M1 EEA SYSTÈMES ET MICROSYSTÈMES EMBARQUÉS
2023-2024

UNIVERSITÉ TOULOUSE III PAUL SABATIER

Université Fédérale Toulouse Midi-Pyrénées

Compte Rendu Des Travaux Pratiques

UE : Réalisation des systèmes

Projet Bureau d'Etudes : Robot Martien

Présenté par : RUGHOONAUTH Pravin Raj
SELATNIA Mohamed Abdellah

Responsable : M. PERISSE

Table de matières

1. Introduction

Le projet vise à programmer et tester un robot d'évitement d'obstacles utilisant un capteur à ultrason pour détecter les obstacles situés devant lui. L'objectif principal de ce projet est de répondre aux exigences spécifiées dans le cahier des charges, tout en approfondissant notre compréhension et nos compétences dans l'utilisation de la carte STM32, ainsi que dans la manipulation des capteurs, actionneurs et environnements de développement associés.

2. Cahier des Charges

Lorsqu'un obstacle est détecté à une distance inférieure à 10 cm, le robot doit exécuter une séquence d'évitement comprenant les actions suivantes :

- Activation d'une LED clignotante de couleur rouge pour indiquer la présence d'un obstacle.
- Recul du robot pendant une durée spécifiée pour s'éloigner de l'obstacle.
- Rotation du robot vers la droite pour contourner l'obstacle.
- Vérification à nouveau de la présence d'un obstacle à 10 cm devant le robot.
    - En cas de présence d'un objet, les séquences décrites précédemment sont réexécutées.
    - En revanche, en l'absence d'objet détecté, le robot poursuit son avancée.

La Figure 1 illustre le flowchart de notre cahier des charges. C'est essentiellement ce que nous prévoyons d'implémenter dans notre boucle 'while' en continu.

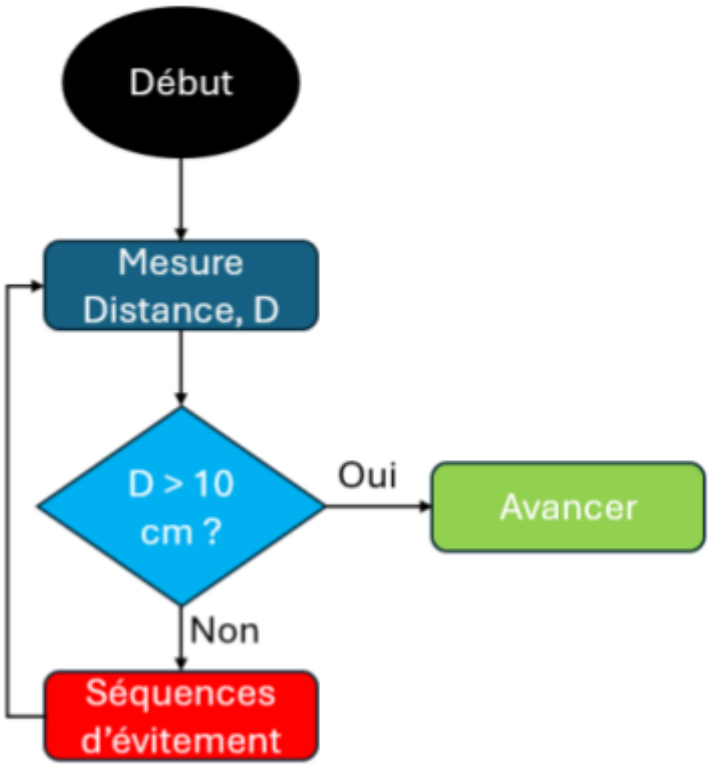

Figure 1 : Flowchart du cahier des charges

3. Matériels utilisés dans ce projet

3.1 Carte Nucleo WB55RG

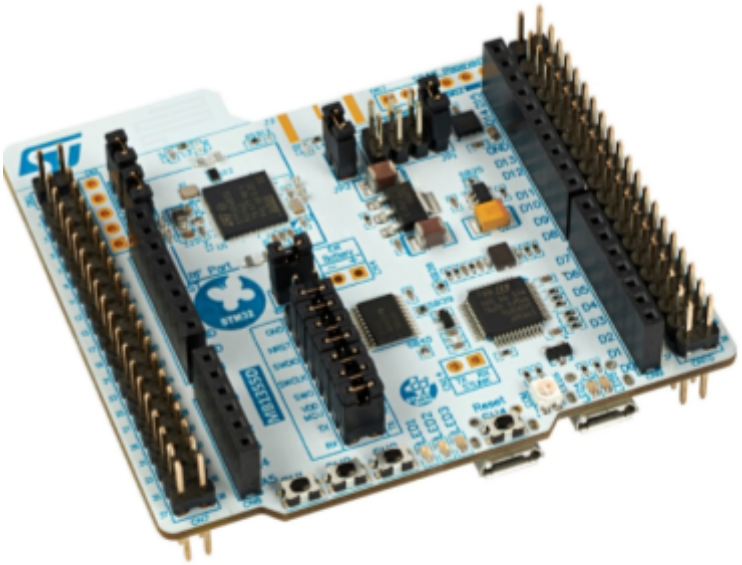

Figure 2 : Carte Nucleo STM32 - WB55RG

Dans notre projet, le microcontrôleur de la carte Nucleo WB55RG est chargé d'analyser en temps réel les données provenant du capteur à ultrasons, fournissant des informations sur la distance jusqu'à un éventuel obstacle devant le robot. Sur la base de ces données, le microcontrôleur prend des décisions pour contrôler les moteurs du robot de manière appropriée.

3.2 Nucleo Shield pour Alphabot

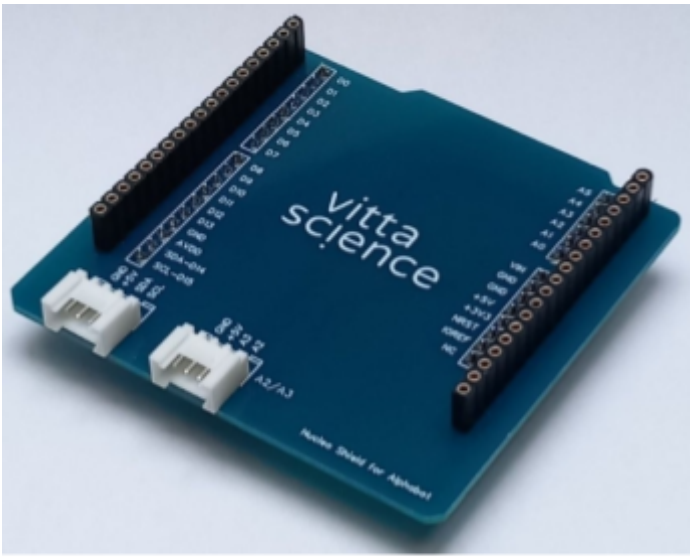

Figure 3 : Nucleo Shield pour Alphabot

La carte Nucleo Shield permet également une connexion interne sans l'utilisation de fils, grâce à ses broches d'extension et ses interfaces intégrées. Cela signifie que certains composants peuvent être directement montés et connectés à la carte Nucleo Shield sans avoir besoin de fils supplémentaires.

3.3 Capteur Ultrason HC SR04

Figure 4 : Capteur Ultrason HC SR04

Le capteur à ultrasons HC-SR04 est un dispositif utilisé pour mesurer la distance entre l'objet et le capteur en utilisant des ondes sonores ultrasonores dans une plage de 1 cm à 400 cm. Il se compose d'un émetteur et d'un récepteur ultrasonores qui fonctionnent en tandem pour émettre des impulsions ultrasonores et mesurer le temps nécessaire pour que ces impulsions soient réfléchies par un objet et reviennent au capteur.

3.3.1 Broches de Connexions

- Vcc = Alimentation +5 V DC
- Trig = Entrée de déclenchement de la mesure (Trigger input)
- Echo = Sortie de mesure donnée en écho (Echo output)
- GND = Masse de l'alimentation

3.3.2 Principe de fonctionnement

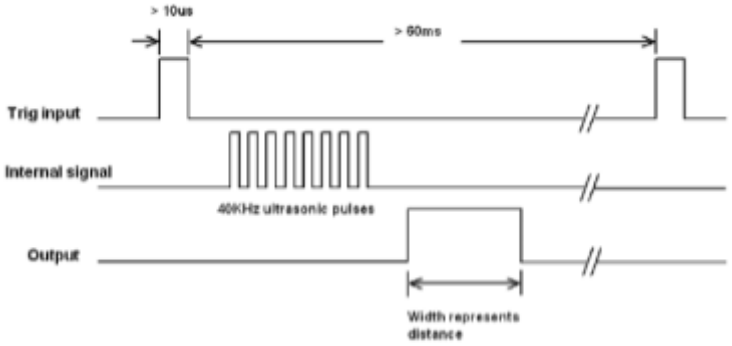

Figure 5 : Diagramme de timing du capteur HC SR04

Pour déclencher une mesure, il faut présenter une impulsion "high" (5 V) d'au moins 10 µs sur l'entrée "Trig". Le capteur émet alors une série de 8 impulsions ultrasoniques à 40 kHz, puis il attend le signal réfléchi. Lorsque celui-ci est détecté, il envoie un signal "high" sur la sortie "Echo", dont la durée est proportionnelle à la distance mesurée.

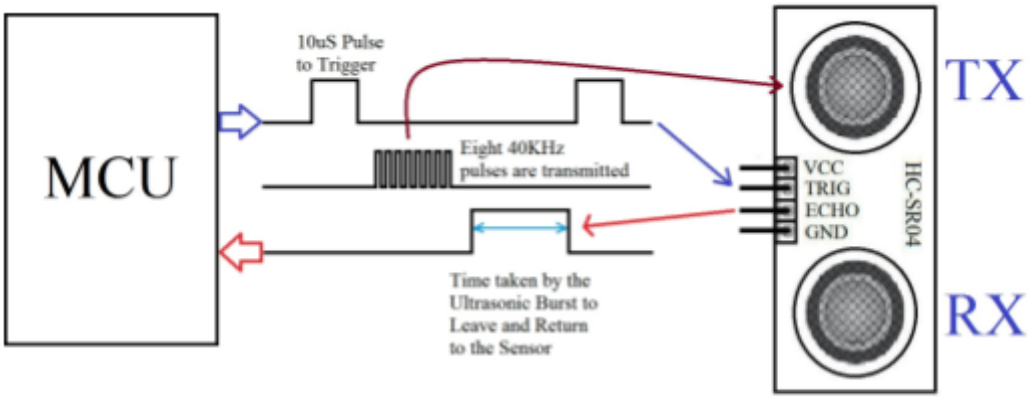

Figure 6 : Principe de fonctionnement

3.3.3 Distance de l'objet

La distance parcourue par un son se calcule en multipliant la vitesse du son, environ 340 m/s (ou 0,034 cm/ µs) par le temps de propagation, soit : $d = v \cdot t$ (distance = vitesse · temps). On sait que le son fait un aller-retour. Le temps vaut donc la moitié. Nous calculons alors la distance avec la formule :
Distance =(0,034*temps)/2

3.4 Motor driver module

AlphaBot utilise la puce conductrice L298P de ST pour commander les 2 moteurs, ayant comme entrées : IN1, IN2, IN3, IN4, ENA et ENB

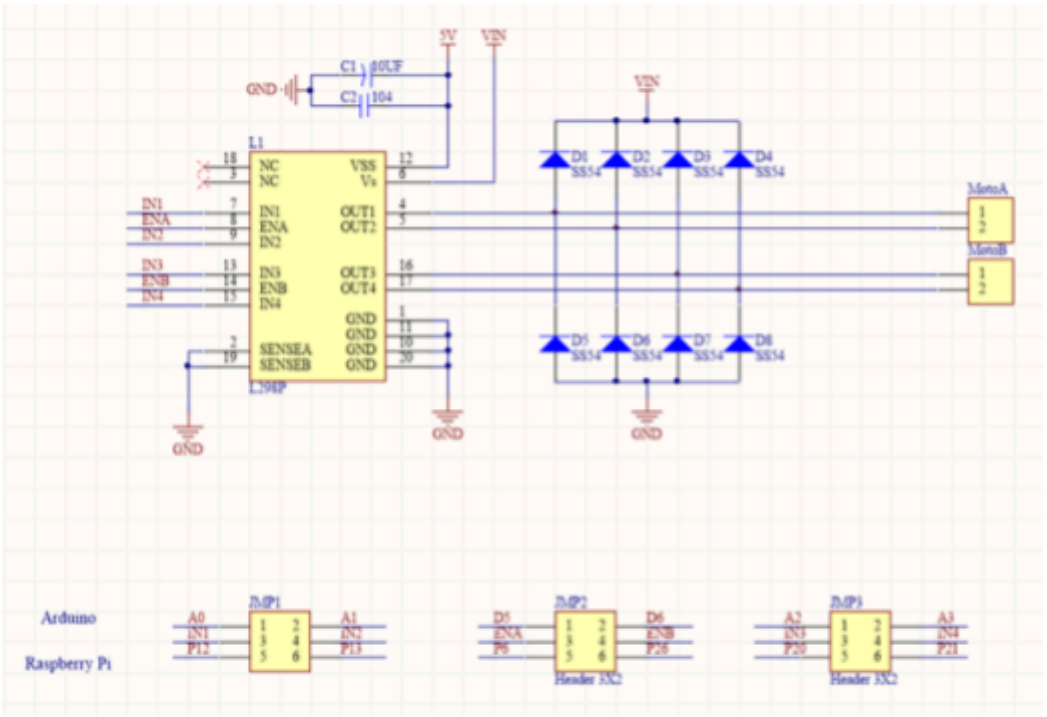

Figure 7 : Driver L298P

IN1 et IN2 sont connectés au moteur gauche, tandis que IN3 et IN4 sont connectés au moteur droit. ENA et ENB sont des broches d'activation de sortie, activées en niveau élevé. Lorsqu'elles sont conduites à un niveau élevé, l'impulsion PWM sera émise à partir de IN1, IN2, IN3 et IN4, afin de contrôler la vitesse du moteur du robot.

Table de vérité pour la commande des moteurs

IN1	IN2	IN3	IN4	Descriptions
1	O	O	1	Les 2 moteurs tournent vers l'avant et donc le robot s'avance tout droit
0	1	1	0	Les 2 moteurs tournent vers l'arrière et donc le robot se recule
0	0	0	1	Lorsque le moteur gauche s'arrête et que le moteur droit tourne vers l'avant, le robot se tourne à gauche.
1	0	0	0	Lorsque le moteur droite s'arrête et que le moteur gauche tourne vers l'avant, le robot se tourne à droite.
0	0	0	0	Lorsque les moteurs s'arrêtent, le robot s'arrête également.

3.5 Le Robot Martien assemblé

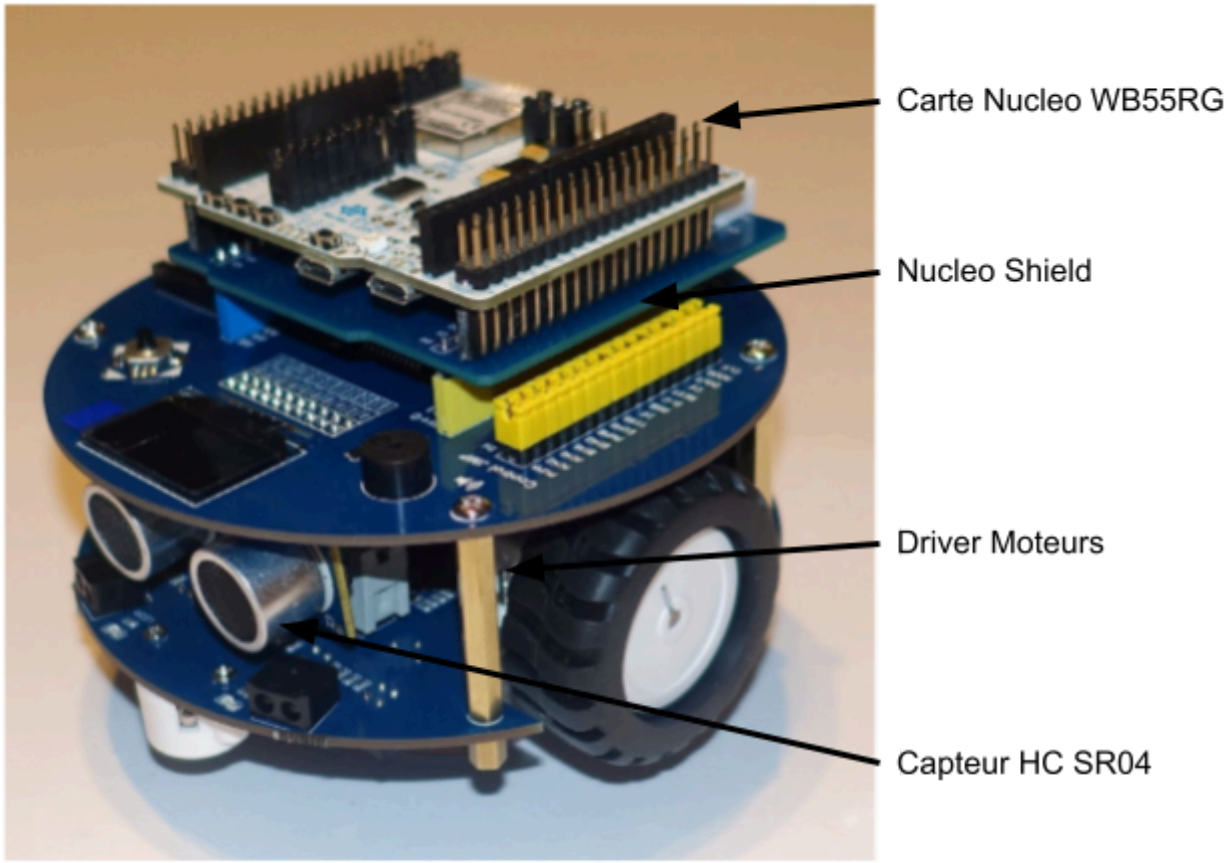

Figure 8 : Le Robot Martien assemblé

4. Schéma de câblage

Étant donné que nous utilisons un Nucleo Shield, nous devons identifier les broches correspondantes sur le microcontrôleur. Pour ce faire, nous consultons les informations disponibles sur le site Mbed.

Interface definition of driver module:

Interfaces	Raspberry Pi	Arudino
IN1	P12	A0
IN2	P13	A1
ENA	P6	D5
IN3	P20	A2
IN4	P21	A3
ENB	P26	D6

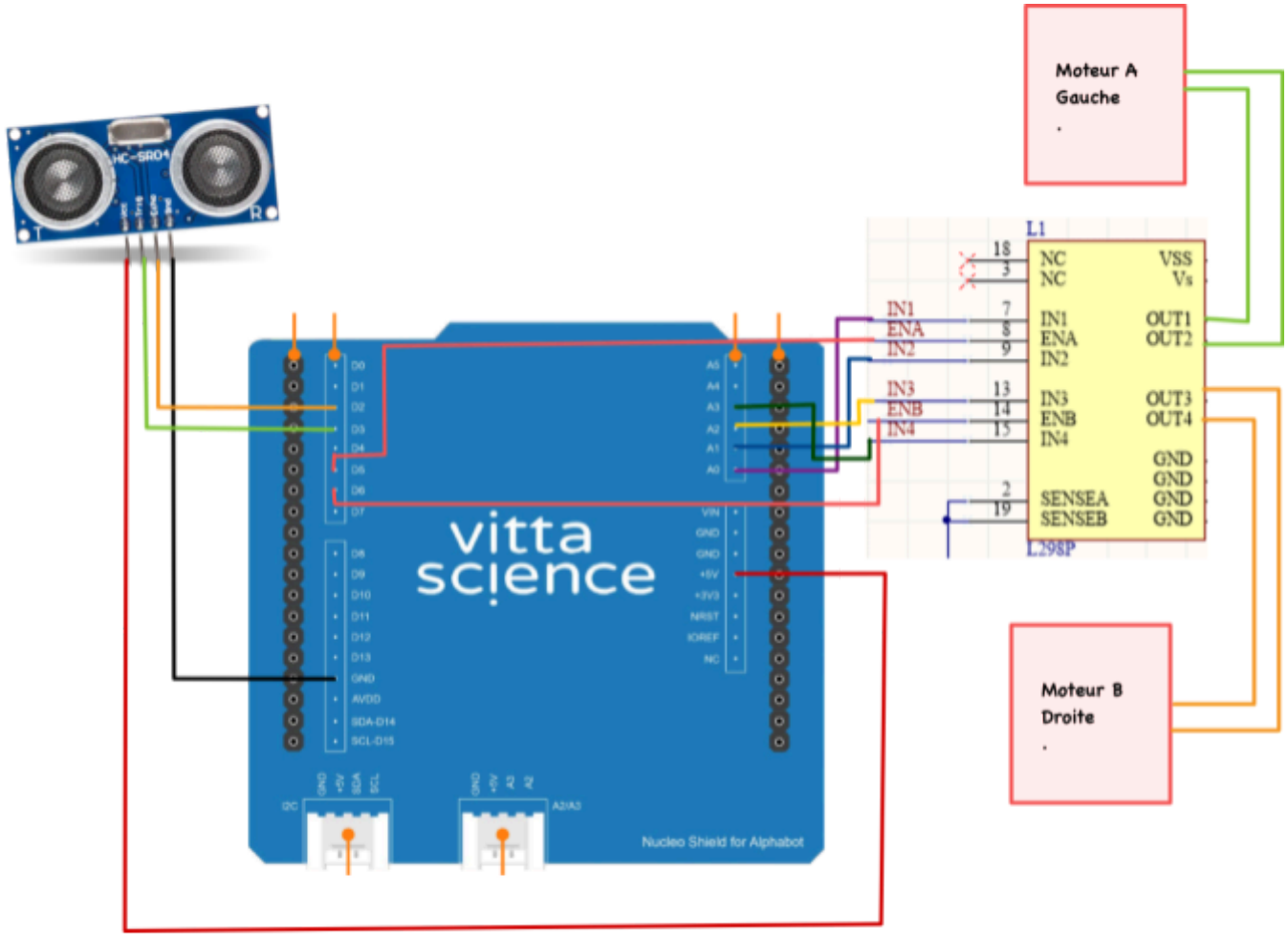

Figure 9 : Schéma de câblage du projet

5. Configuration STM32 CubeMx

5.1 L'horloge et Timers

Nous devons générer des périodes de 1 microseconde pour pouvoir utiliser correctement le capteur HC SR04. L'horloge interne de notre microcontrôleur fonctionne à une fréquence maximum de 32MHz. Nous avons donc joué avec les valeurs pour atteindre cette fréquence maximum vers les timers.

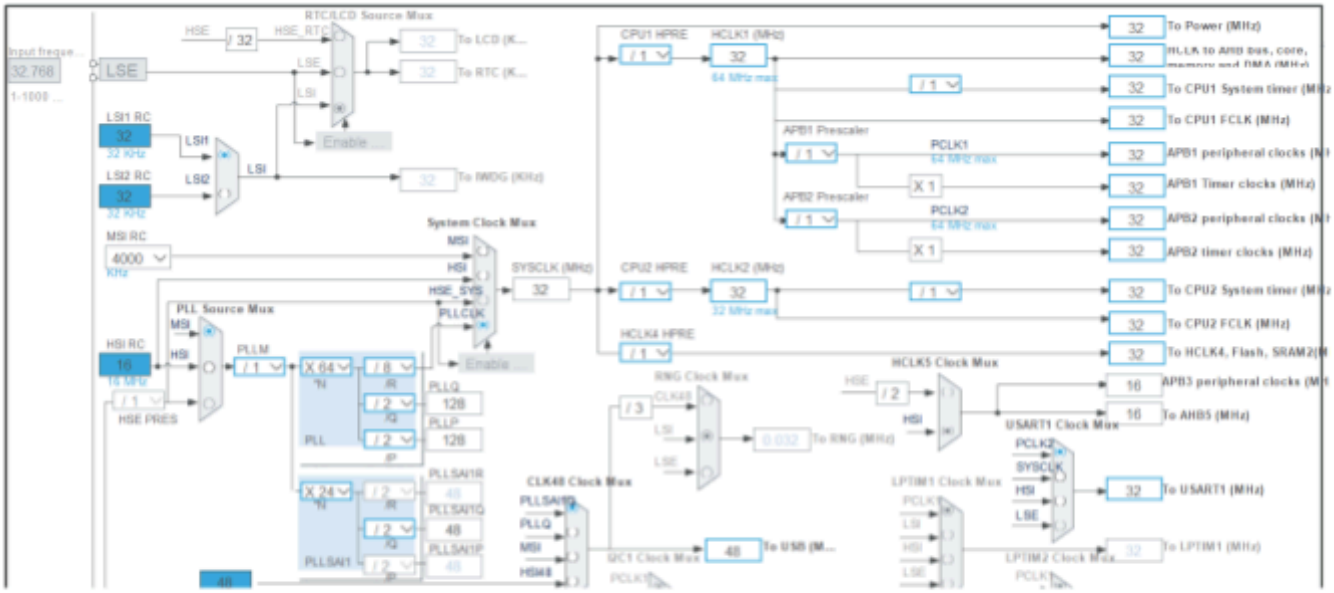

Figure 10 : Configuration de l'horloge

Puis nous avons utilisé le Timer 16 avec un prescaler de 32-1 qui fera une fréquence de 1MHz et ainsi une période de 1us.

Figure 11: Configuration Timer16

Pour générer les signaux PWM nécessaires au fonctionnement des moteurs, nous utilisons le canal 1 des timers 1 et 2.

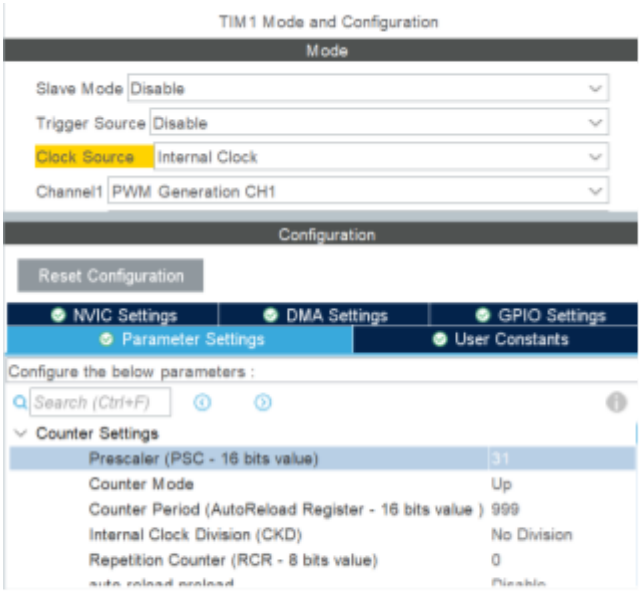

Figure 12 : Configuration Timer1

Notre HCLK est à 32 MHz.
32MHz / ((31 + 1) * (999 + 1)) = 32 * 10 ^ 6 Hz / (32 * 10 ^ 3) = 10 ^ 3 Hz = 1 kHz
Ce paramètre entraîne une sortie PWM avec une période de 1 kHz.

On fait la même configuration pour le Timer 2.

Figure 13: Configuration Timer 2

5.2 IOC : Configurations des broches d'entrées et des sorties

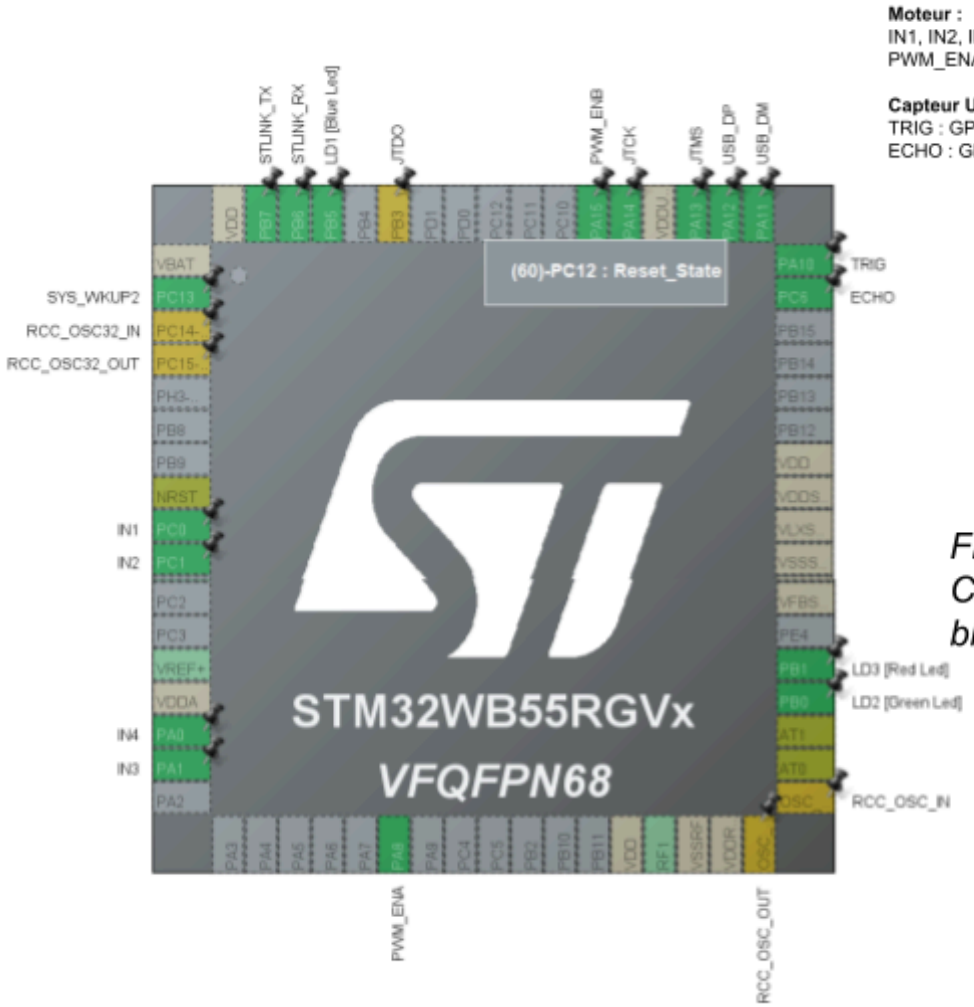

Figure 13 : Configuration des broches

6. Programmation

6.1 Le capteur HC SR04

Cette fonction mesure la distance entre le robot et un objet devant lui.

```
// Fonction pour mesurer la distance entre le robot et un obstacle
uint16_t Mesure_Distance(void)
{
HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_SET);  // pull the TRIG pin HIGH
__HAL_TIM_SET_COUNTER(&htim16, 0);
while (__HAL_TIM_GET_COUNTER(&htim16) < 10);  // wait for 10 us
HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET);  // pull the TRIG pin low

while (!(HAL_GPIO_ReadPin (ECHO_PORT, ECHO_PIN)) ); //Attendre que echo passe à l'état haut
Value1 = __HAL_TIM_GET_COUNTER(&htim16); // valeur du timer à laquelle echo passe à l'état haut

while ((HAL_GPIO_ReadPin (ECHO_PORT, ECHO_PIN))); //Attendre que echo passe à l'état bas
Value2 = __HAL_TIM_GET_COUNTER(&htim16); // valeur du timer à laquelle echo passe à l'état bas

// Value2-Value1 : durée pendant laquelle echo reste en etat haut
// On multiplie la moitié de cette durée par la vitesse de l'ultrason pour avoir la distance

Distance = ((Value2-Value1)* 0.034)/2;
HAL_Delay(50);
return Distance;
}
```

Les variables utilisées sont déclarées ici :

```
/////*****CAPTEUR HC SR04********//////
#define TRIG_PIN GPIO_PIN_10 //definition du port en sortie
#define TRIG_PORT GPIOA
#define ECHO_PIN GPIO_PIN_6 // defnition du port en entree
#define ECHO_PORT GPIOC

uint32_t Value1 = 0; // l'instant quand echo passe à 1
uint32_t Value2 = 0; // l'instant quand echo passe à 0
uint16_t Distance  = 0;  // en cm
```

6.2 Les moteurs commandés par le driver L298P

```
/////*****MOTEUR L298P********//////
const uint16_t pwm =100; // pour ajuster la vitesse des roues
//****Moteur Gauche********
#define IN1_PIN GPIO_PIN_0 //definition du port en sortie
#define IN1_PORT GPIOC
#define IN2_PIN GPIO_PIN_1 //definition du port en sortie
#define IN2_PORT GPIOC

//****Moteur Droite********
#define IN3_PIN GPIO_PIN_1 //definition du port en sortie
#define IN3_PORT GPIOA
#define IN4_PIN GPIO_PIN_0 //definition du port en sortie
#define IN4_PORT GPIOA

/* *****NOTE*********
PWM pour IN1 et IN2 -> PWM_ENA (Timer 1 Channel 1) // Roue Gauche
PWM pour IN2 et IN3 -> PWM_ENB (Timer 2 Channel 1) // Roue Droite
************************************* */
// ********FIN NOTE*************
```

Nous avons ensuite créé plusieurs fonctions pour faire les mouvements du robot selon la table de vérité.

```
// fonction pour initialiser le moteur
void Init_Moteur(void)
{
    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm); //

    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, pwm); //

    HAL_Delay(500); // Attendre 0,5 s
}
```

Initialiser le robot

```
//fonction pour avancer 1001 (IN1=1 ; IN2=0, IN3=0 ; IN4=1)
void Move_Forward(void)
{
    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_SET);
    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm);

    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_SET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, pwm);

    HAL_Delay(500); // Attendre 0,5 s
}
```

Avancer

```
//fonction pour reculer 0110 (IN1=0 ; IN2=1, IN3=1 ; IN4=0)
void Move_Backward(void)
{
    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_SET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm);

    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_SET);
    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, pwm);

    HAL_Delay(500); // Attendre 0,5 s
}
```

Reculer

```
//fonction pour arreter (IN1=0 ; IN2=0, IN3=0 ; IN4=0)
void Stop(void)
{
    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm);
    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, pwm);

    HAL_Delay(500); // Attendre 0,5 s
}
```

Arrêter

```
//fonction pour tourner a droite 1000 (IN1=1 ; IN2=0, IN3=0 ; IN4=0)
void Turn_Right(void)
{
    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_SET);
    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm);

    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_RESET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, pwm);

    HAL_Delay(500); // Attendre 0,5 s
}
```

Tourne à droite

```
//fonction pour tourner a gauche 0001 (IN1=0 ; IN2=0, IN3=0 ; IN4=1)
void Turn_Left(void)
{
    HAL_GPIO_WritePin(IN1_PORT, IN1_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(IN2_PORT, IN2_PIN, GPIO_PIN_RESET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm);

    HAL_GPIO_WritePin(IN3_PORT, IN3_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(IN4_PORT, IN4_PIN, GPIO_PIN_SET);
    // Ajustez le rapport cyclique pour contrôler la vitesse
    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, pwm);

    HAL_Delay(500); // Attendre 0,5 s
}
```

Tourne à gauche

La boucle while :

```c
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

    uint16_t distance = 0; // Initialisation de la distance

        // Mesure la distance pendant que le robot avance
        while (1)
        {
            distance = Mesure_Distance(); // Mesurer la distance

            if (distance <= 10)
            {
                // Obstacle détecté à moins de 10cm, déclenche l'évitement
                // Séquence : reculer, tourner, vérifier à nouveau la distance
                Blink_Red();
                Move_Backward();
                HAL_Delay(50);
                Turn_Right();
                HAL_Delay(50);
                distance = Mesure_Distance();
                if (distance <= 10)
                {
                    continue; // Répéter l'évitement si l'obstacle est toujours détecté
                }
                Move_Forward();
                HAL_Delay(50);
            } else {
                // Pas d'obstacle, avance normalement
                Blink_Green();
                Move_Forward();
                HAL_Delay(50);
            }
        }
} // end while
/* USER CODE END 3 */
}
```

La boucle while est principalement le cahier des charges comme expliqué dans la première partie et le flowchart.

L’intégralité du code est disponible sur GitHub.

7. Conclusion

Dans l'ensemble, ce projet robotique a été une expérience enrichissante qui nous a permis d'explorer en profondeur les aspects pratiques de la conception, de la programmation et de la mise en œuvre de systèmes autonomes. En combinant des composants matériels tels que la carte Nucleo WB55RG et le capteur ultrason HC-SR04 avec des compétences de programmation avancées, nous avons pu développer un robot capable d'éviter les obstacles de manière efficace. Ce projet nous a également permis de mieux comprendre les défis et les possibilités associés à la robotique autonome, tout en renforçant nos compétences techniques et notre capacité à travailler en équipe. En fin de compte, cette expérience nous a apporté une précieuse perspective sur le potentiel futur de la robotique dans divers domaines d'application et a stimulé notre intérêt pour la poursuite de recherches et de projets dans ce domaine passionnant.