TROISIÈME ANNÉE DE BUT

GÉNIE ÉLECTRIQUE ET

INFORMATIQUE INDUSTRIELLE

IUT D'ÉVRY VAL D'ESSONNE

RAPPORT SAE 5

ROBOT POB

Année 2024 - 2025

Préparé par :

Aubin TOURAIS

Melchior DIEGUEZ

Nicolas FELLAH

Enseignant :

Didier CZAPLEWSKI

Noémi LANCIOTTI

SOMMAIRE

INTRODUCTION

Lors de cette SAE, nous avons fait la découverte des robots POB. L'objectif étant de les configurer pour qu'ils puissent participer à une compétition de combat de sumo, le but est de pousser le robot adverse hors du dohyo sans tomber de celui-ci. Par la suite, nous montrerons aux BUT 1 le fonctionnement du robot pour qu'ils le programment et fassent la compétition.

Les robots POB sont particulièrement adaptés à ce type de projets grâce à leur flexibilité et aux multiples capteurs qu'ils peuvent intégrer. Tout au long de ce projet, nous avons utilisé et configuré différents capteurs afin d'optimiser le comportement du robot en combat, vous retrouverez leur fonctionnement ainsi que leur câblage et le code pour pouvoir les faire fonctionner dans ce rapport. Nous avons mis en œuvre différents capteurs tel qu'un télémètre infrarouge ou bien un capteur de ligne pour détecter le bord du dohyo.

Nous avons également pensé et élaboré plusieurs stratégies permettant de pousser le robot adverse sans que le notre tombe.

Ce rapport vise à donner un aperçu des étapes clés de notre projet, la configuration des capteurs, la programmation du robot, et les résultats obtenus lors des tests. Il peut également servir de guide pour les étudiants de BUT 1, leur permettant de mieux comprendre le fonctionnement du robot.

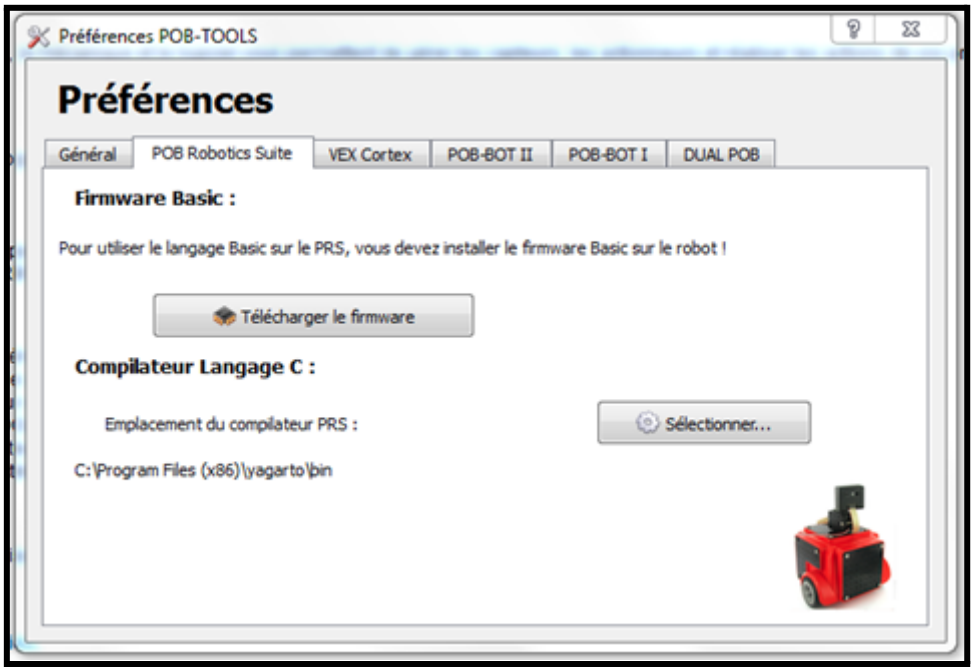
SÉANCE DU 13 SEPTEMBRE 2024

- **Utilisation de POB TOOLS 4 :**

- **Création d'un projet :**

- **Compilation :**

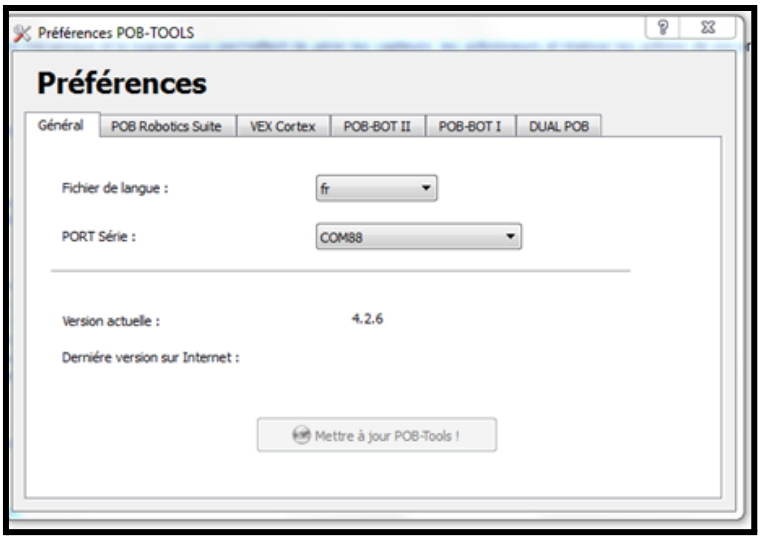
Pour compiler, bien sélectionner la source qui se trouve sur l'image ci-dessous.



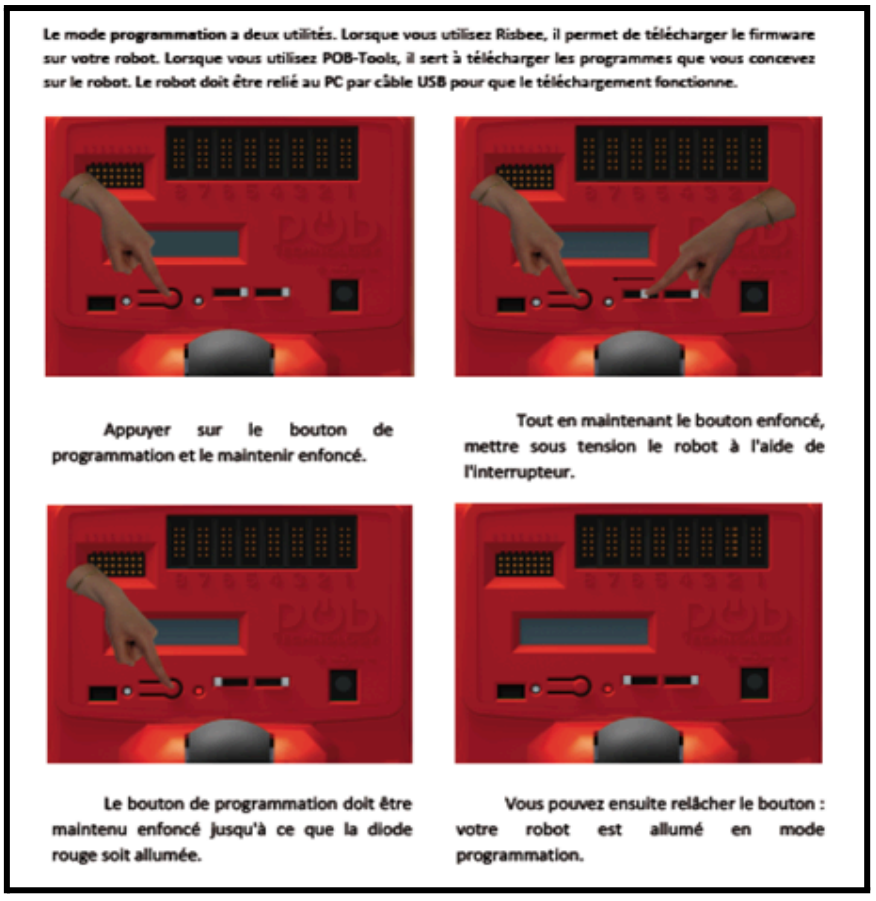
Une fois cela fait, relancer l'application.

- **<u>Téléchargement dans le robot :</u>**

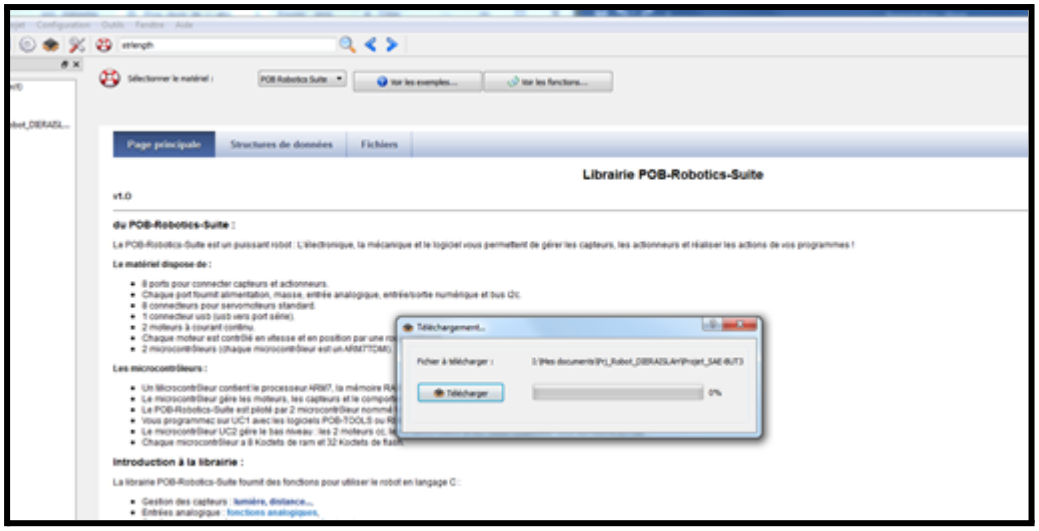
Afin de pouvoir télécharger le programme dans le robot, bien sélectionner le PORT correspondant au branchement du robot sur le PC.



Afin de pouvoir faire le téléchargement, débrancher l'alimentation du robot, puis :



- Brancher l'alimentation au robot.
- Relâcher le bouton.
- Faites le téléchargement comme ci-dessous.



Une fois cela fait dans l'ordre, votre programme est transféré dans le robot en question.

- **QUESTIONS :**

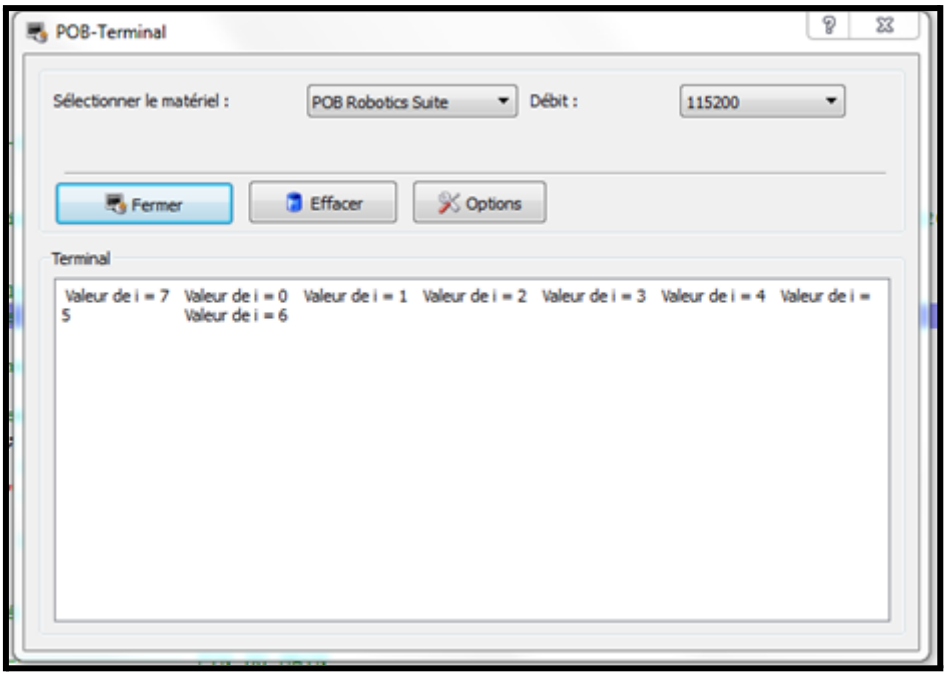
- ***A quoi servent les fonctions : strlength() et DebugPrintf() ?***

La fonction strlength() sert à calculer la longueur d'une chaîne de caractères données.

La fonction DebugPrintf () sert à afficher une chaîne de caractères donnés.

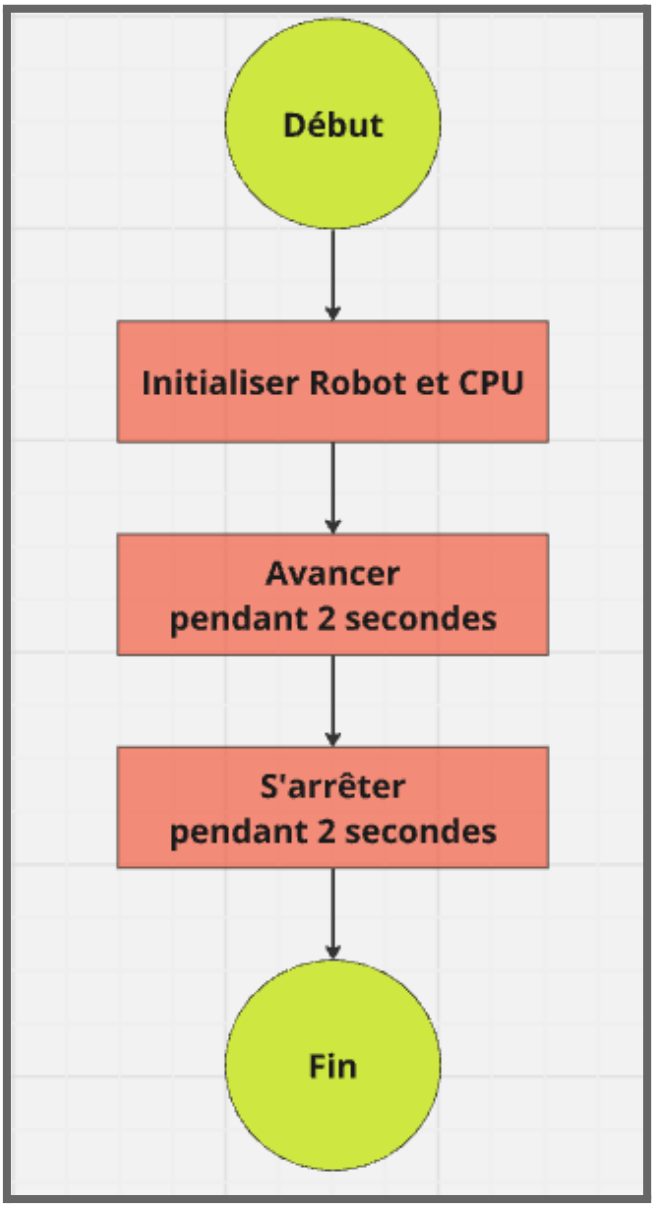
- ***Quelle différence existe-t-il en les fonctions DebugPrintf() et DebugSPrintf() ?***

La fonction DebugPrintf() permet d'afficher sur le terminal dans l'ordinateur.



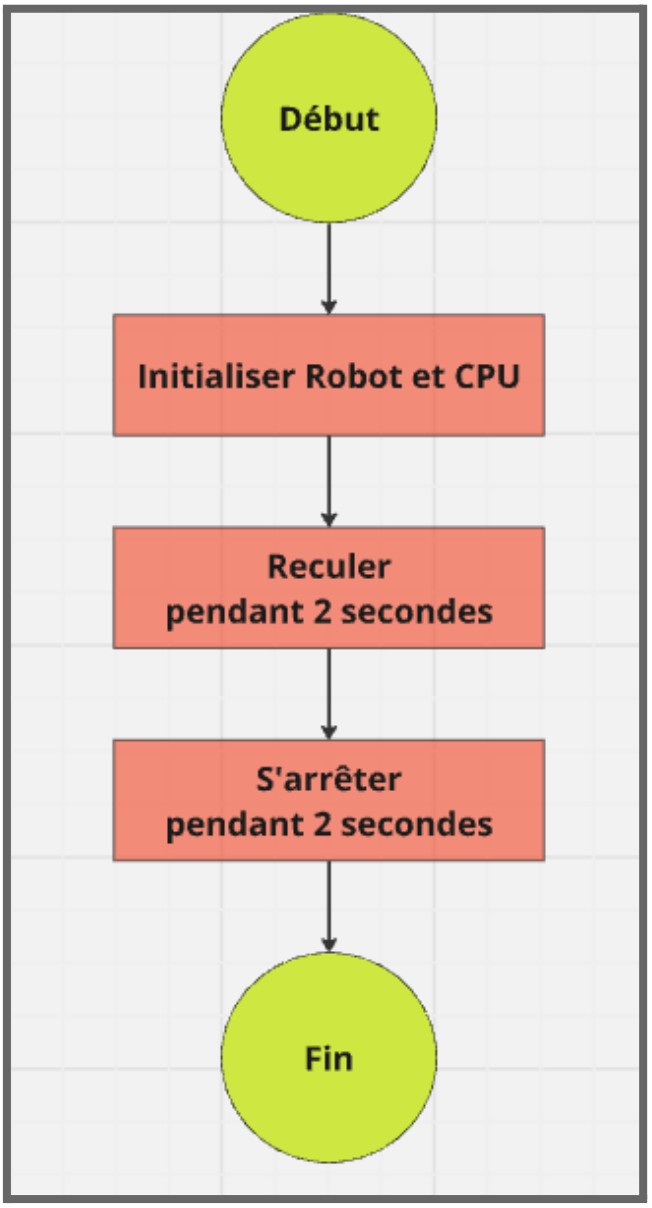
La fonction DebugSPrintf() permet d'afficher sur l'afficheur LCD du robot.

- **Programme pour Avancer :**



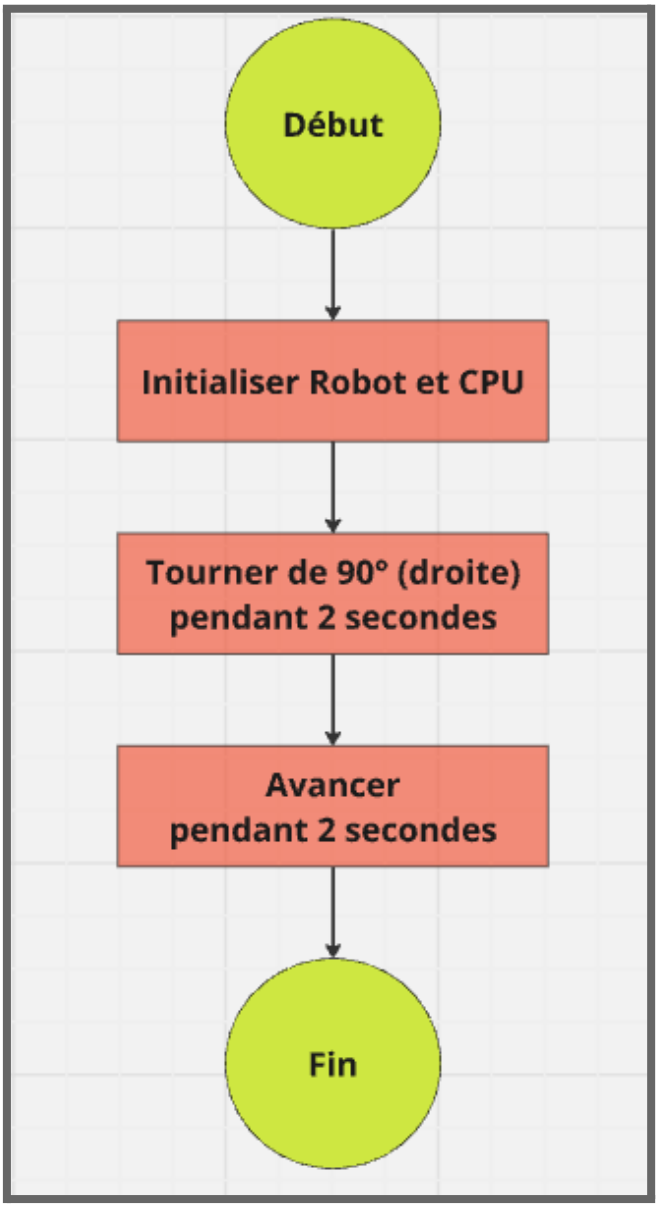
Afin de pouvoir voir le programme qui est associé à l'algorigramme ci-dessus, se référer à l'**Annexe 1**

- **Programme pour Reculer :**



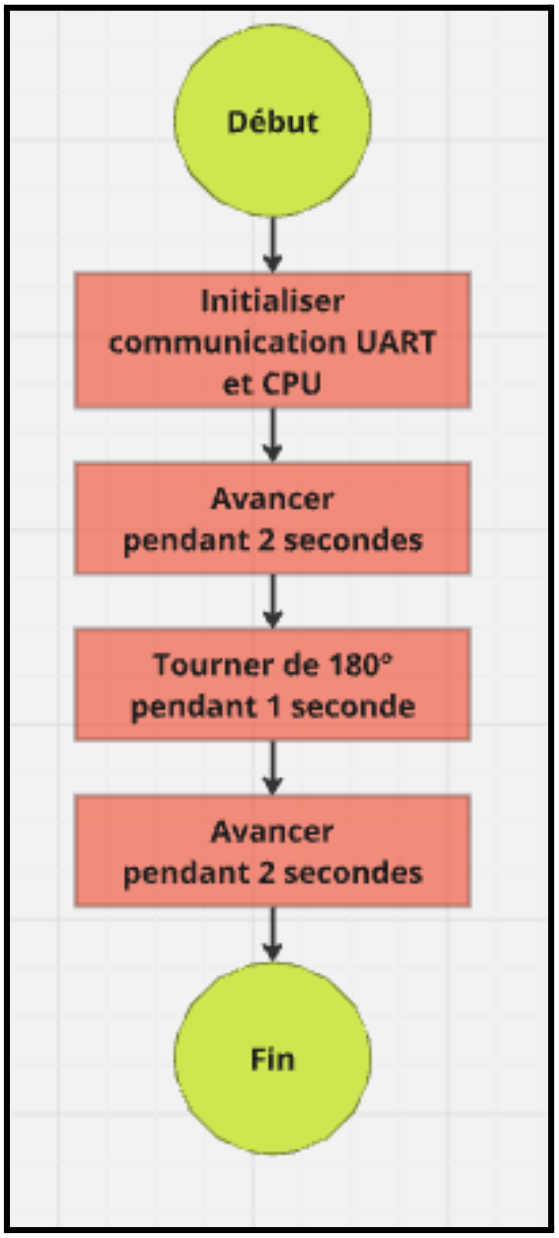
Afin de pouvoir voir le programme qui est associé à l'algorigramme ci-dessus, se référer à l'**Annexe 2**

- **Programme pour Tourner :**



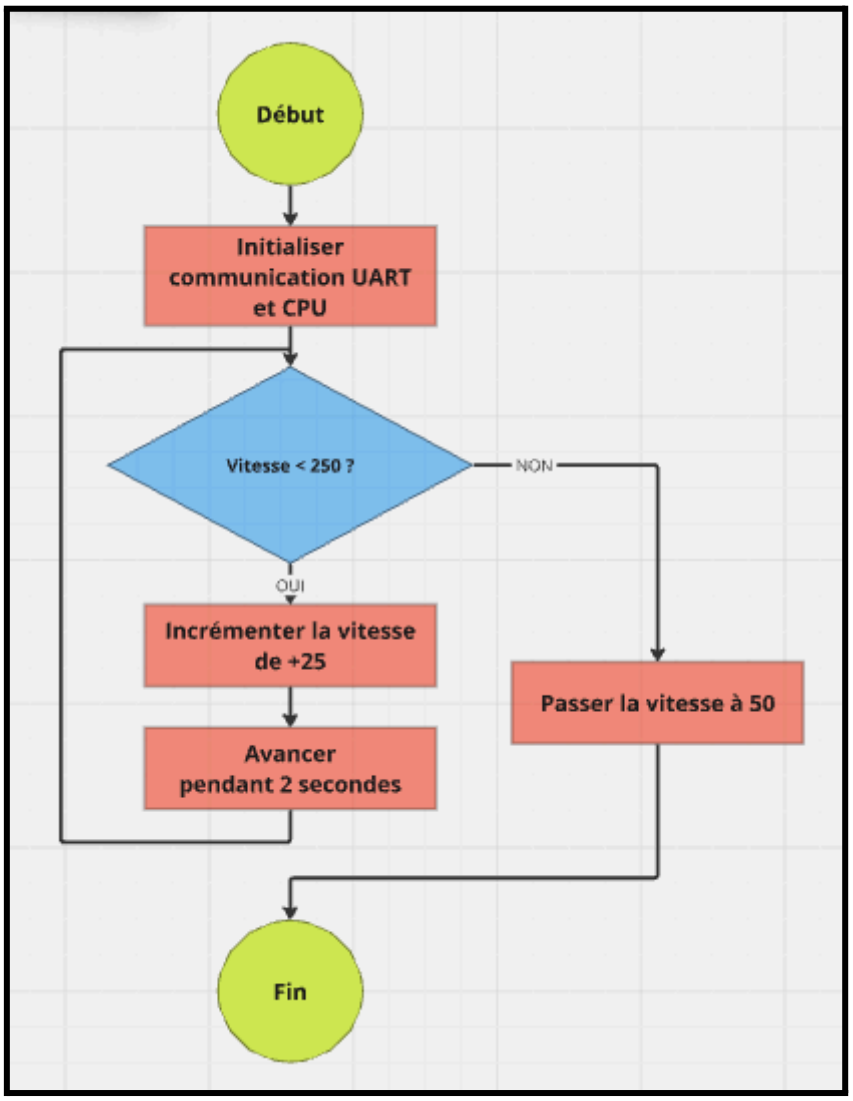
Afin de pouvoir voir le programme qui est associé à l'algorigramme ci-dessus, se référer à l'**Annexe 3**

- **Programme pour Demi-Tour :**



Afin de pouvoir voir le programme qui est associé à l'algorigramme ci-dessus, se référer à l' **Voir Annexe 4**

- **Programme pour Accélération au fur et à mesure :**



Afin de pouvoir voir le programme qui est associé à l'algorigramme ci-dessus, se référer à l'**Annexe 5**

SÉANCE DU 20 SEPTEMBRE 2024

- **Règles du jeu :**

Prérequis :

- Un combat oppose 2 robots sur un Dohyo.
- Le dohyo est noir avec un diamètre de 1540 mm avec un contour blanc sur l'extrémité de 25 mm.
- Le robot a une dimension maximum de 35 cm x 35 cm de côté et est sans limite de hauteur,

Règle :

- Rester dans le Dohyo,
- Le combat commence quand les robots sont mis en route par une personne,
- Une fois activé, les robots doivent démarrer en moins de 5 sec sinon éliminé,
- Si un robot est à 100% en dehors du dohyo il est éliminé,
- Si un robot est retourné (plus sur ses roues) il est éliminé,
- La finale se déroulera en 3 manches gagnantes,
- Les combats sont en 2 manches gagnantes,
- Une manche dure 3 minutes,
- Autorisation d'ajouter des équipements au robot,
- La masse du robot est limité à 2Kg,
- Le robot doit être alimenté électriquement par des piles ou des accumulateurs,
- Le Sumo d'or : Lors des égalités on fait une manche finale où juste toucher la ligne avec les roues du robot est éliminatoire éliminé,
- Les robots sont positionnés face à face,

Qualification :

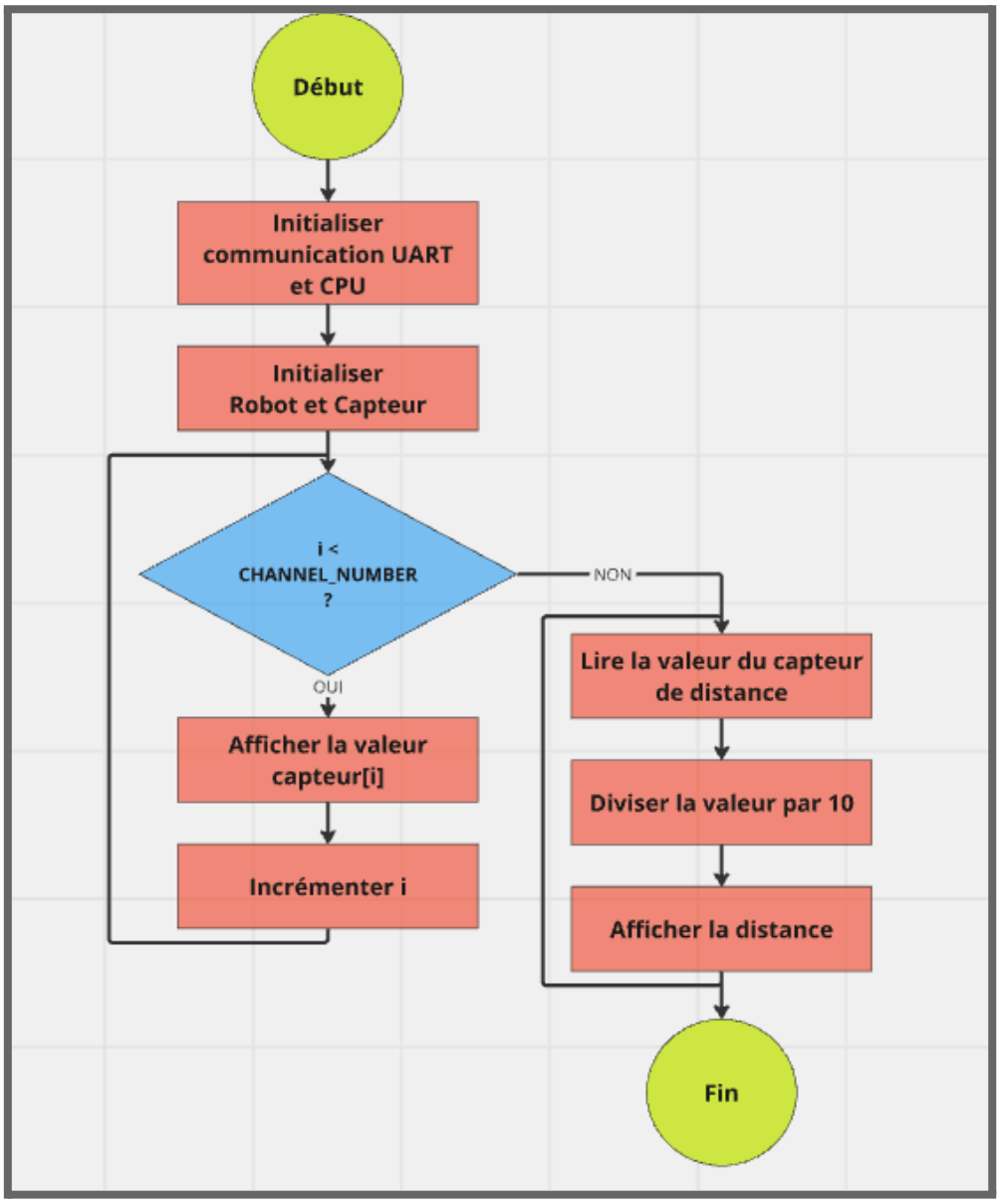
- Tournoi 1vs1 en forme d'arbre classique,
- Gagner 2 manche qualifie pour le prochain match,
- Si égalité sumo d'or,
- S'il y a le temps :
  « Looser bracket » (les perdant du tournoi s'affrontent, et le gagnant de tous les matches affronte le gagnant de toutes les manches du tournoi initiale),
- Perdre un match le robot rentre dans le looser bracket,
- Perdre 2 match = fin du tournoi,

- **Listes des composants :**

- Capteur de distance (infrarouge)
- Capteur pour différencier le noir et blanc
- Bouton de fin de course

- **<u>Programme pour tester le capteur d'infrarouge de distance :</u>**

Branchement du capteur de distance (infrarouge) sur le port 1 du robot POB.

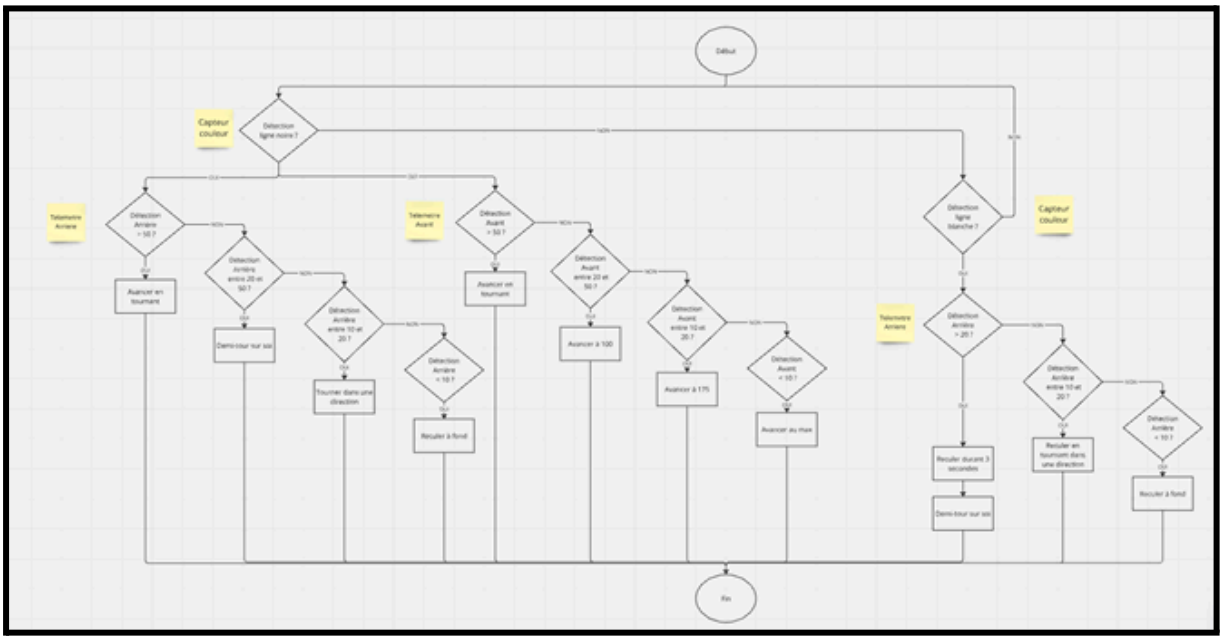


Afin de pouvoir voir le programme qui est associé à l'algorigramme ci-dessus, se référer à l'**<u>Annexe 6</u>**

Value /= 10

Permet de convertir la distance mesurée en cm (est de base en mm).

- **Algorigramme de stratégie :**



- **Stratégie d'exécution**

Détection du terrain :

- Capteur fin de course pour savoir si on est en dehors du terrain.

- Capteur de couleur pour différencier le noir et le blanc pour ne pas sortir du terrain.

Détection de l'adversaire :

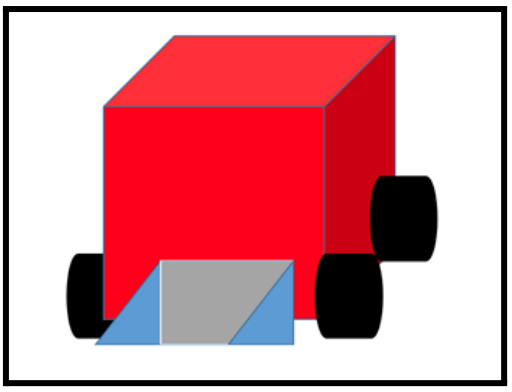
- Capteur de distance à l'avant et à l'arrière afin de savoir si l'adversaire se trouve à une certaine distance pour permettre de prendre de l'élan ou d'accélérer.

- Capteur de distance à l'arrière afin de savoir si l'adversaire se trouve à une certaine distance pour permettre de l'esquiver si le robot est proche du bord du terrain.

Stratégie de Combat :

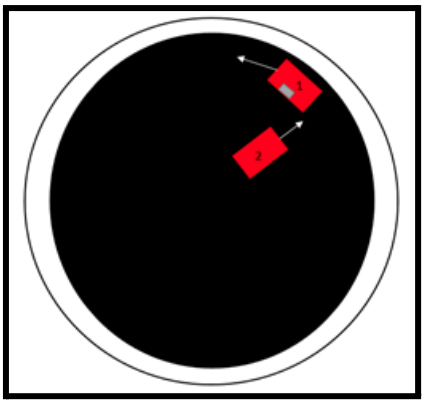
- ***Mode « Fantôme Widget » :***

Équerres à l'avant du robot pour lever l'autre robot pour que ses roues soient en l'air et qu'il ne puisse plus avancer.



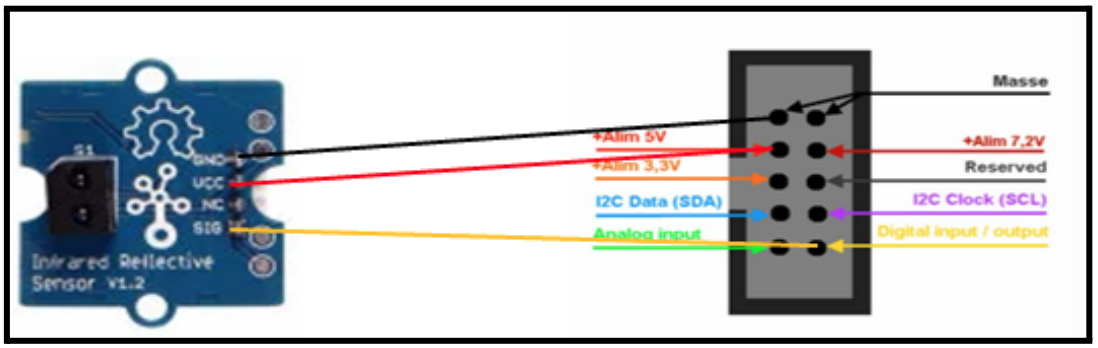
- ***Mode « Pousse Pousse Melchior » :***

Longer la ligne blanche tout le long avec le capteur de distance vers l'intérieur de la zone, attendre que l'adversaire nous arrive dessus, si la détection est inférieur à un certain seuil, alors le robot ennemi est proche, on recule afin de le laisser s'approcher de la ligne blanche. Lui, il s'arrêta car détection ligne blanche, puis nous arrivons à pleine vitesse par derrière.



SÉANCE DU 27 SEPTEMBRE 2024

Il y a 4 broche sur le capteur :
- VCC connecté au 5V
-GND pour la masse
- NC pour non connecté
- SIG pour signal branché sur la broche digital input du robot



- **Programme détéction de couleur + telemetre**

Afin de pouvoir voir le programme qui est associé, se référer à l'**Annexe 7**

- **Programme avec deux détéction de couleur + telemetre avant**

Afin de pouvoir voir le programme qui est associé, se référer à l'**Annexe 8**

- **Information sur le programme en question :**

LigneAvant = GetDigitalIO(5);
LigneArriere = GetDigitalIO(4);
Telemetre = GetDistanceSensor(1);
Initialise les capteurs sur les PORT 1 (Distance sensor), PORT 4 et PORT 5 (Infrared Reflective sensor)

SensorType * sensors;
sensors = InitRobot();
Initialise le capteur de distance infrarouge.

UInt32 Telemetre;
Telemetre = GetDistanceSensor(8);
Permet de relever en valeur analogique la distance du capteur de télémètre.

UInt32 Ligne;
Ligne = GetDigitalIO(5);
Permet de relever en valeur numérique (0,1) l'état du capteur de ligne infrarouge.

---

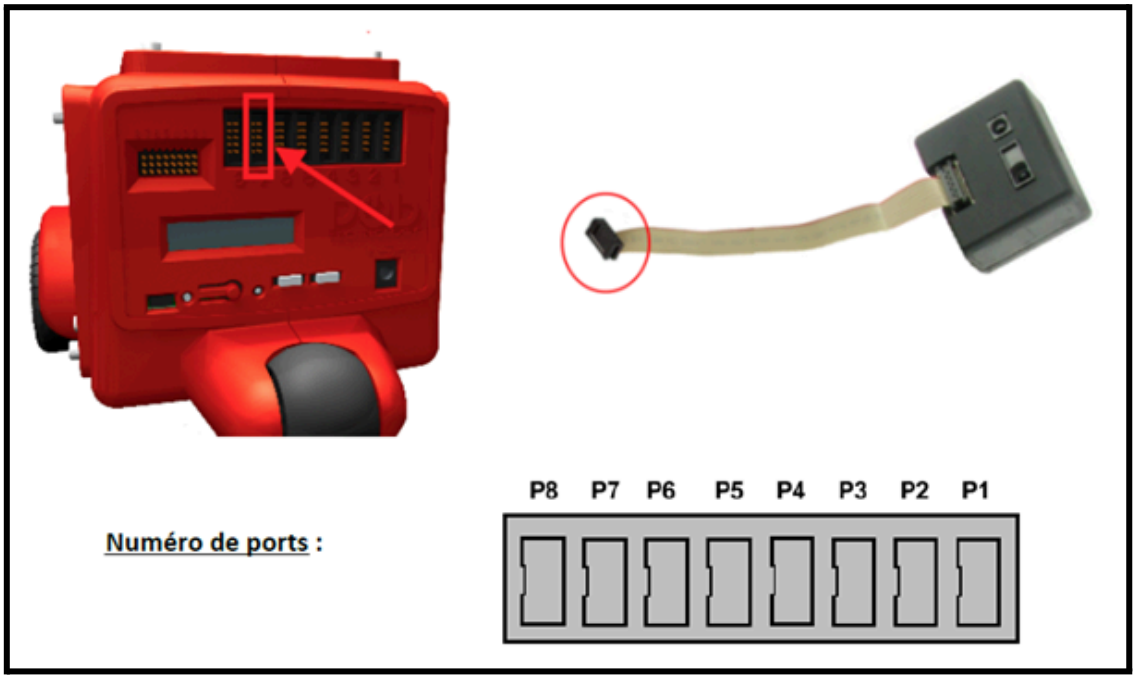
int SGauche = 200;
int SDroite = 200;
SetMotor(-SGauche, -SDroite);
Permet de faire reculer les roues du robot.

SetMotor(SGauche, SDroite);
Permet de faire avancer les roues du robot

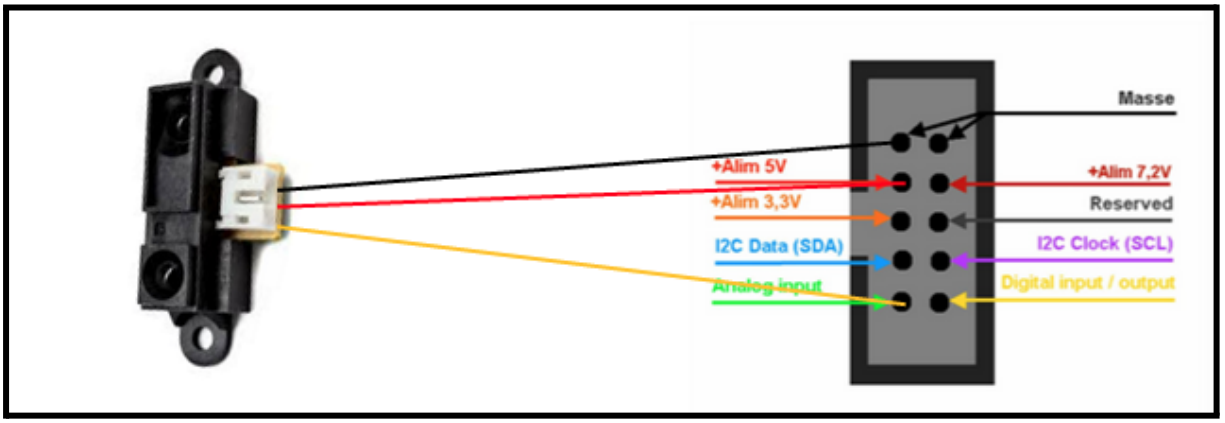
Lorsque l'on voit la couleur noir, notre capteur lit la valeur "0".
Néanmoins, s'il voit la couleur blanche, il lira la valeur "1".

SÉANCE DU 4 OCTOBRE 2024

On a utilisé un nouveau télémètre qui est câblé comme ci-dessous.

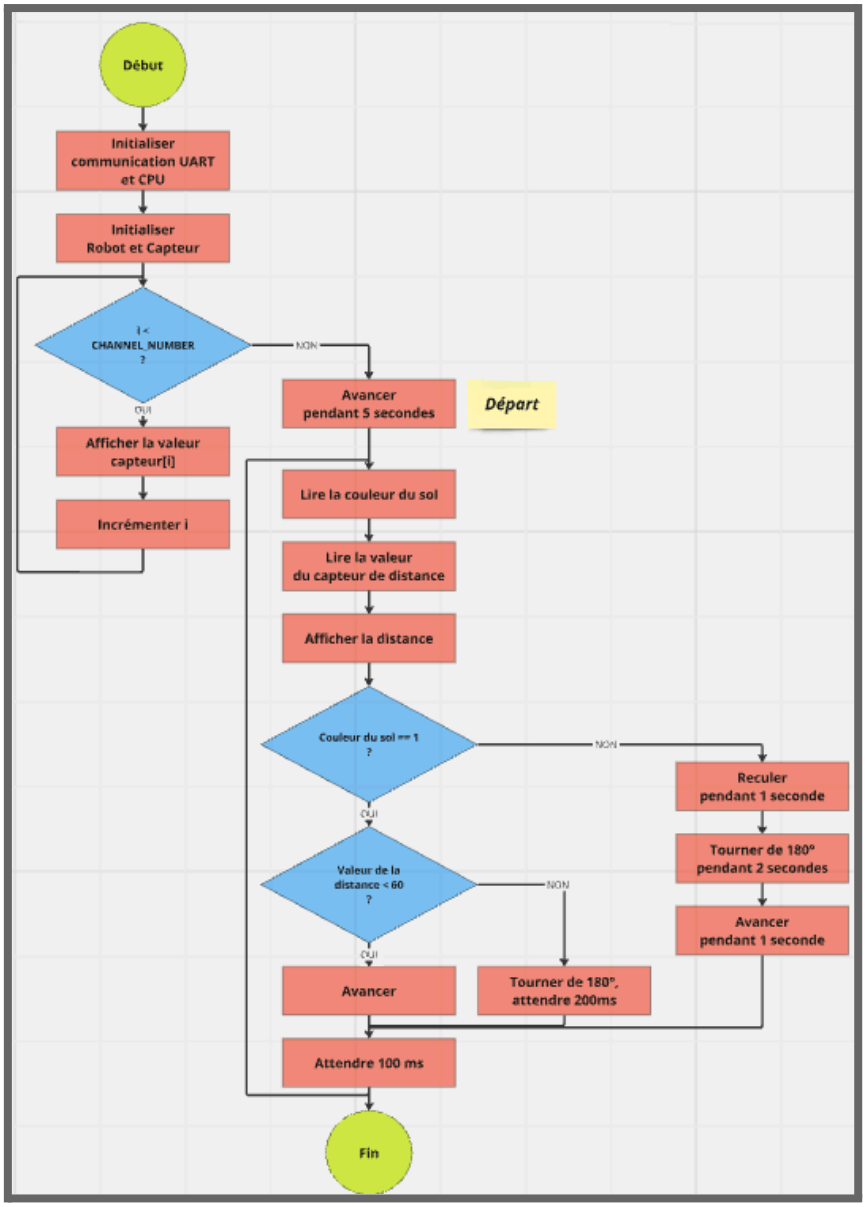


Il y a 3 broche sur le capteur :
-VCC connecté au 5V
-GND pour la masse
-Le cable orange est branché sur la broche analog input du robot



Le programme que nous avons testé fait avancer le robot en ligne droite.
Au moment de la détection de la ligne blanche, le robot fait demi-tour et repart dans le sens inverse et ce indéfiniment le temps de trouver un autre robot pour pouvoir le pousser.

Celui-ci possède le même code que celui que nous avions au début (relié sur les ports de droite)



Afin de pouvoir voir le programme qui est associé à l'algorigramme ci-dessus, se référer à l'**Voir Annexe 9**

Dans ce code, nous faisons l'union entre le télémètre et l'infrarouge (suivit de ligne) afin de pouvoir faire un test sur l'arène.

Nous avons réfléchi à la stratégie de faire en sorte que notre robot tourne sur lui-même jusqu'à la détection d'un robot adversaire, une fois la détection effectuée, le robot avance sur l'adversaire.

L'infrarouge permet de savoir où se trouve les limites de la zone, quand nous voyons du blanc, il s'agit de la bordure, nous faisons un demi-tour avec une marche arrière afin de pouvoir mieux repartir. Le robot repartira et recherchera à nouveau l'adversaire.

Nous avons également rajouté une plaque à l'avant afin de pouvoir permettre de protéger notre télémètre qui se trouve à l'avant du robot. Cette rampe permet également d'élever le robot adverse en passant par en dessous.

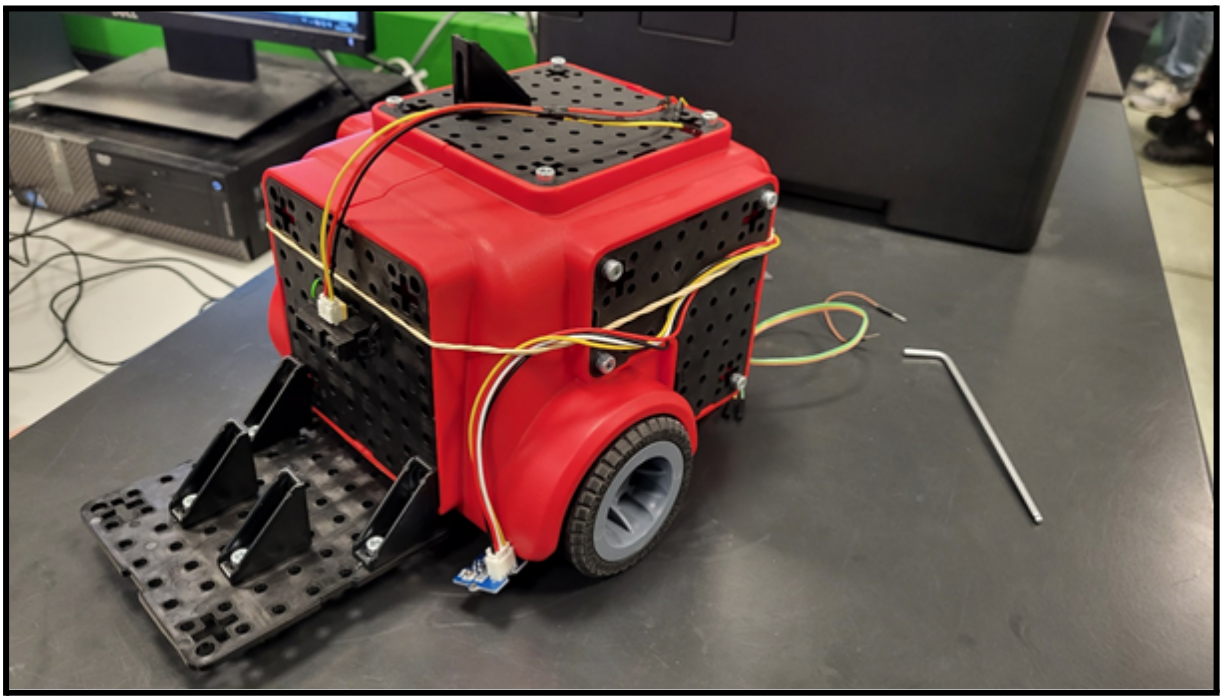
Afin de pouvoir atteindre ce résultat, nous avons rencontré de nombreux problèmes :

- Télémètre qui ne détecte pas -> se trouvait au-dessus du robot, donc changement de position.
- Robot qui se bloque -> nous avions vu qu'il fallait mettre les fonctions d'initialisation des capteurs dans la boucle while(1)

Pour la prochaine fois, nous ferons le test final, en ajoutant seulement un autre capteur infrarouge (ligne) que l'on mettra soit de l'autre côté ou à l'arrière, voici les cas possibles :

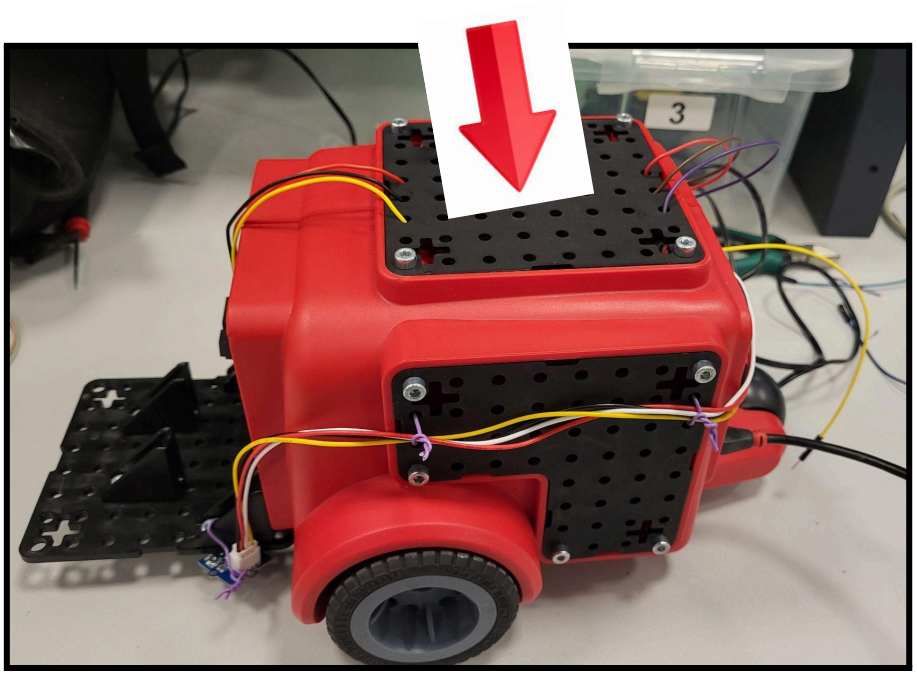
- Sur l'autre côté -> Permettre d'avoir 2 capteurs à l'avant pour éviter de tomber de l'autre côté.
- A l'arrière –> Au cas où notre robot se fait pousser par l'avant, et ainsi choisir une autre solution, comme tourner sur un côté.

Nous verrons également pour terminer le programme en utilisant potentiellement un servomoteur, que l'on utilisera pour faire tourner en permanence notre télémètre et ainsi pouvoir faire avancer notre robot en continu.
Il changera de direction seulement quand il détectera un robot dans le champ de vu du télémètre.



SÉANCE DU 11 OCTOBRE 2024

Cette séance a amélioré notre câblage. Nous avons fait passer les fils à l'intérieur des caches noirs comme ci dessous :



Ceci nous evite que des fils soient detacher pendant le combat les autres fils ne pouvant pas passer a linterieur, nous les avons er solidement a la plaque noir. Nous avons fait de même avec le capteur infrarouge pour qu'il ne puisse pas être déplacé lors du combat.

Nous avons fait des tests contre certains robots, afin de voir si notre programme et stratégie était opérationnel.

Résultat :

- Départ du robot : on avance pendant 5 secondes = OK
- Le robot tourne sur lui-même afin de pouvoir détecter l'adversaire. = OK
- Détection de l'adversaire parfaite, il se dirige vers lui = OK
- Si l'adversaire s'enfuit ou est décalé, notre robot continue sa recherche puis retourne à l'attaque = OK
- Avec la rampe à l'avant il réussit à soulever des robots adversaire, les décaler en tournant car il détecte pas en étant de côté = OK
- Le robot peut donc pousser de l'avant, ou forcer sur le côté en tournant = OK +
- Si un robot est de dos, ils avancent, nous le détectons, arrivons par derrière et ainsi forçant le robot adversaire à sortie du terrain étant donné que nous soulevons sa roue arrière = OK
- Notre robot fait parfaitement le demi tour et ainsi ne tombe jamais même si il pousse, car bord de terrain prioritaire.

Après un total de 13 duels, notre robot en a gagné 10, avec 2 défaites liés à des combats face à deux robots (test d'affrontement de 3 robots sur le dohyo) et une défaite, car le capteur a frotté contre le sol donc décaler = trouver un moyen de le maintenir en continue dans la même position.).

Pour résumer, la stratégie souhaitée fonctionne très bien, nous pourrions rajouter un deuxième capteur pour la couleur afin d'en mettre un à gauche et un à droite, afin d'être sûr de ne jamais tomber et toujours détecter la bordure.
De même pour un capteur de distance, nous pouvons ainsi détecter encore plus facilement l'adversaire en évitant de tourner en continu.

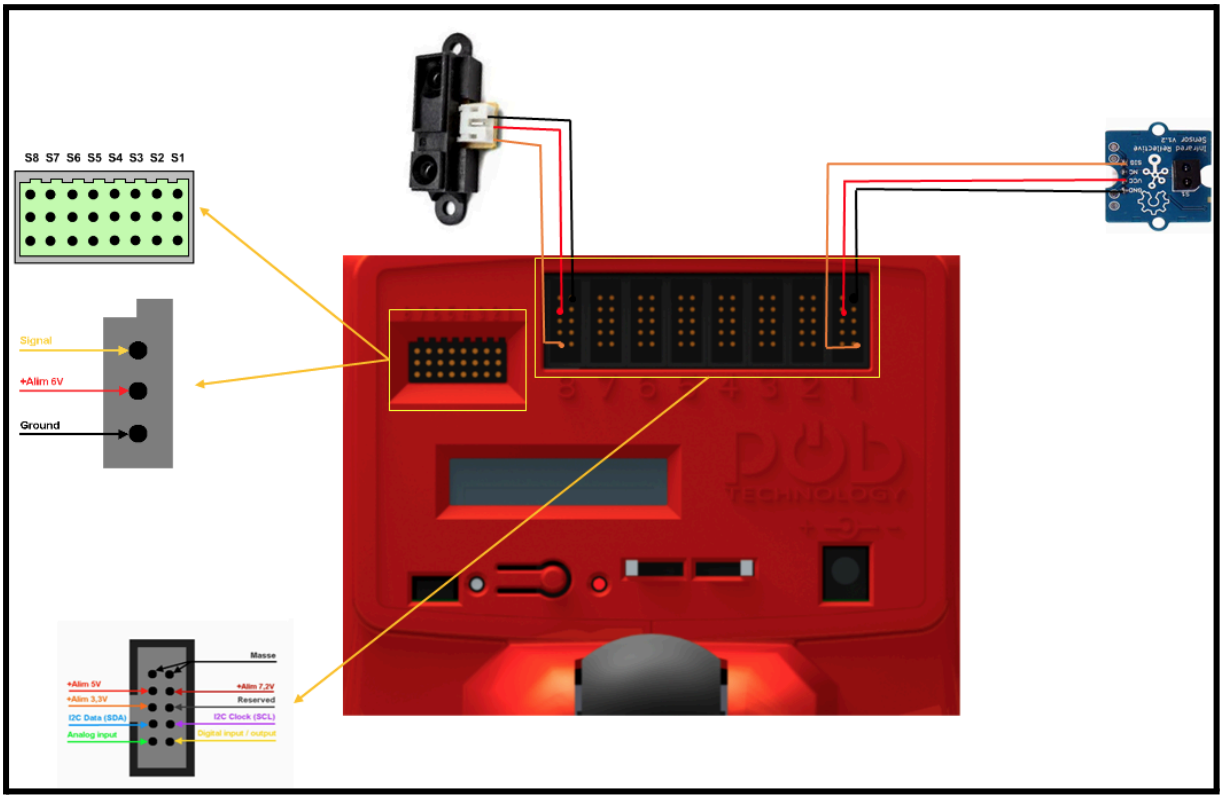
Le test avec le servo moteur aurait pu être concluant, afin d'y mettre le capteur de distance au dessus de celui-ci et ainsi faire le fonctionnement suivant :

- Avancer tout en faisant tourner le servo moteur en continue
- Lire la valeur du capteur de distance
- Si lecture d'un robot adverse, prendre en compte la rotation du servomoteur et ainsi faire tourner le robot à la position lue.
- Résultat de ces bonus = gain de temps important et approche sur l'adversaire plus important = on obtient un avantage conséquent sur l'adversaire qui lui tourne ou s'arrête.

On a essayé de coder le capteur à ultrasons, on a dû utiliser une carte arduino avec un shield. On peut lire les valeurs du capteur sur la carte arduino mais on arrive pas à les lire/ recevoir dans le robot.

SÉANCE DU 18 OCTOBRE 2024

Voici le schéma de câblage final.



Il y a 4 broche sur le capteur :

- VCC connecté au 5V
-GND pour la masse
- NC pour non connecté
- SIG pour signal branché sur la broche digital input du robot

Après un combat acharné contre le robot avantagé de l'équipe de M.Mochet, notre capteur de ligne s'est fait tragiquement écraser par la masse informe que représentait le Robot de l'équipe de M.Mochet et ne fonctionne plus après le combat.

Nous avons fait de plus des combats, afin de tester face aux autres robots.

CONCLUSION

En conclusion, durant cette SAE nous avons pu découvrir les possibilités et fonctionnalités offertes par les robots POB. La configuration des capteurs nous a permis d'élaborer des stratégies.

Nous avons pu mettre en œuvre différents capteurs qui ont fonctionné avec brio tel que le télémètre pour la distance ou encore le capteur de ligne. Nous avons essayé de mettre en œuvre le capteur ultrason mais malheureusement, cela fut un échec.

Grâce à ce projet, nous avons préparé une base pour les étudiants de BUT 1 pour les aider à comprendre le fonctionnement de ce robot et également pour les différents programmes qu'ils auront à développer.

Ce rapport, en plus de documenter notre progression, pourra les guider dans leur propre démarche pour programmer et améliorer les robots POB, en vue de la compétition.

Annexes

- ## Annexe 1 : Programme ‘AVANCER’

```
#include <pobRoboticSuite.h>

int SGauche = 200;
int SDroite = 200;

int main(void){
        InitCpu();
        InitRobot(); // Initialise le robot

        SetMotor(SGauche, SDroite); // Avancer
        WaitMs(2000);       // Pendant 2 secondes

        SetMotor(0, 0); // S’arreter
        WaitMs(2000);       // Pendant 2 secondes

        return 0;
}
```

- ## Annexe 2 : Programme ‘RECULER’

```
#include <pobRoboticSuite.h>

int SGauche = -150;
int SDroite = -150;

int main(void){
        InitCpu();
        InitRobot();

        SetMotor(SGauche, SDroite);      // Reculer
        WaitMs(2000);       // Pendant 2 secondes

        SetMotor(0, 0); // S’arrêter
        WaitMs(2000);       // Pendant 2 secondes

        StopMotor();

        return 0;
}
```

- **Annexe 3 : Programme ‘TOURNER’**

```
#include <pobRoboticSuite.h>

int SGauche = 200;
int SDroite = 200;

int main(void){
	InitCpu();
	InitRobot(); // Initialise le robot

	DoRotation(90, 250);	// tourner à droite (-90 pour à gauche)
	WaitMs(2000);	// Pendant 2 secondes

	SetMotor(SGauche, SDroite); // Avancer
	WaitMs(2000);	// Pendant 2 secondes

	return 0;
}
```

- **Annexe 4 : Programme ‘DEMI-TOUR’**

```
#include <pobRoboticSuite.h>

int SGauche = 100;
int SDroite = 100;

int main(void){
	InitCpu();
	InitRobot();

	SetMotor(SGauche, SDroite); // Avancer
	WaitMs(2000);	// Pendant 2 secondes

	DoRotation(180, 250);	// tourner à droite
	WaitMs(1000);	// Pendant 1 secondes

	SetMotor(SGauche, SDroite); // Avancer
	WaitMs(2000);	// Pendant 2 secondes

	return 0;
}
```

- **Annexe 5 : Programme ‘AVANCER AU FUR ET À MESURE’**

```
#include <pobRoboticSuite.h>

int SGauche = 50;
int SDroite = 50;
int Up = 0;

int main(void){
	InitCpu();
	InitRobot(); // Initialise le robot

	while(1) {
		for(; SGauche<250; Up=Up+25) {
			SGauche = SGauche + Up;
			SDroite = SDroite + Up;

			SetMotor(SGauche, SDroite);
			WaitMs(2000);	// attendre
		}
		SGauche = 50;
		SDroite = 50;
	}
	return 0;
}
```

- **Annexe 6 : Programme ‘DISTANCE SENSOR’**

```
#include <pobRoboticSuite.h>

UInt32 i;
UInt32 value;

int main(void){
	InitCpu();
	InitUART0Intr(115200);
	SensorType * sensors;
	sensors = InitRobot();

	for(i=0;i<CHANNEL_NUMBER;i++){
		DebugPrintf("Sensor[%d] is: %d\r\n", i, sensors[i] );
	}
	while(1){
		value = GetDistanceSensor(1);
		value /= 10;
		DebugPrintf("Distance from port 1 is: %d\r\n", value);
	}
	return 0;
}
```

- **Annexe 7 : Programme ‘TEST COLOR SENSOR & DISTANCE SENSOR’**

```
#include <pobRoboticSuite.h>

UInt32 Telemetre;
UInt32 LigneAvant;
UInt32 i = 0;

int main(void){
	InitCpu();
	InitRobot(); // Initialise le robot
	InitUART0(115200);

	SensorType * sensors;
	sensors = InitRobot();

	for(;i<CHANNEL_NUMBER;i++){
		DebugPrintf("Sensor[%d] is: %d\r\n", i, sensors[i] );
	}

	while(1) {
		LigneAvant = GetDigitalIO(8);
		Telemetre = GetDistanceSensor(1);

		Telemetre /= 10;
		DebugPrintf("Distance from port 1 is: %d\r\n", Telemetre);

		if(LigneAvant == 1) {
			SetMotor(250, 250);
		}
		else { // BLANC AVANT
			SetMotor(-250, -250);
			WaitMs(1000);

			DoRotation(180, 250);		// tourner puis avancer
			WaitMs(2000);
		}
	} // Fin du while

	WaitMs(200);

	return 0;
}
```

- **Annexe 8 : Programme ‘TEST DOUBLE COLOR SENSOR & DISTANCE SENSOR’**

```
#include <pobRoboticSuite.h>

int SGauche = 200;
int SDroite = 200;
UInt32 Telemetre;
UInt32 LigneAvant;
UInt32 LigneArriere;

int main(void){
	InitCpu();
	InitRobot(); // Initialise le robot
	InitUART0(115200);

	SensorType * sensors;
	sensors = InitRobot();

	while(1) {
		LigneAvant = GetDigitalIO(8);
		LigneArriere = GetDigitalIO(6);
		Telemetre = GetDistanceSensor(1);

		Telemetre /= 10;
		if((LigneAvant == 1) && (LigneArriere == 1)) { // NOIR
			if(Telemetre < 20) {
				DoRotation(90, 250);  // tourner
			}
			else {
				SetMotor(SGauche, SDroite); // Avancer
			}
		}
		else if((LigneAvant == 1) && (LigneArriere == 0)) {
			if(Telemetre < 20) {
				DoRotation(90, 250);  // tourner puis avancer
				WaitMs(1000);
			}
			else {
				SetMotor(SGauche, SDroite);
			}
		}
		else if((LigneAvant == 0) && (LigneArriere == 1)) {
			SetMotor(-SGauche, -SDroite);
			WaitMs(1000);
```

```
                    DoRotation(180, 250);          // tourner puis avancer
                    WaitMs(1800);
                }
                else { // BLANC AVANT ET ARRIERE
                    SetMotor(0, 0);
                    DebugPrintf("GG JSUIS MORT");
                }
        } // Fin du while

        WaitMs(200);

        return 0;
}
```

- **Annexe 9 : Programme ‘FINAL’**

```
#include <pobRoboticSuite.h>

int SGPlus = 250;
int SDPlus = 250;
UInt32 Telemetre;
UInt32 Couleur;
UInt32 i = 0;

int main(void){
	InitCpu();
	InitRobot(); // Initialise le robot
	InitUART0(115200);

	SensorType * sensors;
	sensors = InitRobot();

	for(;i<CHANNEL_NUMBER;i++){
		DebugPrintf("Sensor[%d] is: %d\r\n", i, sensors[i] );
	}

	// Depart
	SetMotor(SGPlus, SDPlus);
	WaitMs(5000);

	while(1) {
		Couleur = GetDigitalIO(8);
		Telemetre = GetDistanceSensor(1);
		Telemetre /= 10;

		DebugPrintf("Distance : %d ", Telemetre);
		if(Couleur == 1) {
			if(Telemetre < 60) {
				SetMotor(SGPlus, SDPlus);
			 }
			 else {
				DoRotation(180, 250);		// tourner puis avancer
				WaitMs(200);
			}
		 }
		 else { // BLANC AVANT
			SetMotor(-250, -250);
			WaitMs(1000);
			DoRotation(180, 250);		// tourner puis avancer
			WaitMs(2000);
			SetMotor(SGPlus, SDPlus);
```

```
                WaitMs(2000);
            }
            WaitMs(100);
        }
        return 0;
}
```

TEST CONVERSION ULTRASONIC SENSOR DEPUIS ARDUINO JUSQU'A ROBOT POB : (RECEPTION ROBOT POB)

```
#include <pobRoboticSuite.h>

UInt32 Telemetre;

int main(void){
        InitCpu();
        InitRobot(); // Initialise le robot
        InitUART0(38400);

        while(1) {
            Telemetre = GetAnalog(5);

            DebugPrintf("Value Telemetre : %d \n", Telemetre);
            WaitMs(1000);
        }
        return 0;
}
```