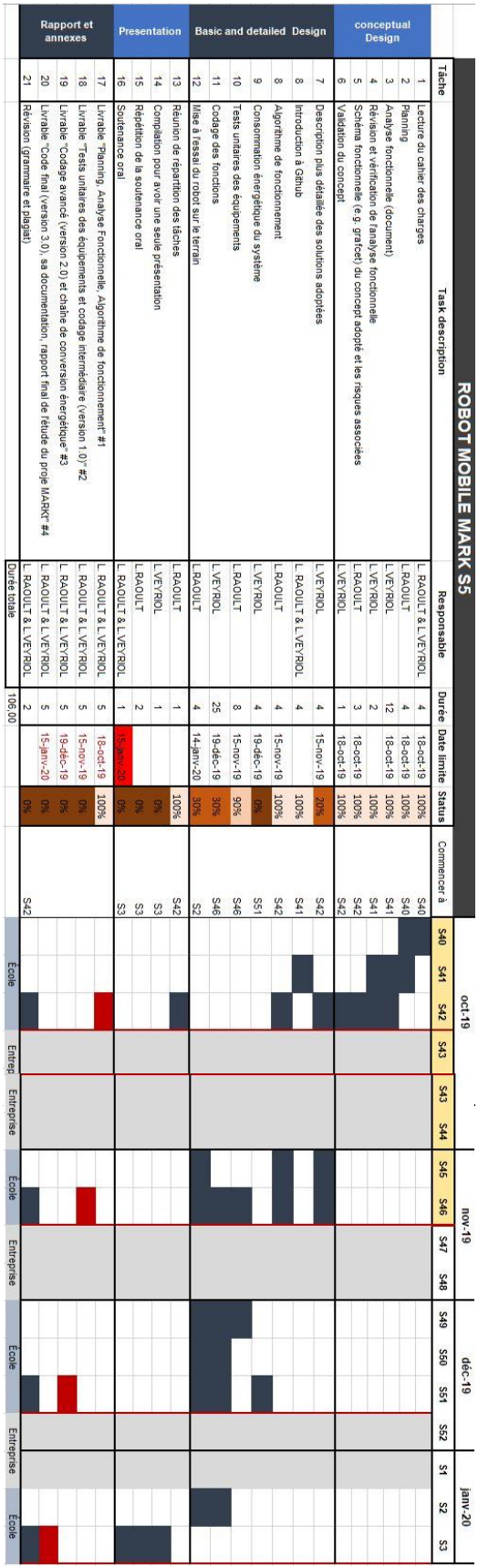
« Télémètre à ultrasons Grove SEN10737P ». https://www.generationrobots.com/fr/401817-grove-telemetre-ultrason.html (17 octobre 2019).

« LOCODUINO - Les interruptions (1) ». https://www.locoduino.org/spip.php?article64 (17 novembre 2019).

« PinMap2560big.png (1257×1498) ». https://www.arduino.cc/en/uploads/Hacking/PinMap2560big.png (17 novembre 2019).

Annexes

Annexe 1 : Planning



Annexe 2 : code intermédiaire (au 15 Novembre 2019)

```
//Ajout des fichiers sources (provenance des fonctions)
#include "Ultrasonic.h"
#include "Grove_LED_Bar.h"
#include "rgb_lcd.h"
#include "Grove_I2C_Motor_Driver.h"
#include "Encoder.h"
#include "SparkFunLSM6DS3.h"
//*****Définition des équipements et déclaration de leurs variables associées*****
//Ultrasons
Ultrasonic us_front(8);
Ultrasonic us_right(10);
Ultrasonic us_left(2);
volatile int dist_us_right, dist_us_left, dist_us_front;
volatile int mode;
volatile  int ecran = 0;
volatile  int start = 0;
volatile int tour_roue_droite = 0;
volatile int tour_roue_gauche = 0;
volatile int tempo = 0;
volatile int test = 0 ;
volatile int compt_dist ;

//Encoder
Encoder knobLeft(18, 29);
Encoder knobRight(19, 27);
long newLeft, newRight;
long positionLeft  = -999;
long positionRight = -999;

//Afficheur LCD
rgb_lcd lcd;
```

```
//*****FONCTIONS D'INTERRUPTION PAR TIMER2*****

ISR(TIMER2_OVF_vect){

  tempo++;  //A CHAQUE TOP ATTEINT --> INCRÉMENTATION DU COMPTEUR = TEMPORISATION ENTRE 2 MESURES

  if(tempo > 30){  // QUAND LE TOP A ÉTÉ ATTEINT 30 fois --> ON AFFECTE LES MESURES DES ULTRASONS À LEUR VARIABLE RESPECTIVE

    //*****CAPTEURS ULTRASONS*****

    dist_us_front = us_front.MeasureInCentimeters(); //déclaration d'une variable globale: prend la valeur de la mesure en cm de l'US de front

    dist_us_left = us_left.MeasureInCentimeters();
//..... de gauche

    dist_us_right = us_right.MeasureInCentimeters();
//..... de droite

    tempo = 0;  //Remise à 0 du compteur

  }

}

//*****Configuration du Timer 2*****

void configTimer2(){

  TCCR2A = 0;           //choix mode: WGM2 = 0, WGM1 = 0, WGM0 = 0 -->  0 = mode normal

  TCCR2B = (1<<CS22)+(1<<CS21);          //CHOIX DU PRÉDIVISEUR = 256

  TIMSK2 |= (1<<TOIE2);       //TOIE2: "Timer/Counter2 Overflow Interrupt Enable" --> Rend possible l'interrution de débordement

}


void setup() {

  pinMode(A2,INPUT);  //DÉCLARATION DE LA PIN ANALOGIQUE "A2" EN ENTRÉE --> appuie sur JOYSTICK

  pinMode(A3,INPUT);  //DÉCLARATION DE LA PIN ANALOGIQUE "A3" EN ENTRÉE --> mouvement JOYSTICK

  lcd.begin(16, 2);   //INITIALISATION DE LA TAILLE DE L'AFFICHEUR

  configTimer2();   //APPEL DE LA FONCTION "CONFIGTIMER2"

  sei();          //AUTORISE LES INTERRUPTIONS A SE DÉLENCHER

}
```

```
void loop() {
//------------------- DECLARATION DE VARIABLE --------------------------------
  long newLeft, newRight;
  float vit_max_droite;
  float test_vit_max_droite;
  float vit_max_gauche;
  float test_vit_max_gauche;
  float temps;
  const byte infrared = 6;


//------------------- AFFICHAGE ECRAN L'ECRAN --------------------------------
  if (analogRead(A3) > 700 ){ //GESTION CHANGEMENT D'ÉCRAN QUAND JOYSTICK
VIRE A GAUCHE
    ecran = ecran+1;
    lcd.clear();
    _delay_ms(200);
  }


   if (analogRead(A3) < 400 ){  //GESTION CHANGEMENT D'ÉCRAN QUAND JOYSTICK
VIRE A DROITE
    ecran = ecran-1;
    lcd.clear();
    _delay_ms(200);
  }
  switch (ecran){
    case 0:
      lcd.setCursor(0,0);
      lcd.print("ETAT DU ROBOT :");
      lcd.setCursor(0,1);
      if(start == 1) lcd.print("Marche");
      else lcd.print("Arret");
      break;
    case 1:
```

```
  lcd.setCursor(0,0);
  lcd.print("VITESSE MAX :");
  lcd.setCursor(0,1);
  lcd.print("R :");
  lcd.print(vit_max_droite);
  lcd.setCursor(9,1);
  lcd.print("L :");
  lcd.print(vit_max_gauche);
  break;
 case 2:
  lcd.setCursor(0,0);
  lcd.print("NBR TOUR ROUE :");
  lcd.setCursor(0,1);
  lcd.print("R :");
  lcd.print(positionLeft);
  lcd.setCursor(9,1);
  lcd.print("L :");
  lcd.print(positionRight);
  break;
 case 3:
  lcd.setCursor(0,0);
  lcd.print("Distance parcourue:");
  lcd.setCursor(0,1);
  lcd.print((positionLeft+positionRight)*19 /2);
  break;
 case 4:
  lcd.setCursor(6,0);
  lcd.print(dist_us_front);
  lcd.setCursor(0,1);
  lcd.print(dist_us_left);
  lcd.setCursor(14,1);
  lcd.print(dist_us_right);
```

```
      break;
    case 5:
      lcd.setCursor(0,0);
      lcd.print("COMPT DIST < 20");
      lcd.setCursor(0,1);
      lcd.print(compt_dist);
      break;
    case 6 :
      lcd.setCursor(0,0);
      lcd.print("ENE CONS");
      break;
    default :
      ecran = 0 ;
      break;
  }

//-------------------- MARCHE / ARRET DU ROBOT ------------------------
  if (analogRead(A2) == 1023 & start == 0){
    start=1;
    _delay_ms(200);
  }
  if (analogRead(A2) == 1023 & start == 1){
    start=0;
    _delay_ms(200);
  }
  if (start==1){
    if( (dist_us_right <= 50) && (dist_us_right >= 40)  ){//ok
      mode = 0;
    }
    if( (dist_us_right > 50) ){
      mode = 1;
    }
```

```
if(dist_us_right < 40){
  mode = 2;
}
if(dist_us_front < 80){
  mode = 3;
}
if( (dist_us_right > 250) && (dist_us_front > 150) ){
    mode = 4;
}

switch(mode){
case 0:
    Motor.speed(MOTOR1,100);  //vitesse (-100 ; +100)
    Motor.speed(MOTOR2,100);
    break;
case 1:
    Motor.speed(MOTOR1,100);  //MOTOR1 coté gauche
    Motor.speed(MOTOR2,75);  //MOTOR2 coté droite
    break;
case 2:
    Motor.speed(MOTOR1,90);
    Motor.speed(MOTOR2,100);
    break;
case 3:
    Motor.speed(MOTOR1,-100);  //MOTOR1 coté gauche
    Motor.speed(MOTOR2,100);  //MOTOR2 coté droite
    break;
case 4:
    Motor.speed(MOTOR1,100);
    Motor.speed(MOTOR2,-100);
    break;
default:
```

```
        Motor.stop(MOTOR1);
        Motor.stop(MOTOR2);
        break;
    }
  }else{
      Motor.stop(MOTOR1);
      Motor.stop(MOTOR2);
  }

//DETECTION US + 1er virage

//-------------------- Nombre de tour de chaque Roue --------------------
  newLeft = knobLeft.read();
  newRight = knobRight.read();
  if (newLeft != positionLeft || newRight != positionRight) {
    positionLeft = newLeft / 3600;
    positionRight = newRight / 3600;
  }
/*
//-------------------- Calcul Vitesse MAX de chaque roue ----------------

  if (test_vit_max_gauche > vit_max_gauche){
    vit_max_gauche = test_vit_max_gauche;
  }

  if (test_vit_max_droite > vit_max_droite){
    vit_max_droite = test_vit_max_droite;
  }



//-------------------- Compter distance < 20 cm ------------------------
```

```
*/


  if ( ((dist_us_front < 20) || (dist_us_right < 20) || (dist_us_left < 20)) and test == 0 ){  // or
dist_us_left < 20 or dist_us_right < 20)
   compt_dist = compt_dist + 1;                                                    // or dist_us_front <
20
   test = 1;
  }
  else{
    if (dist_us_front > 20 and test == 1){   //dist_us_left < 20 and dist_us_right < 20 and
     test = 0;
    }
  }
```