Rapport de Projet

Robot Suiveur de Ligne avec Évitement d'Obstacles

2 juin 2024

Nicolas SILINOU & Uriel Anthony KOUADJO

Etudiants en 3e année d'informatique
Année scolaire : 2023-2024

Enseignants référants

M. SELLAMI
Mme. TRABELSI

Établissement :

ESIEA

74 Bis av Maurice Thorez, 94200 Ivry sur Seine
0143902121
www.eisea.fr

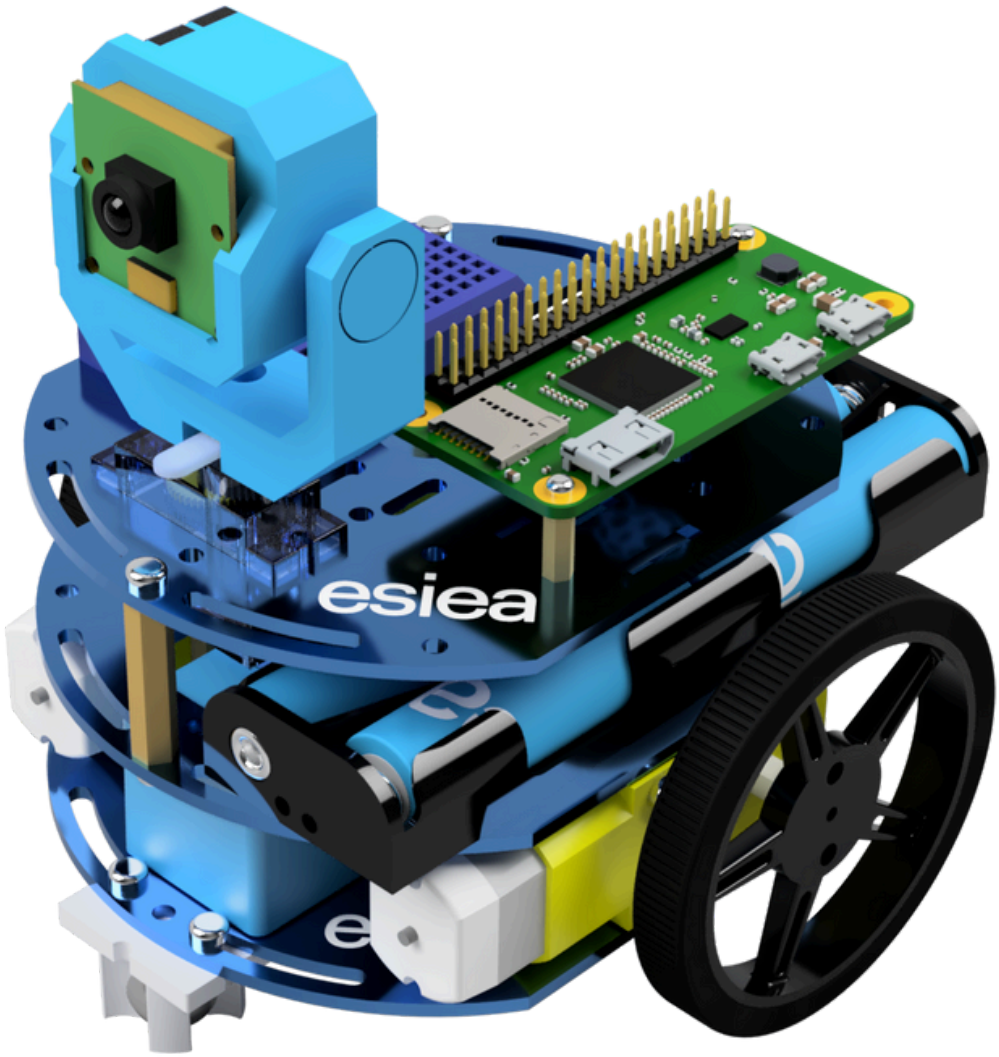
Sommaire

Introduction

1

Contexte et Idée Générale du Projet Proposé

Le projet de robot suiveur de ligne avec évitement d'obstacles s'inscrit dans le cadre du cours de programmation embarquée. L'objectif est de concevoir et de programmer un robot capable de suivre une ligne noire tracée au sol tout en détectant et en évitant les obstacles présents sur son chemin. Ce projet permet de mettre en pratique les concepts d'électronique, de programmation embarquée et de contrôle de moteurs.

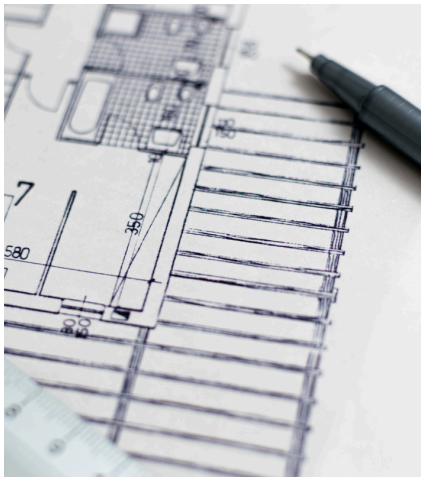


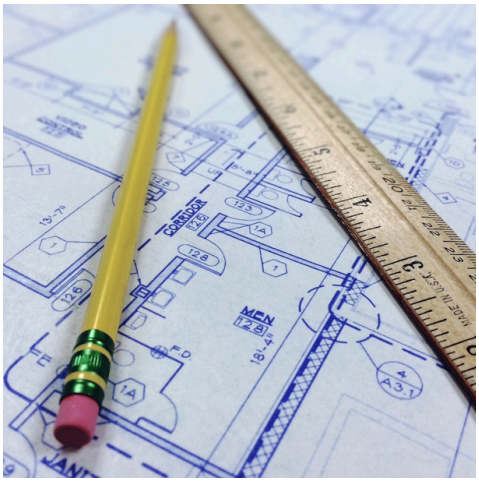
Projet

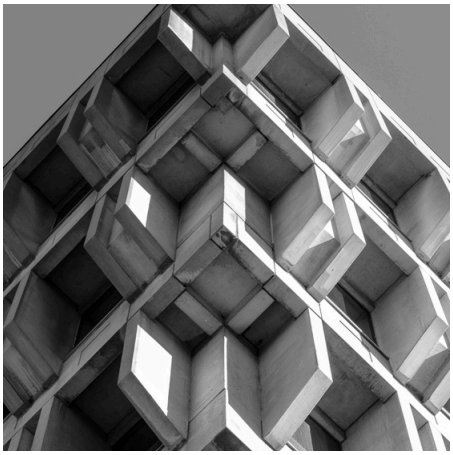
Détails de l'Objectif du Projet et de ses Principales Fonctionnalités

L'objectif principal de ce projet est de concevoir et de programmer un robot autonome capable de suivre une ligne tracée au sol tout en détectant et en évitant les obstacles sur son chemin. Les fonctionnalités clés du robot incluent :

- Suivi de Ligne : Utilisation de capteurs réflecteurs pour détecter et suivre une ligne noire sur une surface blanche.
- Évitement d'Obstacles : Utilisation d'un capteur ultrason pour mesurer la distance aux obstacles et arrêter le robot lorsqu'un obstacle est détecté à moins de 10 cm.
- Contrôle des Moteurs : Utilisation de PWM (modulation de largeur d'impulsion) pour ajuster la vitesse des moteurs et contrôler la direction du robot.
- Indications Visuelles : Utilisation de LEDs RGB pour signaler l'état du robot (vert pour en mouvement, rouge pour à l'arrêt).
- Indication Sonore : Utilisation d'un buzzer pour signaler la détection d'obstacles.







Organisation et Répartition des Tâches

Le développement du projet a été divisé en trois étapes principales afin de structurer efficacement le travail et de garantir une progression logique et ordonnée.

Étape 1 : Contrôle des Moteurs

- Objectif : Faire rouler les deux roues du robot.
- Détails :
  - Initialisation des GPIO pour les moteurs.
  - Développement des fonctions de base pour contrôler les moteurs (avance, arrêt).
  - Test des moteurs pour s'assurer qu'ils répondent correctement aux commandes.

Étape 2 : Suivi de Ligne

- Objectif : Permettre au robot de suivre une ligne tracée au sol.
- Détails :
  - Intégration des suiveurs de ligne (capteurs optiques) avec le microcontrôleur.
  - Développement de la logique de suivi de ligne en utilisant les lectures des capteurs.
  - Ajustement des moteurs en fonction des données des capteurs pour suivre la ligne.
  - Validation du comportement du robot sur différentes trajectoires.

Étape 3 : Détection d'Obstacles et Indication LED RGB

- Objectif : Ajouter la détection d'obstacles et indiquer l'état des roues à l'aide d'une LED RGB.
- Détails :
  - Intégration du capteur de distance ultrason pour la détection d'obstacles.
  - Développement de la logique d'évitement d'obstacles, arrêt du robot à la détection et reprise une fois l'obstacle retiré.
  - Ajout et configuration de la LED RGB pour indiquer l'état des roues (rouge à l'arrêt, vert en mouvement).
  - Tests finaux pour s'assurer que le robot peut suivre une ligne, éviter les obstacles et indiquer l'état des roues de manière fiable.

L'Architecture Matérielle Schématisée sur Kicad et sa Description

L'architecture matérielle du projet a été soigneusement planifiée et réalisée en utilisant Kicad. Le schéma suivant illustre les connexions principales entre les composants :

Schéma de Connexion :

- Microcontrôleur STM32L053R8 :
  - Moteurs : PA1, PA2, PA5, PA6
  - Capteur Ultrason : Trigger sur PA7, Echo sur PA8
  - Suiveurs de Ligne : Capteurs sur PA0, PA4
  - LED RGB : Rouge sur PA9, Vert sur PA10, Bleu sur PA11

Description des Connexions :

- Moteurs : Connectés via des ponts en H pour permettre le contrôle directionnel (avant, arrière, arrêt).
- Capteur Ultrason : Utilisé pour mesurer la distance par écho, configuré avec un pin de déclenchement et un pin de réception.
- Suiveurs de Ligne : Capteurs optiques détectant la ligne noire pour guider le robot le long de la trajectoire.
- LED RGB : Utilisée pour indiquer l'état du robot, configurée pour s'allumer en vert lorsque les roues tournent et en rouge lorsque le robot est à l'arrêt.

Description de la Structure de votre Programme

Initialisation des Périphériques : La configuration initiale des GPIO, du Timer, et des capteurs a été cruciale pour le bon fonctionnement du robot. Chaque composant a été configuré pour optimiser l'interaction avec le microcontrôleur.

- GPIO : Configuration des pins pour les moteurs, capteurs, et LED RGB.
- Timer : Utilisation de TIM2 pour mesurer les durées des impulsions du capteur ultrason.
- Capteurs : Initialisation des capteurs de ligne et du capteur ultrason pour des lectures précises.

Fonctions Principales :

- Contrôle des Moteurs :
  - Motor_Init(): Initialise les GPIO des moteurs.
  - Motor_Forward(): Fait avancer le robot en activant les moteurs appropriés.
  - Motor_Left(), Motor_Right(): Tourne le robot à gauche ou à droite.
  - Motor_Stop(): Arrête le robot en désactivant les moteurs.
- Lecture du Capteur Ultrason :
  - Ultrasonic_Init(): Initialise les GPIO et le Timer pour le capteur ultrason.
  - Ultrasonic_ReadDistance(): Mesure la distance en utilisant le temps de vol de l'onde ultrason.
- Indication LED RGB :
  - LED_RGB_Init(): Initialise les GPIO pour la LED RGB.
  - LED_RGB_SetRed(), LED_RGB_SetGreen(), LED_RGB_Off(): Contrôle les couleurs de la LED pour indiquer l'état des roues.

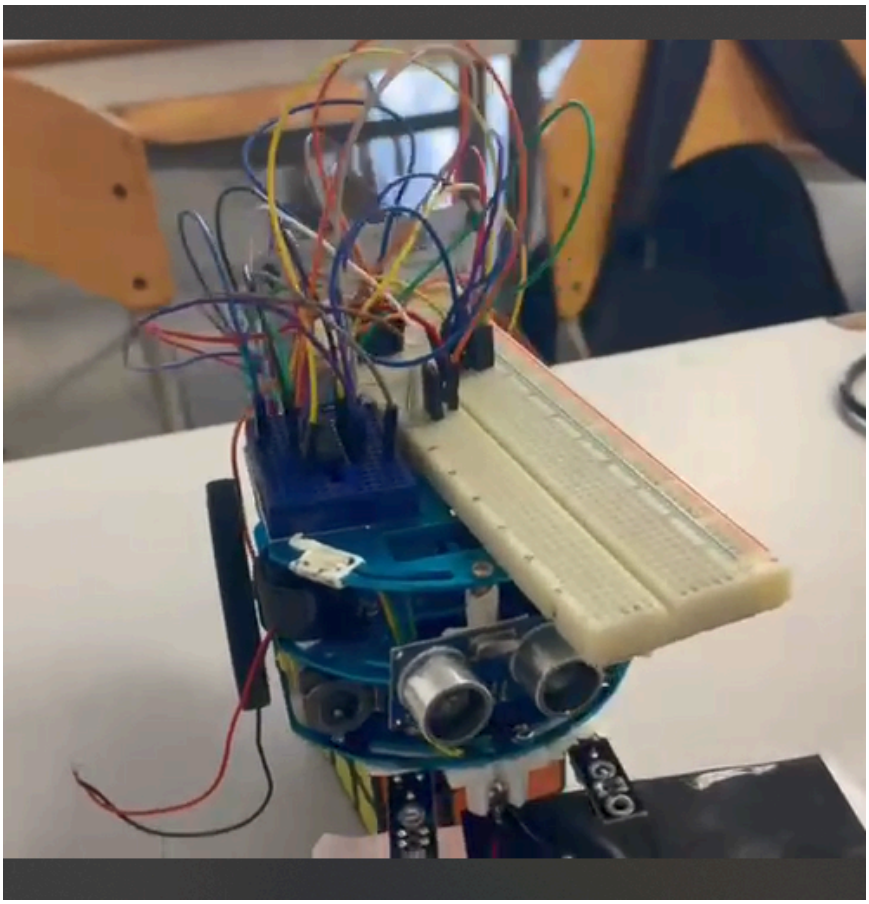
Boucle Principale : La boucle principale combine les lectures des capteurs et le contrôle des moteurs pour guider le robot :

```
while (1) {
    uint8_t leftSensor = GPIO_ReadPin(GPIOA, 0);  // Lecture du suiveur de ligne gauche (PA0)
    uint8_t rightSensor = GPIO_ReadPin(GPIOA, 4);  // Lecture du suiveur de ligne droit (PA4)
    uint32_t distance = Ultrasonic_ReadDistance();  // Lire la distance

    if (distance < 10) {  // Si un obstacle est détecté à moins de 10 cm
        Motor_Stop();  // Arrêter les moteurs
        LED_RGB_SetRed();  // Allumer la LED rouge
    } else {
        // Suivre la ligne si aucun obstacle n'est détecté
        if (leftSensor && rightSensor) {  // Les deux détectent la ligne
            Motor_Forward();  // Avancer
            LED_RGB_SetGreen();  // Allumer la LED verte
        } else if (!leftSensor && rightSensor) {  // Seul le suiveur de ligne gauche détecte la ligne
            Motor_Right(); // Tourner à droite
            LED_RGB_SetGreen();  // Allumer la LED verte
        } else if (leftSensor && !rightSensor) {  // Seul le suiveur de ligne droit détecte la ligne
            Motor_Left();  // Tourner à gauche
            LED_RGB_SetGreen();  // Allumer la LED verte
        } else {
            Motor_Stop();  // Arrêter si aucun ne détecte la ligne
            LED_RGB_SetRed();  // Allumer la LED rouge
        }
    }

    delay_ms(100);  // Attendre 100 ms avant la prochaine lecture
}
```

Illustration des Fonctionnalités Principales



Vue d'ensemble du robot avec les composants montés.

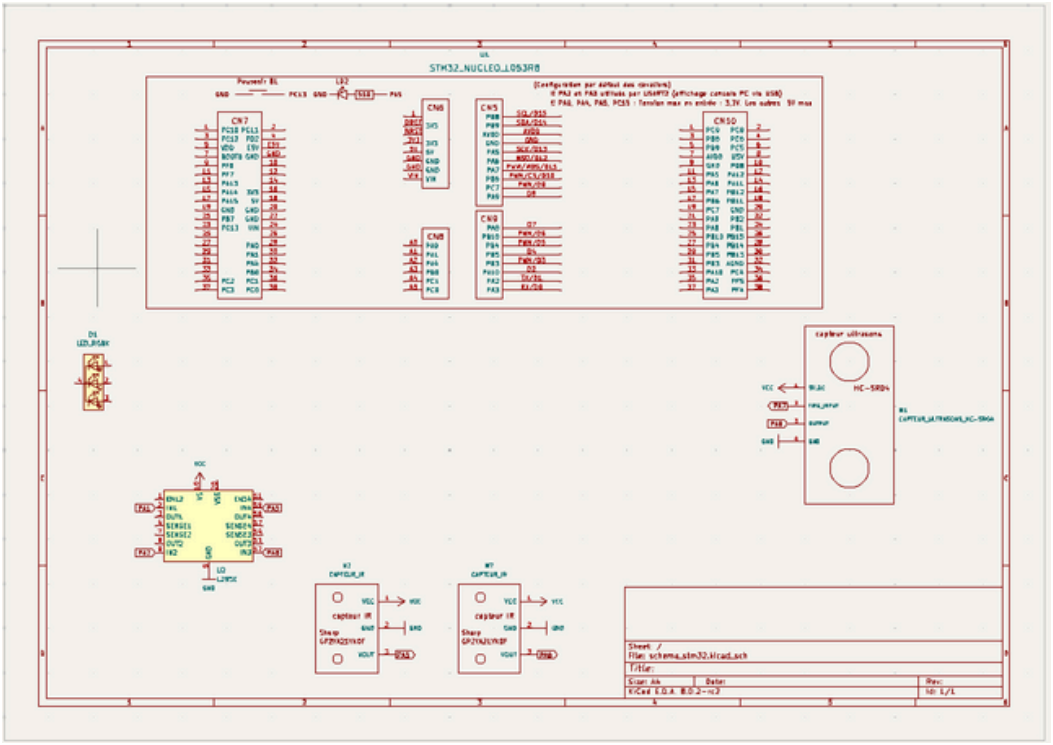


Schéma de connexion sur Kicad

FICHIER GPIO.C:

```
#include "gpio.h"

void GPIO_Init(void) {
 // Activer l'horloge GPIOA
 RCC->IOPENR |= RCC_IOPENR_GPIOAEN;

 // Configurer les pins PA1, PA2, PA5, PA6, PA7, PA8, PA9, PA10, PA11 en mode sortie et entrée
  GPIOA->MODER &= ~(GPIO_MODER_MODE1 | GPIO_MODER_MODE2 | GPIO_MODER_MODE5 | GPIO_MODER_MODE6 | GPIO_MODER_MODE7 | GPIO_MODER_MODE8 | GPIO_MODER_MODE9 | GPIO_MODER_MODE10 | GPIO_MODER_MODE11);
 GPIOA->MODER |= (GPIO_MODER_MODE1_0 | GPIO_MODER_MODE2_0 | GPIO_MODER_MODE5_0 | GPIO_MODER_MODE6_0 | GPIO_MODER_MODE7_0 | GPIO_MODER_MODE9_0 | GPIO_MODER_MODE10_0 | GPIO_MODER_MODE11_0);
 GPIOA->MODER &= ~GPIO_MODER_MODE8; // PA8 en entrée pour Echo

 // Configurer les pins en mode push-pull
 GPIOA->OTYPER &= ~(GPIO_OTYPER_OT_1 | GPIO_OTYPER_OT_2 | GPIO_OTYPER_OT_5 | GPIO_OTYPER_OT_6 | GPIO_OTYPER_OT_7 | GPIO_OTYPER_OT_9 | GPIO_OTYPER_OT_10 | GPIO_OTYPER_OT_11);

 // Configurer les pins en mode haute vitesse
  GPIOA->OSPEEDR |= (GPIO_OSPEEDER_OSPEED1 | GPIO_OSPEEDER_OSPEED2 | GPIO_OSPEEDER_OSPEED5 | GPIO_OSPEEDER_OSPEED6 | GPIO_OSPEEDER_OSPEED7 | GPIO_OSPEEDER_OSPEED9 | GPIO_OSPEEDER_OSPEED10 | GPIO_OSPEEDER_OSPEED11);

 // Désactiver les résistances de pull-up/pull-down
 GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPD1 | GPIO_PUPDR_PUPD2 | GPIO_PUPDR_PUPD5 | GPIO_PUPDR_PUPD6 | GPIO_PUPDR_PUPD7 | GPIO_PUPDR_PUPD8 | GPIO_PUPDR_PUPD9 | GPIO_PUPDR_PUPD10 | GPIO_PUPDR_PUPD11);

 // Configurer PA0 et PA4 comme entrées pour les suiveurs de ligne
 GPIOA->MODER &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE4);
 GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD4); // Pas de pull-up ni pull-down
}

void GPIO_SetPinHigh(GPIO_TypeDef *GPIOx, uint32_t PinNumber) {
 GPIOx->BSRR = (1 << PinNumber);
}

void GPIO_SetPinLow(GPIO_TypeDef *GPIOx, uint32_t PinNumber) {
 GPIOx->BRR = (1 << PinNumber);
}

uint8_t GPIO_ReadPin(GPIO_TypeDef *GPIOx, uint32_t PinNumber) {
 return (GPIOx->IDR & (1 << PinNumber)) ? 1 : 0;
}
```

Timer.c:

```c
#include "timer.h"

void Timer_Init(void) {
    // Activer l'horloge pour TIM2
    RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;

    // Configurer le timer
    TIM2->PSC = 16 - 1;  // Prescaler pour obtenir une résolution de 1 microseconde (16 MHz / 1
    TIM2->ARR = 0xFFFFFFFF;  // Auto-reload à la valeur maximale
    TIM2->CNT = 0;  // Initialiser le compteur à 0
    TIM2->CR1 = TIM_CR1_CEN;  // Activer le timer
}

void Timer_Start(void) {
    TIM2->CNT = 0;  // Réinitialiser le compteur
    TIM2->CR1 |= TIM_CR1_CEN;  // Démarrer le timer
}

void Timer_Stop(void) {
    TIM2->CR1 &= ~TIM_CR1_CEN;  // Arrêter le timer
}

uint32_t Timer_GetValue(void) {
    return TIM2->CNT;  // Lire la valeur du compteur
}
```

FICHIER ultrasonic.c:

```
#include "ultrasonic.h"

// Fonction d'initialisation du capteur ultrason
void Ultrasonic_Init(void) {
 GPIO_Init(); // Initialiser les GPIO
 Timer_Init(); // Initialiser le timer

 // Configurer PA7 en sortie pour Trigger
 GPIOA->MODER |= GPIO_MODER_MODE7_0;
 GPIOA->MODER &= ~GPIO_MODER_MODE8; // PA8 en entrée pour
Echo
}

// Fonction pour lire la distance du capteur ultrason
uint32_t Ultrasonic_ReadDistance(void) {
 uint32_t startTime, endTime, duration;

 // Générer une impulsion de 10us sur Trigger
 GPIO_SetPinLow(GPIOA, TRIG_PIN);
 for (volatile int i = 0; i < 1000; i++); // Délai de quelques cycles
 GPIO_SetPinHigh(GPIOA, TRIG_PIN);
 for (volatile int i = 0; i < 1000; i++); // Attendre 10us
 GPIO_SetPinLow(GPIOA, TRIG_PIN);

 // Attendre que Echo passe à HIGH
 while (!GPIO_ReadPin(GPIOA, ECHO_PIN));

 // Enregistrer le temps de départ
 Timer_Start();
 startTime = Timer_GetValue();

 // Attendre que Echo repasse à LOW
 while (GPIO_ReadPin(GPIOA, ECHO_PIN));

 // Enregistrer le temps de fin
 endTime = Timer_GetValue();
 Timer_Stop();

 // Calculer la durée de l'impulsion Echo
 duration = endTime - startTime;

 // Convertir la durée en distance (cm)
 uint32_t distance = (duration * 0.0343) / 2;

 return distance;
}
```

FICHIER line_sensor.c:

```
#include "line_sensor.h"

void LineSensor_Init(void) {
 // Activer l'horloge GPIOA
 RCC->IOPENR |= RCC_IOPENR_GPIOAEN;

 // Configurer PA5 et PA6 en mode analogique (capteurs de ligne)
 GPIOA->MODER |= GPIO_MODER_MODE5 | GPIO_MODER_MODE6;

 // Activer l'horloge ADC
 RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;

 // Configurer et calibrer l'ADC
 ADC1->CR |= ADC_CR_ADCAL; // Démarrer la calibration
 while (ADC1->CR & ADC_CR_ADCAL); // Attendre la fin de la calibration

 // Configurer l'ADC
 ADC1->CFGR1 |= ADC_CFGR1_RES_1; // Résolution 8 bits
 ADC1->CR |= ADC_CR_ADEN; // Activer l'ADC
 while (!(ADC1->ISR & ADC_ISR_ADRDY)); // Attendre que l'ADC soit prêt
}

uint32_t LineSensor_ReadLeft(void) {
 //Sélectionner le canal 5 (PA5)
 ADC1->CHSELR = ADC_CHSELR_CHSEL5;
 //Démarrer la conversion
 ADC1->CR |= ADC_CR_ADSTART;
 //Attendre la fin de la conversion
 while (!(ADC1->ISR & ADC_ISR_EOC));
 //Lire la valeur convertie
 return ADC1->DR;
}

uint32_t LineSensor_ReadRight(void) {
 // Sélectionner le canal 6 (PA6)
 ADC1->CHSELR = ADC_CHSELR_CHSEL6;
 // Démarrer la conversion
 ADC1->CR |= ADC_CR_ADSTART;
 // Attendre la fin de la conversion
 while (!(ADC1->ISR & ADC_ISR_EOC));
 // Lire la valeur convertie
 return ADC1->DR;
}
```

```c
#include "motor.h"

void Motor_Init(void) {
    // Initialiser les GPIO pour les moteurs
    GPIO_Init();
}

void Motor_Forward(void) {
    // Faire avancer les moteurs
    GPIO_SetPinHigh(GPIOA, MOTOR1_IN1_PIN);  // PA1 HIGH - Moteur droit en avant
    GPIO_SetPinLow(GPIOA, MOTOR1_IN2_PIN);   // PA2 LOW - Moteur droit en avant
    GPIO_SetPinHigh(GPIOA, MOTOR2_IN1_PIN);  // PA5 HIGH - Moteur gauche en avant
    GPIO_SetPinLow(GPIOA, MOTOR2_IN2_PIN);   // PA6 LOW - Moteur gauche en avant
}

void Motor_Stop(void) {
    // Arrêter les moteurs
    GPIO_SetPinLow(GPIOA, MOTOR1_IN1_PIN);   // PA1 LOW
    GPIO_SetPinLow(GPIOA, MOTOR1_IN2_PIN);   // PA2 LOW
    GPIO_SetPinLow(GPIOA, MOTOR2_IN1_PIN);   // PA5 LOW
    GPIO_SetPinLow(GPIOA, MOTOR2_IN2_PIN);   // PA6 LOW
}

void Motor_Left(void) {
    // Tourner à gauche en arrêtant le moteur gauche et faisant avancer le moteur droit
    GPIO_SetPinHigh(GPIOA, MOTOR1_IN1_PIN);  // PA1 HIGH - Moteur droit en avant
    GPIO_SetPinLow(GPIOA, MOTOR1_IN2_PIN);   // PA2 LOW - Moteur droit en avant
    GPIO_SetPinLow(GPIOA, MOTOR2_IN1_PIN);   // PA5 LOW - Moteur gauche à l'arrêt
    GPIO_SetPinLow(GPIOA, MOTOR2_IN2_PIN);   // PA6 LOW - Moteur gauche à l'arrêt
}

void Motor_Right(void) {
    // Tourner à droite en arrêtant le moteur droit et faisant avancer le moteur gauche
    GPIO_SetPinLow(GPIOA, MOTOR1_IN1_PIN);   // PA1 LOW - Moteur droit à l'arrêt
    GPIO_SetPinLow(GPIOA, MOTOR1_IN2_PIN);   // PA2 LOW - Moteur droit à l'arrêt
    GPIO_SetPinHigh(GPIOA, MOTOR2_IN1_PIN);  // PA5 HIGH - Moteur gauche en avant
    GPIO_SetPinLow(GPIOA, MOTOR2_IN2_PIN);   // PA6 LOW - Moteur gauche en avant
}
```

Conclusion

1. Connaissances et Apprentissages Acquis
La réalisation de ce projet a permis d'acquérir et de consolider plusieurs compétences techniques et pratiques dans le domaine de l'électronique embarquée et de la robotique. Les principaux apprentissages incluent :

- **Programmation Embarquée** : La maîtrise de la programmation bas niveau sur un microcontrôleur STM32, sans utiliser les fonctions de haut niveau du constructeur. Cela a permis de mieux comprendre le fonctionnement interne des périphériques et des registres du microcontrôleur.
- **Configuration des GPIO et Timers :** La configuration et l'utilisation des GPIO pour contrôler les moteurs, lire les capteurs et gérer la LED RGB. L'utilisation du timer pour mesurer les durées d'impulsion a été particulièrement instructive.
- **Intégration de Capteurs** : L'intégration et la gestion de différents capteurs, notamment les suiveurs de ligne et le capteur de distance ultrason. Cela a permis de comprendre comment ces capteurs fonctionnent et comment les interfacer avec un microcontrôleur.
- **Développement d'Algorithmes de Contrôle :** La conception d'algorithmes pour le suivi de ligne et l'évitement d'obstacles. La logique conditionnelle pour la prise de décision en temps réel a été une compétence clé développée au cours de ce projet.

2. Difficultés Principales Surmontées
Au cours du projet, plusieurs défis ont été rencontrés et surmontés :

- **Configuration du Timer** : La configuration précise du timer pour mesurer le temps d'impulsion du capteur ultrason a été complexe. Il a fallu comprendre le fonctionnement des registres et des interruptions pour obtenir des mesures précises.
- **Interfaçage des Capteurs** : La lecture fiable des suiveurs de ligne et du capteur ultrason a nécessité des ajustements dans la configuration des GPIO et des tests répétés pour assurer la précision des lectures.
- **Logique de Contrôle :** La conception de la logique pour le suivi de ligne et l'évitement d'obstacles a été un défi majeur. Trouver les conditions optimales pour la détection et la prise de décision en temps réel a nécessité plusieurs itérations et tests.
- **Gestion de la LED RGB** : L'intégration de la LED RGB pour qu'elle réagisse correctement à l'état des roues a nécessité une synchronisation précise avec les autres fonctions du robot.

3. Améliorations Possibles du Projet
Bien que le projet ait atteint ses objectifs principaux, plusieurs améliorations pourraient être apportées :

- **Communication Sans Fil :** Ajouter des capacités de communication sans fil (Bluetooth ou Wi-Fi) permettrait de contrôler le robot à distance et de surveiller son état en temps réel.
- **Algorithme de Suivi de Ligne** : Améliorer l'algorithme de suivi de ligne pour qu'il soit plus robuste et réactif, notamment en utilisant des techniques de filtrage des données des capteurs pour réduire le bruit et les fausses détections.
- **Capteurs Additionnels :** Intégrer des capteurs supplémentaires, comme des capteurs de proximité infrarouge ou des gyroscopes, pour améliorer la navigation et la détection des obstacles.
- **Autonomie Énergétique** : Optimiser la gestion de l'énergie pour prolonger l'autonomie du robot, en utilisant des techniques de mise en veille et de gestion efficace de la batterie.
- **Intelligence Artificielle :** Explorer l'utilisation d'algorithmes d'intelligence artificielle pour améliorer la prise de décision en temps réel et rendre le robot plus autonome et adaptable à des environnements complexes.

En conclusion, ce projet a été une expérience enrichissante et formatrice, offrant une opportunité de travailler sur des technologies avancées et de résoudre des problèmes complexes de manière créative et innovante. Les connaissances et compétences acquises seront précieuses pour des projets futurs et pour le développement professionnel dans le domaine de l'électronique embarquée et de la robotique.