



UNIVERSITÉ DE CARTHAGE
ÉCOLE POLYTECHNIQUE DE TUNISIE
PROJET D'ÉCOLE

Rapport projet d'école
Robot tondeuse "ROBOMOW 1"

TRAVAIL ÉLABORÉ PAR :

Hamdane Brini

Chahine Ghanoudi

Omar Kammoun

Mohamed Dridi

Zakaria M'hemed

Nidhal Ben Rajeb

ENCADRÉ PAR :

M.Ben Hassena Mohamed Amine

Mme.Rezgui Taysir

Résumé

Ce projet se concentre sur la conception et la réalisation d'un robot tondeuse à gazon capable de fonctionner de manière autonome. Tondre la pelouse et entretenir l'herbe en général a toujours été une cible d'automatisation en raison de la monotonie et de la simplicité de la tâche. Pourtant, une solution parfaitement autonome et entièrement fiable n'a pas encore été mise sur le marché. Dans ce projet, nous visons à fournir les qualités susmentionnées dans un package abordable. Les piliers principaux de ce projet comportent la conception et construction d'un robot naviguant entièrement de manière autonome avec le modèle de vision par ordinateur nécessaire, l'équiper des outils nécessaires pour traverser les terrains herbeux, des pneus tout-terrain crantés au système de suspension et la conception et implémentation d'un système de coupe de l'herbe et dans le châssis. Nos études comporteront principalement 3 départements :

- Vision par ordinateur : Nécessaire pour la navigation sur le terrain et l'évitement des obstacles.
- Mécanique : Concevoir le système de suspension approprié pour supporter le robot, et choisir les dimensions appropriées pour le mécanisme de coupe de l'herbe.
- Électronique : Concevoir le circuit de contrôle du robot et le tester sur le logiciel de simulation disponible.

Ce rapport sera

Un certain nombre de difficultés ont été rencontrées au cours de ce projet. Certaines étaient liées aux compétences techniques. Par exemple, il a fallu déterminer un modèle mécanique approprié pour l'interaction des roues avec l'herbe afin de calculer le couple de direction nécessaire et trouver les paramètres nécessaires pour dimensionner correctement le moteur alimentant le mécanisme de la lame. D'autres difficultés concernaient les compétences interpersonnelles. Des problèmes de gestion et de coopération ont été rencontrés tout au long du processus, mais ils ont été résolus grâce à une communication claire et à une résolution proactive des conflits.



Table des matières

1 LISTE DES FIGURES	4
2 CONTEXTE DU PROJET ET CAHIER DE CHARGE	5
2.1 Introduction	5
2.2 Cahier de charges	5
2.2.1 Objectifs	5
2.2.2 Périmètre du projet	6
2.2.3 Spécifications	7
3 CONCEPTION ET ANALYSE : Programmation	7
3.1 Inclusions de bibliothèques	7
3.2 Définition des broches	7
3.3 Déclaration d'objets et variables globales	7
3.4 Prototypes de fonctions	8
3.5 Fonction de configuration initiale (<code>setup</code>)	8
3.6 Lecture de la distance ultrasonique	9
3.7 Contrôle du moteur sans balais	9
3.8 Réglage de la hauteur de la lame	10
3.9 Contrôle du mouvement du robot	10
3.10 Évitement des obstacles	10
3.11 Boucle principale (<code>loop</code>)	10
4 Conception et analyse : Mécanique	17
4.1 Dessin d'ensemble	17
4.2 Conception des suspensions	18
4.2.1 Calcul de couple résistant sur les roues	18
4.2.2 Description du fonctionnement du système	20
4.3 Conception du système de découpage	22
4.3.1 Description du fonctionnement du système	22
4.4 Conception du châssis	23
5 Conception et analyse : Vision par ordinateur	24
5.1 Choix de l'Architecture de Calcul	25
5.2 Acquisition de Données	25
5.3 Collaboration avec le Groupe de Commande et d'Asservissement	25
5.4 Pré-Traitement des Données	25
5.5 Choix du Modèle de Vision par Ordinateur	25
5.6 Inference et Traitement	25
5.7 Conclusion	26
6 Experimentation et test	26

1 LISTE DES FIGURES

Table des figures

1	Définition du besoin	5
2	Diagramme d'analyse des fonctions de service.	5
3	Partie locomotion et navigation	6
4	Partie système de découpage	6
5	Shématisation du terrain	12
6	Navigation du robot dans le terrain	13
7	Passage à la colonne suivante	13
8	Rotation des roues	14
9	Rotation traditionnelle du robot	14
10	Etape 1	15
11	Etape 2	15
12	Etape 3	15
13	Contournement d'un obstacle	15
14	Détection de la fin du terrain	16
15	Vue de dessus	17
16	Section A-A	17
17	Vue de face	17
18	Vue dessous	18
19	Valeurs de c et ϕ selon les classifications des sols	19
20	Schéma simplifié de la roue	19
21	Schéma du tracé de la roue après braquage	19
22	Schéma cinématique du système de suspension	20
23	Modèle du système de suspension réaliser avec le logiciel SolidWorks	21
24	Système de découpage avec régulation du niveau d'herbe	23
25	Vue de face de la maquette	23
26	Vue de droite de la maquette	23
27	Vue de perspective	24
28	Modèle 3D du robot réalisé avec Solidworks	24
29	Fonctionnement de la vision par ordinateur	25
30	Le modèle sous fonctionnement	26
31	Simulation de quelques parties du code	27

2 CONTEXTE DU PROJET ET CAHIER DE CHARGE

2.1 Introduction

Le robot **ROBOMOW 1** est un robot équipé d'une tondeuse réglable, de système de vision par ordinateur, capable d'éviter les obstacles ainsi qu'une capacité de locomotion sur terrain accidenté.

2.2 Cahier de charges

2.2.1 Objectifs

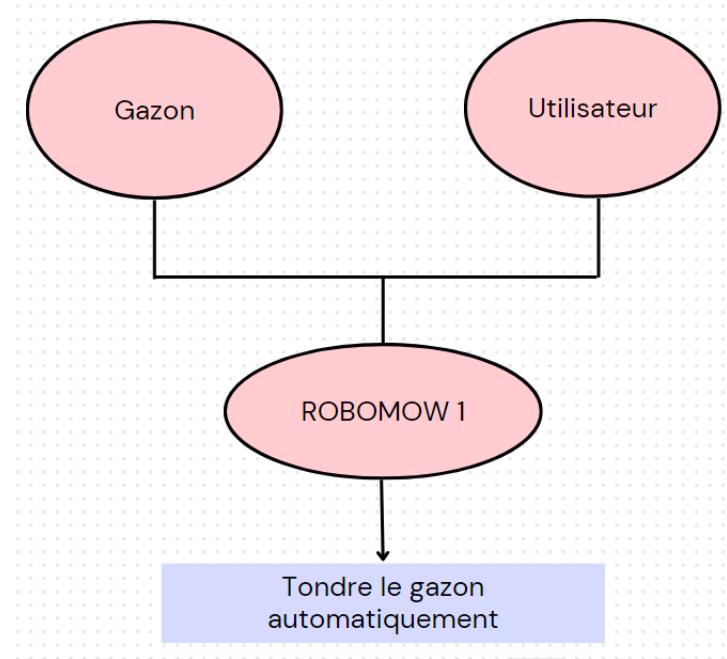


FIGURE 1 – Définition du besoin.

L'objectif principal de ce projet est de fournir une solution de tonte de gazon entièrement autonome et financièrement abordable. Nous souhaitons créer un robot capable de tondre la pelouse de manière autonome, avec précision et efficacité, tout en s'adaptant aux différents types de terrains et hauteurs d'herbe. Une interface utilisateur intuitive sera également développée pour permettre une configuration et un contrôle simples du robot. En résumé, notre but est de rendre la tonte de gazon plus facile, et accessible à tous.

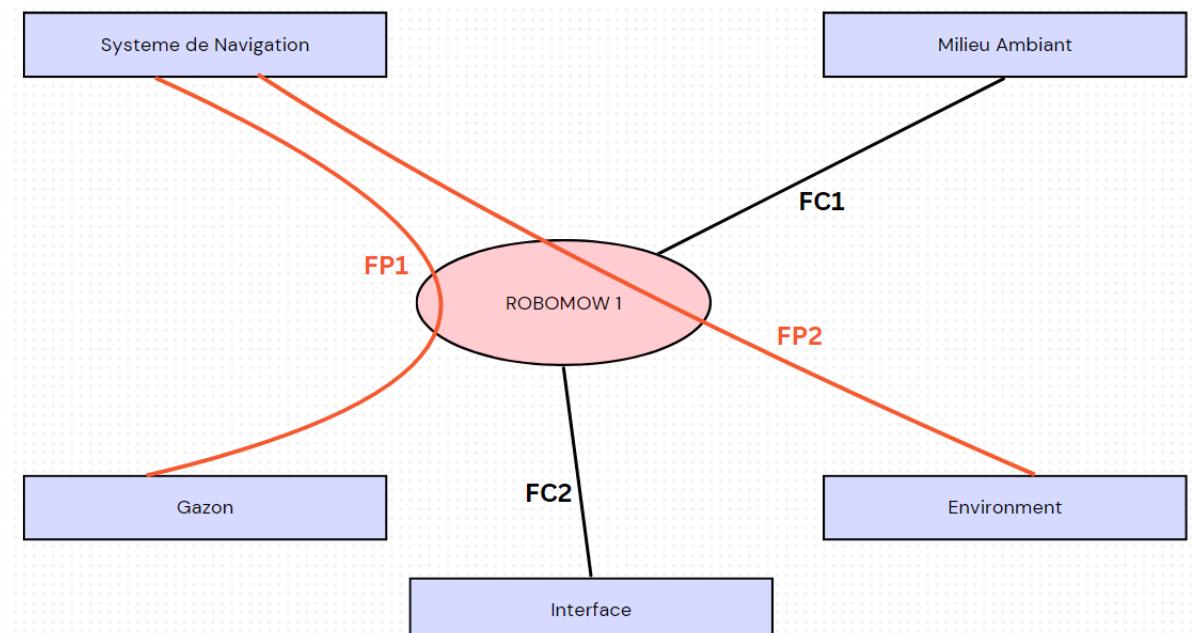


FIGURE 2 – Diagramme d'analyse des fonctions de service.

- **FP1** : Tondre le gazon automatiquement selon un cycle fourni par le système de navigation.
- **FP2** : Naviguer l'environnement grâce aux données fournis par le système de navigation.
- **FC1** : Résister aux conditions du milieu ambiant.
- **FC2** : Avoir une interface facile à utiliser.

2.2.2 Périmètre du projet

Le robot ROBOMOW 1 sera équipé d'un système de vision par ordinateur, d'un ensemble de capteurs et d'une caméra pour la détection d'obstacles et l'analyse de l'environnement. Il disposera également d'un système de locomotion adapté aux terrains difficiles et d'un mécanisme de coupe réglable en hauteur pour une tonte précise et uniforme. Le développement comprendra des phases de recherche portant sur trois volets :

Volet vision par ordinateur, volet conception des suspensions et du système de tondeuse, et finalement volet programmation et cablage.

Le diagramme FAST suivant détaillera plus en profondeur la composition des différentes parties du robot, en mettant particulièrement l'accent sur la locomotion, la navigation et le système de découpe.

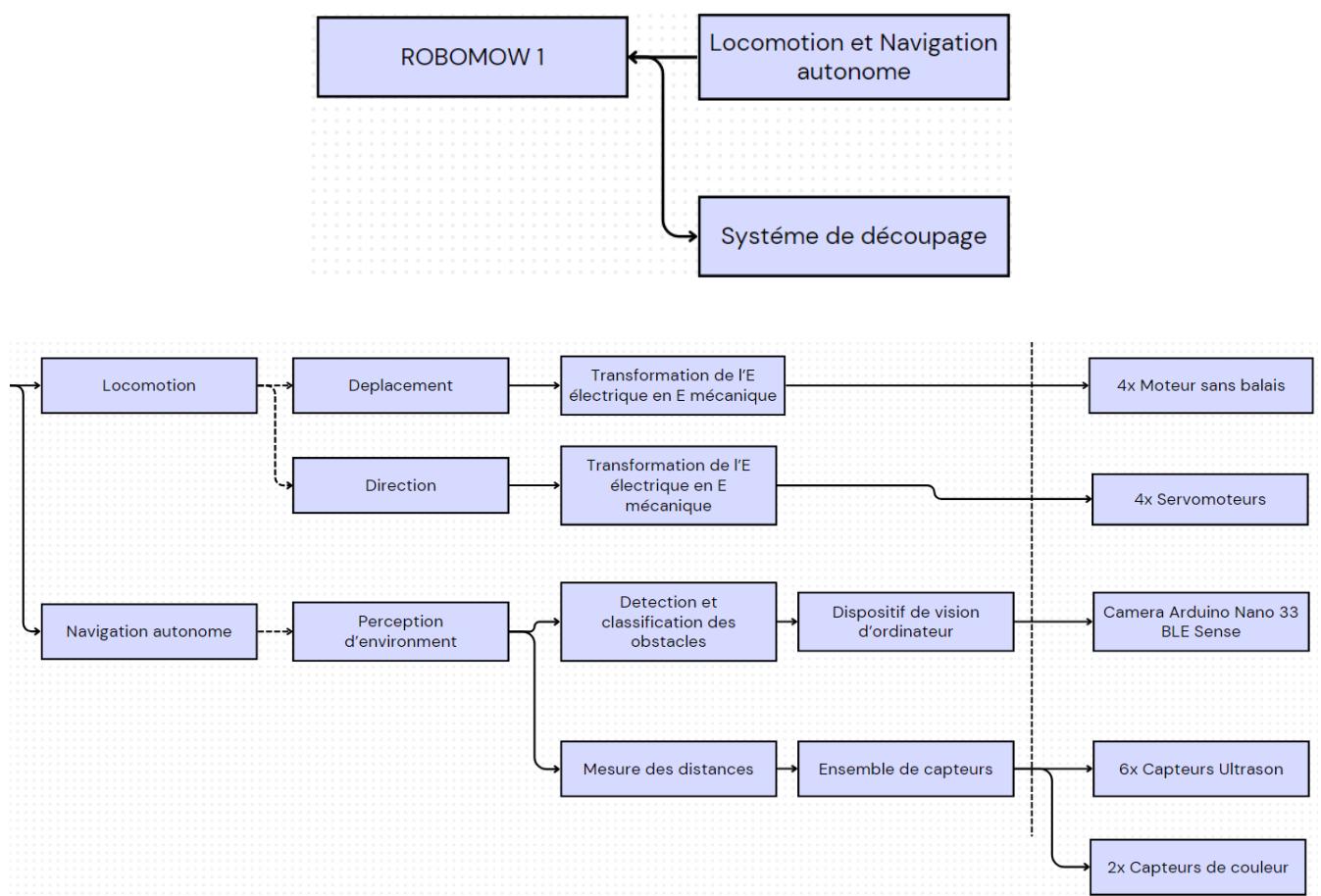


FIGURE 3 – Partie locomotion et navigation.

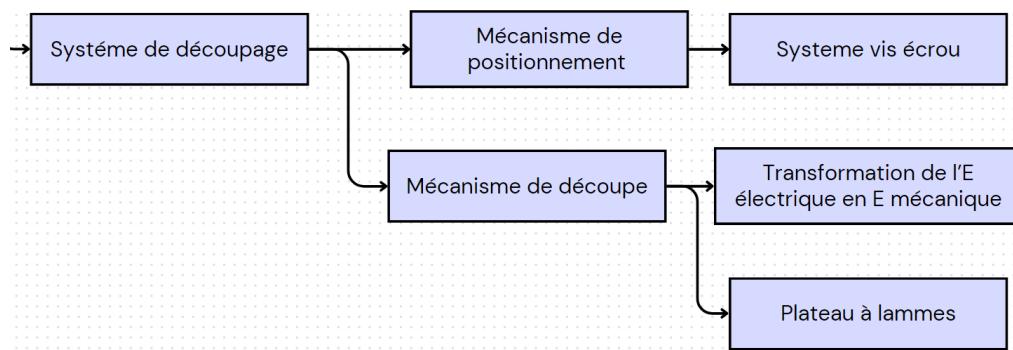


FIGURE 4 – Partie système de découpage.

2.2.3 Spécifications

Catégorie	Spécifications
Système de contrôle	Arduino Méga
Capteurs	6x Capteurs ultrason, 2x Capteurs de couleurs Φ mm
Mécanisme de découpage	3x Lames
Interface	Afficheur LCD
Mécanisme de découpage	3x Lames, étandue 76 Φ mm
Roues	4x Roues crantées 200 Φ mm

3 CONCEPTION ET ANALYSE : Programmation

3.1 Inclusions de bibliothèques

```
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <Stepper.h>
```

Cette partie contient les inclusions des bibliothèques nécessaires au projet. LiquidCrystal_I2C.h est utilisée pour contrôler l'écran LCD via le protocole I2C, Servo.h est utilisée pour contrôler les servomoteurs, et Stepper.h est utilisée pour contrôler le moteur pas à pas.

3.2 Définition des broches

```
// Définition des broches pour les capteurs TCS3200
const int S0[] = {2, 12};
const int S1[] = {3, 13};
const int S2[] = {4, A0};
const int S3[] = {5, A1};
const int outPin[] = {6, A2};
const int whiteThreshold = 5000;

// Définition des broches pour les capteurs à ultrasons
const int trigPin[] = {22, 23, 24, 25};
const int echoPin[] = {26, 27, 28, 29};

// Définition des broches pour les moteurs
const int motorPin[] [2] = {
    {2, 3}, {4, 5}, {6, 7}, {8, 9}
};
const int brushlessMotorPin1 = 32;
const int brushlessMotorPin2 = 33;

// Définition des broches pour le moteur pas à pas
const int stepperPin1 = 30;
const int stepperPin2 = 31;

// Définition des broches pour les servomoteurs
const int servoPin[] = {22, 23, 24, 25};
Servo servo[4];
```

Cette partie définit les broches utilisées pour chaque composant du projet.

3.3 Déclaration d'objets et variables globales

```
// Définition des servomoteurs pour les roues du robot
const int servoPin[] = {22, 23, 24, 25};
Servo servo[4];
```

```
// Définition de l'objet LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Variables globales
int tempsDuTour = 1000;
int direction = 1;
bool security = true;
```

Cette partie déclare les objets et les variables globales utilisées dans le programme, y compris les objets Servo pour les servomoteurs, un objet LCD pour l'écran LCD, et des variables de contrôle pour la direction, la sécurité et l'activation de la lame.

3.4 Prototypes de fonctions

```
// Prototypes de fonctions
void setup();
void setupTCS3200();
void setupMotors();
void setupServos();
void setupUltrasonic();
long readUltrasonicDistance(int trigPin, int echoPin);
void moveForward();
void moveBackward();
void stopMoving();
void setServos(int angle);
void avoidObstacle(int trigPinfront, int echoPinfront, int trigPinback, int echoPinback);
void controlBrushlessMotor(int speed);
void stopBrushlessMotor();
void adjustBladeHeight(int steps);
void loop();
```

Cette partie contient les prototypes de toutes les fonctions utilisées dans le programme. Les prototypes fournissent des informations sur les fonctions avant leur définition, permettant ainsi au compilateur de les reconnaître lorsqu'elles sont utilisées.

3.5 Fonction de configuration initiale (setup)

```
void setup() {
    // Initialisation de la communication série
    Serial.begin(9600);

    // Initialisation de l'écran LCD
    lcd.init();
    lcd.backlight();
    lcd.setCursor(0, 0);
    lcd.print("Robot Init...");

    // Configuration des composants
    setupTCS3200();
    setupMotors();
    setupServos();
    setupUltrasonic();

    // Initialisation du moteur pas à pas
    stepperMotor.setSpeed(200); // Vitesse (modifiable)

    // Affichage de "Prêt" sur l'écran LCD
    lcd.setCursor(0, 1);
    lcd.print("Ready!");
    delay(1000);
}

void setupTCS3200() {
```

```

for (int i = 0; i < 2; i++) {
    pinMode(S0[i], OUTPUT);
    pinMode(S1[i], OUTPUT);
    pinMode(S2[i], OUTPUT);
    pinMode(S3[i], OUTPUT);
    pinMode(outPin[i], INPUT);
    digitalWrite(S0[i], HIGH); // Set output frequency to 100%
    digitalWrite(S1[i], HIGH);
}
}

void setupMotors() {
    for (int i = 0; i < 4; i++) {
        pinMode(motorPin[i][0], OUTPUT);
        pinMode(motorPin[i][1], OUTPUT);
    }
    // Setup for stepper motor
    pinMode(stepperPin1, OUTPUT);
    pinMode(stepperPin2, OUTPUT);
}

void setupServos() {
    for (int i = 0; i < 4; i++) {
        servo[i].attach(servoPin[i]);
    }
}

void setupUltrasonic() {
    for (int i = 0; i < 4; i++) {
        pinMode(trigPin[i], OUTPUT);
        pinMode(echoPin[i], INPUT);
    }
}

```

Cette fonction `setup` réalise la configuration initiale du système, y compris l'initialisation de la communication série, de l'écran LCD, des composants, et du moteur pas à pas.

3.6 Lecture de la distance ultrasonique

```

long readUltrasonicDistance(int trigPin, int echoPin) {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    long duration = pulseIn(echoPin, HIGH);
    return duration * 0.034 / 2; // Conversion de la durée en centimètres
}

```

Cette fonction `readUltrasonicDistance` envoie un signal ultrasonique, mesure la durée du signal de retour et calcule la distance correspondante en centimètres.

3.7 Contrôle du moteur sans balais

```

void controlBrushlessMotor(int speed) {
    analogWrite(brushlessMotorPin1, speed);
    analogWrite(brushlessMotorPin2, speed);
}

void stopBrushlessMotor() {
    digitalWrite(brushlessMotorPin1, LOW);
    digitalWrite(brushlessMotorPin2, LOW);
}

```

Ces fonctions `controlBrushlessMotor` et `stopBrushlessMotor` permettent de contrôler la vitesse du moteur sans balais en envoyant un signal PWM aux broches de contrôle. La première fonction active le moteur avec une vitesse donnée, tandis que la seconde l'arrête en mettant les broches à l'état bas.

3.8 Réglage de la hauteur de la lame

```
void adjustBladeHeight(int steps) {
    stepperMotor.setSpeed(50); // Réglage de la vitesse du moteur pas à pas (modifiable)
    stepperMotor.step(steps); // Fait tourner le moteur pas à pas d'un certain nombre de pas
}
```

Cette fonction `adjustBladeHeight` ajuste la hauteur de la lame en faisant tourner le moteur pas à pas d'un nombre spécifié de pas.

3.9 Contrôle du mouvement du robot

```
void moveForward() {
    setServos(0); // Réglage des servos pour avancer
}

void moveBackward() {
    setServos(180); // Réglage des servos pour reculer
}

void stopMoving() {
    setServos(90); // Réglage des servos pour s'arrêter
}

void setServos(int angle) {
    for (int i = 0; i < 4; i++) {
        servo[i].write(angle); // Réglage de l'angle des servos
    }
}
```

Ces fonctions `moveForward`, `moveBackward`, `stopMoving` et `setServos` contrôlent les mouvements du robot en ajustant l'angle des servomoteurs pour avancer, reculer ou s'arrêter.

3.10 Évitement des obstacles

```
void avoidObstacle(int trigPinfront, int echoPinfront, int trigPinback, int echoPinback) {
    long distanceFront = readUltrasonicDistance(trigPinfront, echoPinfront);
    long distanceBack = readUltrasonicDistance(trigPinback, echoPinback);

    if (distanceFront < 20) {
        stopMoving();
        moveBackward();
        delay(500);
        stopMoving();
    } else if (distanceBack < 20) {
        stopMoving();
        moveForward();
        delay(500);
        stopMoving();
    }
}
```

Cette fonction `avoidObstacle` utilise les capteurs à ultrasons pour détecter les obstacles devant et derrière le robot. Si un obstacle est détecté à moins de 20 cm, le robot s'arrête et change de direction pour éviter la collision.

3.11 Boucle principale (loop)

```
void loop() {
    if (!security) {
        stopBrushlessMotor();
```

```

} else if (!blade_active) {
    controlBrushlessMotor(200);
}
if (direction == 1) {
    // Lire les couleurs pour le capteur avant
    long redFrequencyfront, greenFrequencyfront, blueFrequencyfront;
    readColor(S2[0], S3[0], outPin[0], redFrequencyfront, greenFrequencyfront, blueFrequencyfront);

    // Vérifier si le capteur avant détecte une autre couleur que le blanc
    if (!isWhite(redFrequencyfront, greenFrequencyfront, blueFrequencyfront, whiteThreshold)) {
        stopMoving();
        stopBrushlessMotor();
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Terrain couvert");
        return;
    } else {
        // Vérifier la présence d'obstacles et ajuster la direction si nécessaire
        long distanceFront = readUltrasonicDistance(trigPin[0], echoPin[0]);
        delay(1000);
        if (distanceFront < 20) {
            stopMoving();
            avoidObstacle();
        } else {
            long redFrequencyfront, greenFrequencyfront, blueFrequencyfront;
            readColor(S2[0], S3[0], outPin[0], redFrequencyfront, greenFrequencyfront, blueFrequencyfront);

            // Vérifier si la couleur est blanche
            if (redFrequencyfront > whiteThreshold && greenFrequencyfront > whiteThreshold && blueFrequencyfront > whiteThreshold) {
                stopMoving();
                direction = changeDirection(direction);
            } else {
                moveForward();
            }
        }
    }
} else if (direction == -1) {
    // Lire les couleurs pour le capteur avant
    long redFrequencyfront, greenFrequencyfront, blueFrequencyfront;
    readColor(S2[1], S3[1], outPin[1], redFrequencyfront, greenFrequencyfront, blueFrequencyfront);

    // Vérifier si les deux capteurs détectent le blanc
    if (!isWhite(redFrequencyfront, greenFrequencyfront, blueFrequencyfront, whiteThreshold)) {
        stopMoving();
        stopBrushlessMotor();
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Terrain couvert");
        return;
    } else {
        // Vérifier la présence d'obstacles et ajuster la direction si nécessaire
        long distanceFront = readUltrasonicDistance(trigPin[0], echoPin[0]);
        delay(1000);
        if (distanceFront < 20) {
            stopMoving();
            avoidObstacle();
        } else {
            // Lire les couleurs pour le capteur avant
            long redFrequencyfront, greenFrequencyfront, blueFrequencyfront;
            readColor(S2[1], S3[1], outPin[1], redFrequencyfront, greenFrequencyfront, blueFrequencyfront);

            // Vérifier si la couleur est blanche
            if (redFrequencyfront > whiteThreshold && greenFrequencyfront > whiteThreshold && blueFrequencyfront > whiteThreshold) {

```

```
        stopMoving();
        direction = changeDirection(direction);
    } else {
        moveBackward();
    }
}
}
```

Cette fonction `loop` constitue la boucle principale du programme. Elle contrôle les mouvements du robot en appelant les fonctions de déplacement et en vérifiant constamment les distances ultrasoniques pour éviter les obstacles. Elle inclut également des exemples d'appels pour contrôler le moteur sans balais et ajuster la hauteur de la lame.

Le robot est conçu pour naviguer sur un terrain rectangulaire délimité par une couleur blanche. Il utilise des capteurs de couleur pour détecter les limites du terrain, des capteurs à ultrasons pour éviter les obstacles, et des moteurs pour se déplacer et ajuster la hauteur de la lame de tonte. Ce document décrit en détail le fonctionnement du robot.

Avant de commencer il est nécessaire d'expliquer comment on va traiter le terrain qu'on va tendre . Le terrain va etre découpé en une sorte de grille , le robot va tendre chaque une des colonnes puis passer à la colonne suivante . La dernière ligne doit etre d'une couleur différente pour que le moteur s'arrete lorsqu'il la détecte.

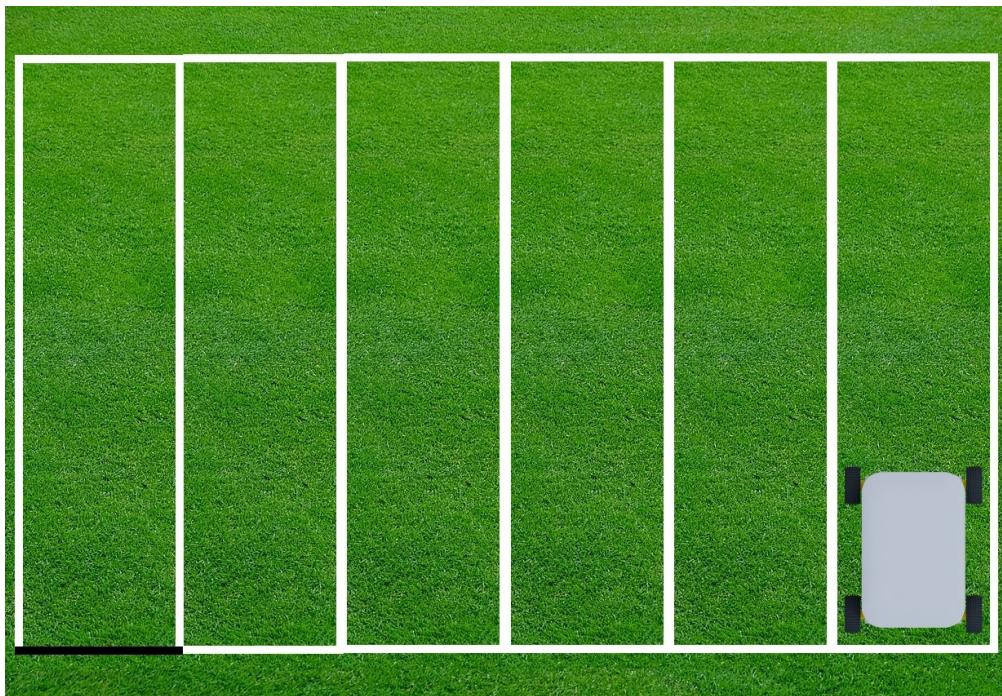


FIGURE 5 – Shématisation du terrain

1. Initialisation et Configuration

- **Initialisation des Composants :**
 - Le robot initialise l'écran LCD pour afficher des messages d'état notamment le niveau de la batterie .
 - Les capteurs de couleur (TCS3200), les capteurs à ultrasons, les moteurs, les servomoteurs et le moteur pas à pas sont configurés avec leurs pins respectifs.

2. Navigation sur le Terrain

- **Début de la Navigation :**
 - Le robot commence à naviguer en ligne droite.
 - Les moteurs sont activés pour faire avancer le robot.
 - Etant donnée que notre robot est parfaitement symétrique , la navigation se fait de la même manière que ce soit pour un aller ou pour un retour .
 - Pour un aller les capteurs situés à la face avant sont activés alors que pour un retour , ce sont ceux situés en arrière qui sont utilisés .

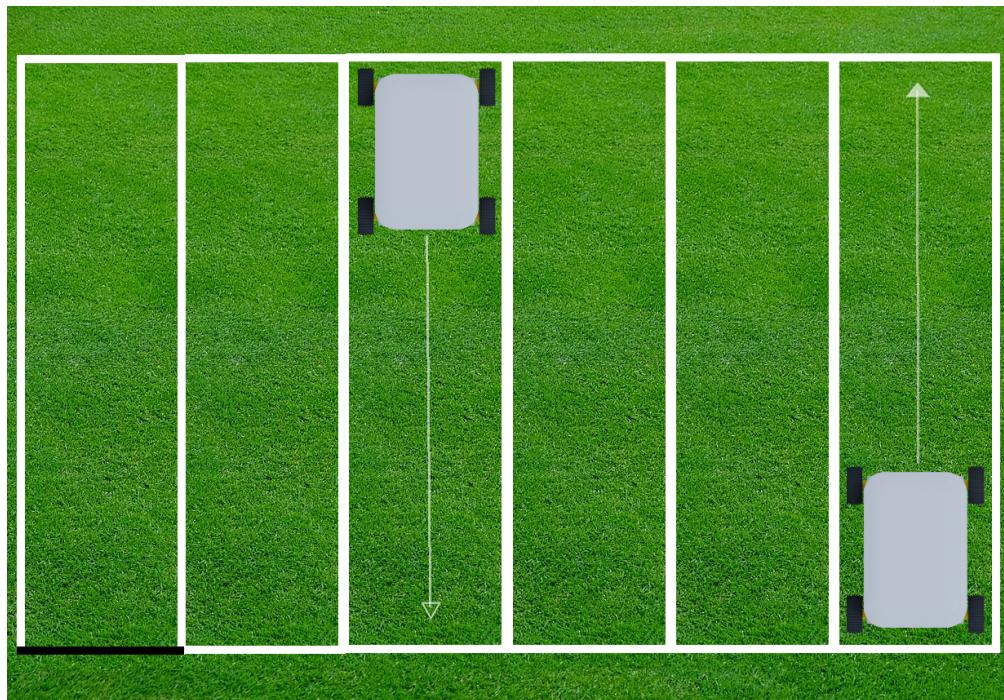


FIGURE 6 – Navigation du robot dans le terrain

3. Détection des Limites du Terrain

— Capteurs de Couleur :

- Les capteurs de couleur situés à l'avant et en arrière et sur les côtés du robot détectent la présence de la couleur blanche.
- Lorsque le capteur avant ou arrière détecte une ligne blanche, cela signifie que le robot atteint une limite du terrain.
- Dans ce cas , les roues vont subir une rotation de 90°et le robot se décale vers la colonne suivante , les roues vont ensuite retourner à leurs positions initiale.
- Le robot est maintenant prêt à tondre la nouvelle colonne.

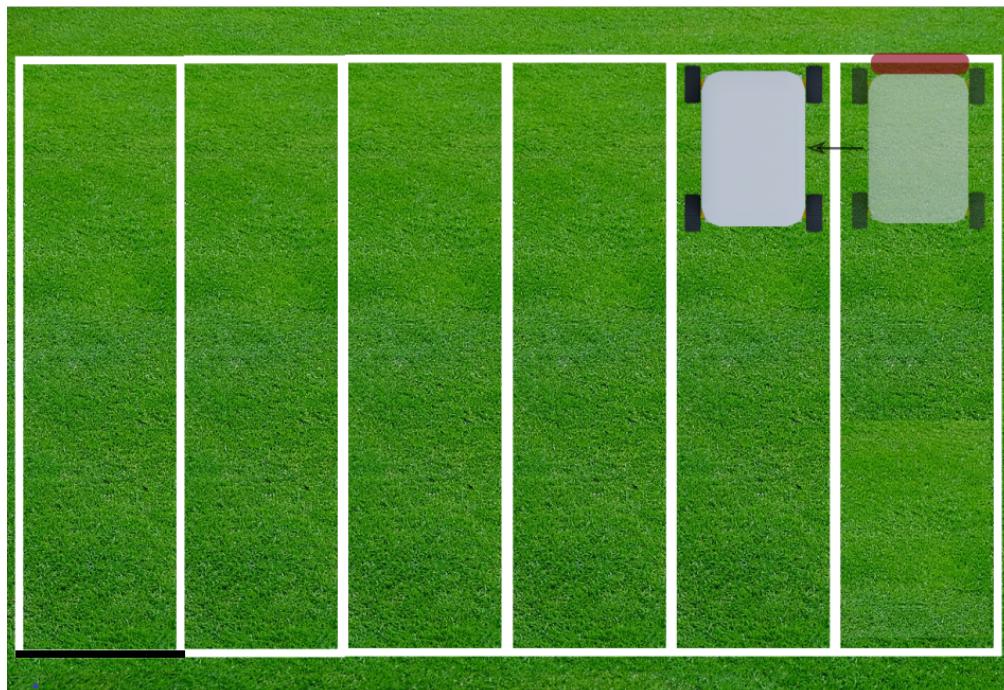


FIGURE 7 – Passage à la colonne suivante

4. Changement de Direction

— Fonction de Changement de Direction :

- Lorsque la ligne blanche est détectée, le robot s'arrête.
- Le robot utilise les servomoteurs pour tourner les roues à gauche, permettant au robot de se décaler latéralement.

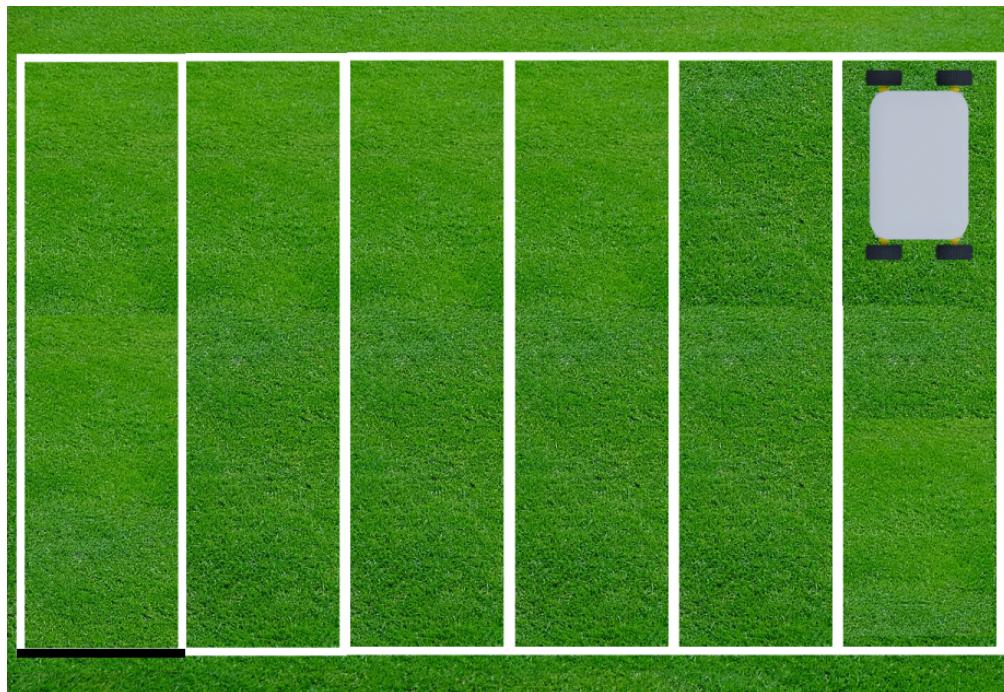


FIGURE 8 – Rotation des roues

- Le robot change de direction ça veut dire les capteurs qui sont situés en avant ne seront plus utilisés , et on utilisera les capteurs situés en arrière du robot (comme si la face arrière devient celle d'avant) . Le robot commence ensuite à se déplacer dans le sens opposé, parallèlement à sa trajectoire précédente.
- On a choisi d'adopter cette approche pour maximiser l'aire de la pelouse traitée et réduire la marge d'erreurs .

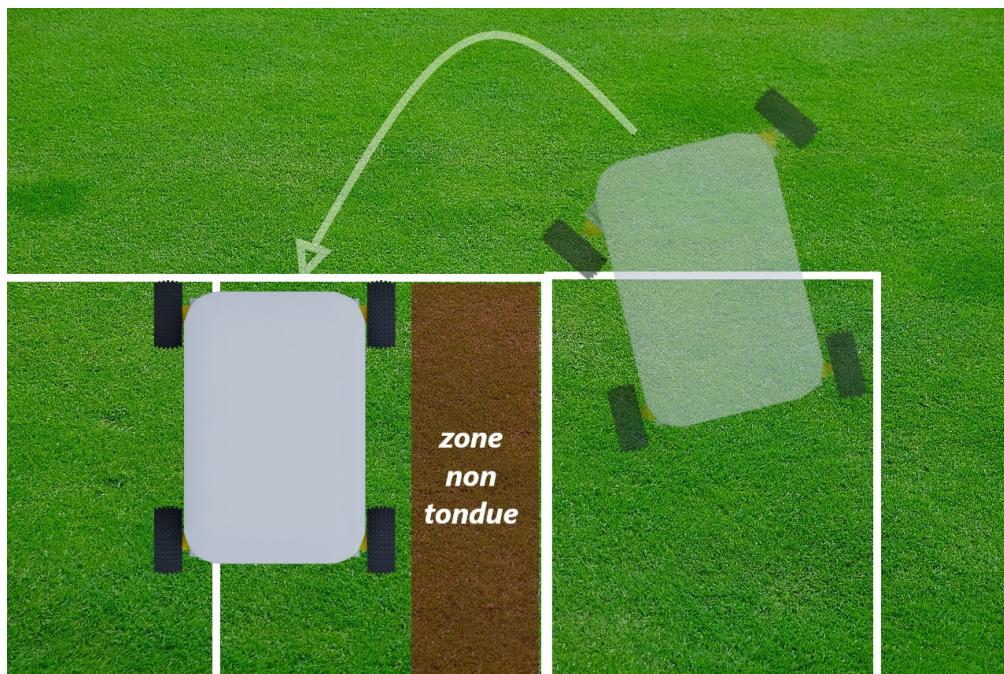


FIGURE 9 – Rotation traditionnelle du robot

5. Évitement des Obstacles

— Capteurs à Ultrasons :

- Les capteurs à ultrasons mesurent la distance entre le robot et les objets devant lui.
- Si un obstacle est détecté à une distance inférieure à un seuil prédéfini (par exemple, 20 cm), le robot s'arrête.

— Contournement de l'Obstacle :

- Les servomoteurs tournent les roues à gauche pour que le robot contourne l'obstacle.
- Le robot avance jusqu'à ce que l'obstacle soit contourné.
- Une fois l'obstacle évité, le robot revient à sa trajectoire initiale.

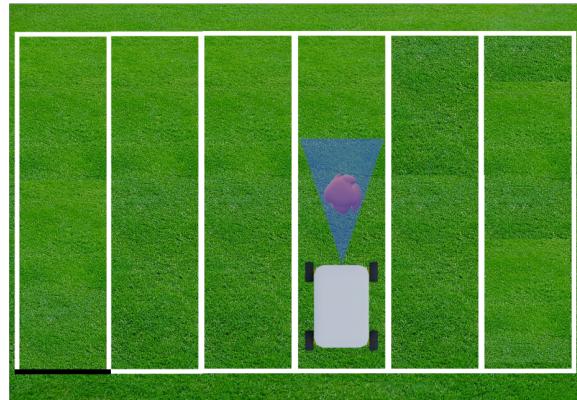


FIGURE 10 – Etape 1

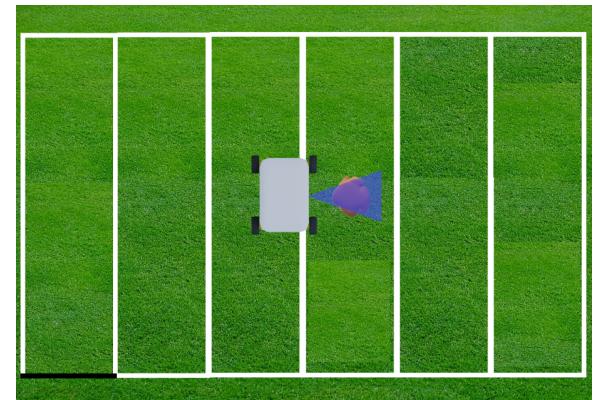


FIGURE 11 – Etape 2

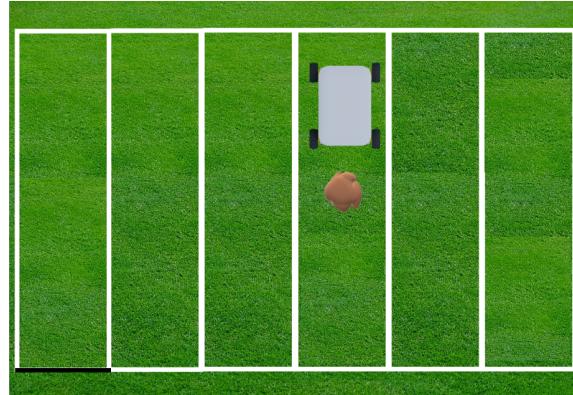


FIGURE 12 – Etape 3

FIGURE 13 – Contournement d'un obstacle

6. Gestion de la Lame

— Moteur Brushless :

- La lame du robot, actionnée par un moteur brushless, peut être activée ou désactivée en fonction de l'état de sécurité.
 - La hauteur de la lame peut être ajustée à l'aide d'un moteur pas à pas pour s'adapter aux conditions du terrain.
 - Pour ajuster la hauteur du moteur il faut choisir le nbre de pas (steps) que le moteur pas à pas doit effectuer pour atteindre la hauteur souhaitée . Expliquons de plus : La distance qu'un moteur pas à pas (stepper motor) parcourt par pas dépend de deux facteurs principaux :
 - **L'angle de pas du moteur :** Par exemple, un moteur avec un angle de 1,8° par pas nécessite 200 pas pour une rotation complète de 360° ($360^\circ / 1,8^\circ = 200$ pas).
 - **Le pas de la vis :** C'est la distance que la vis avance linéairement pour un tour complet de 360°. Par exemple, une vis avec un pas de 2 mm avance de 2 mm pour chaque rotation complète.
- Pour trouver la distance linéaire que le moteur avance par chaque pas, nous divisons la distance linéaire d'un tour complet par le nombre de pas par tour :

$$\begin{aligned} \text{Distance par pas} &= \frac{\text{Pas de la vis (distance linéaire par tour complet)}}{\text{Nombre de pas par tour}} \\ &= \frac{2 \text{ mm}}{200 \text{ pas}} \\ &= 0,01 \text{ mm par pas} \end{aligned}$$

Donc pour déplacer le moteur de tonte d'une hauteur h il faut déplacer le moteur pas à pas il faut un nombre de pas égal à

$$\begin{aligned} \text{steps} &= \frac{\text{Hauteur de déplacement}}{\text{Distance par pas}} \\ &= \frac{\text{Hauteur de déplacement}}{0,01 \text{ mm par pas}} \end{aligned}$$

7. Fin de la Navigation

— Détection de Couverture Complète du Terrain :

- Le robot continue de naviguer en zigzag, changeant de direction à chaque fois qu'il atteint une ligne blanche.
- Le processus se répète jusqu'à ce que le robot ait couvert toute la surface du terrain.

— Arrêt du Robot :

- Une fois que le robot a parcouru tout le terrain (détection de la couleur noire), il s'arrête.
- Le moteur brushless est arrêté, et le robot affiche un message indiquant que le terrain est couvert.

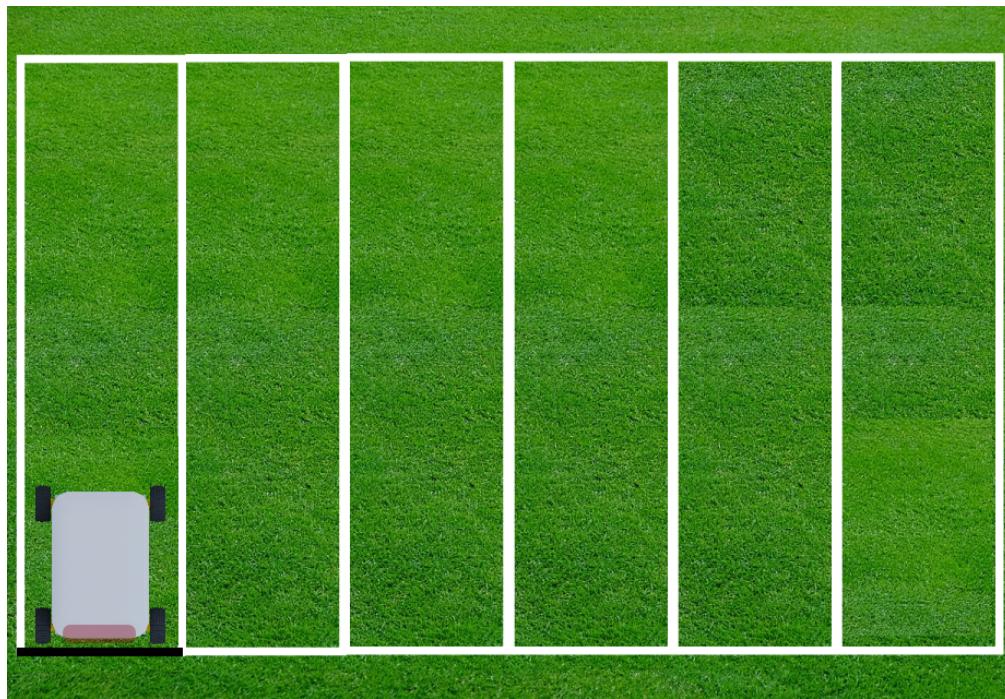


FIGURE 14 – Détection de la fin du terrain

Matériels utilisés

Le matériel utilisé dans ce code Arduino est le suivant :

- **Arduino Mega** : Pour contrôler l'ensemble du projet.
- **Écran LCD avec interface I2C (LiquidCrystal_I2C.h)** : Pour afficher des informations.
- **Servomoteurs (Servo.h)** : Utilisés pour tourner les roues de la voiture.
- **Moteur pas à pas (Stepper.h)** : Utilisé pour ajuster la hauteur de la lame.
- **Capteurs de couleur TCS3200** : Pour détecter la couleur du sol.
 - Broches utilisées : S0, S1, S2, S3, et outPin.
- **Capteurs à ultrasons** : Pour détecter les obstacles.
 - Broches utilisées : trigPin et echoPin.
- **Moteurs DC** : Pour le déplacement de la voiture.

- Broches utilisées : `motorPin`.
- **Moteurs brushless** : Pour des opérations spécifiques comme le fonctionnement de la lame.
- Broches utilisées : `brushlessMotorPin1` et `brushlessMotorPin2`.

4 Conception et analyse : Mécanique

4.1 Dessin d'ensemble

En raison de l'encombrement, le dessin d'ensemble sera divisé en différentes images et répertorié ici. La version originale sera dans un fichier séparé.

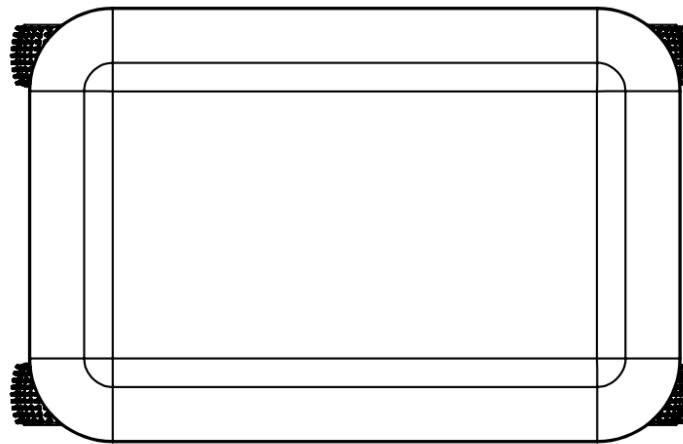
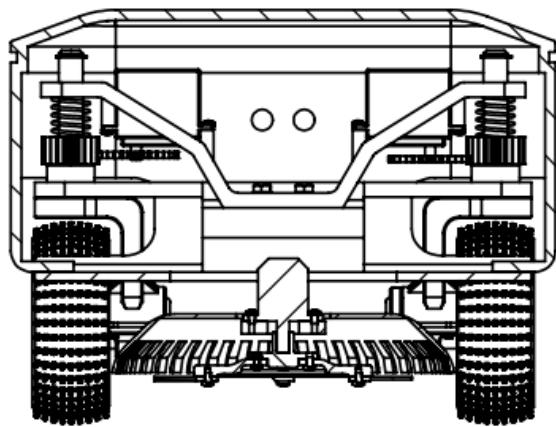


FIGURE 15 – Vue de dessus



SECTION A-A

FIGURE 16 – Section A-A

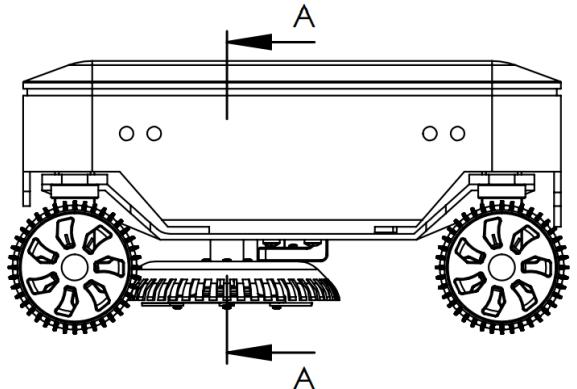


FIGURE 17 – Vue de face

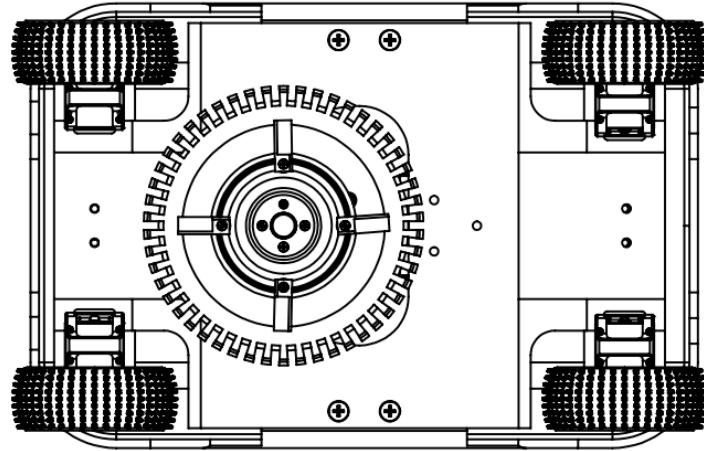


FIGURE 18 – Vue dessous

4.2 Conception des suspensions

4.2.1 Calcul de couple résistant sur les roues

Dans cette partie, nous allons essayer de déterminer l'expression du couple résistant exercé par le sol sur les roues lors du braquage des roues avec un angle ψ . On considère une roue de rayon R dont la partie inférieure est enfoncee dans le sol sous l'effet du poids du robot. La profondeur atteinte sera notée e . Lorsque les roues sont en phase de braquage, elles tournent d'un angle ψ suivant l'axe z , ce qui provoque le cisaillement du sol suivant la section colorée en rouge (voir fig. 20).

Nous commençons par exprimer α en fonction de la profondeur e et du rayon de la roue R :

$$\tan \alpha = \frac{\sqrt{R^2 - (R - e)^2}}{R - e} \quad (1)$$

$$\alpha = \arctan \left(\frac{\sqrt{2Re - e^2}}{R - e} \right) \quad (2)$$

La surface cisaillée est donnée comme suit :

$$S = \int_0^\psi \int_\alpha^\pi R^2 \sin \theta \, d\theta \, d\phi = (2R^2 - Re)\psi \quad (3)$$

Pour qu'on puisse tourner la roue suivant l'axe vertical z , il faut un couple minimal, qui est le produit de la force de cisaillement par le bras de levier $r = R \sin \left(\frac{\alpha}{2} \right)$, car on suppose que le point d'application de la force de cisaillement est donné pour l'angle $\frac{\alpha}{2}$.

Pour calculer cette force, nous utilisons la formule de la contrainte de cisaillement du sol donnée comme suit :

$$\tau = c + \sigma \tan(\phi) \quad (4)$$

où,

- τ : Contrainte de cisaillement (kPa)
- c : Cohésion du sol (kPa)
- σ : Contrainte normale (kPa)
- ϕ : Angle de frottement interne

Le robot tondeuse va naviguer généralement sur des sols cohésifs, donc on peut prendre des valeurs empiriques des paramètres c , ϕ et σ comme suit :

- $c = 11$ kPa
- $\sigma = 100$ kPa
- $\phi = 31^\circ$

Ci-dessous un tableau qui donne les valeurs de c et ϕ selon les classifications des sols (d'après <https://structville.com/angle-of-internal-friction>).

Remarque : Nous avons choisi le type *SC*. Le type de sol SC désigne un sol argileux sableux (Sand-Clay mix) selon la classification unifiée des sols (Unified Soil Classification System, USCS).

Soil Group Symbol	Cohesion (saturated) kPa	The angle of internal friction (ϕ)
GW	0	> 38°
GP	0	> 37°
GM	-	> 34°
GC	-	> 31°
SW	0	38°
SP	0	37°
SM	20	34°
SM-SC	14	33°
SC	11	31°
ML	9	32°
ML-CL	22	32°
CL	13	28°
OL	-	-
MH	20	25°
CH	11	19°
OH	-	-

FIGURE 19 – Valeurs de c et ϕ selon les classifications des sols

Comme nous avons deux surfaces cisaillées, le couple résistant devient donc :

$$C = 2rF \quad \text{avec} \quad F = \tau S \quad (5)$$

Soit finalement,

$$C = 2R\tau \sin\left(\frac{\alpha}{2}\right) (2R^2 - Re)\psi \quad (6)$$

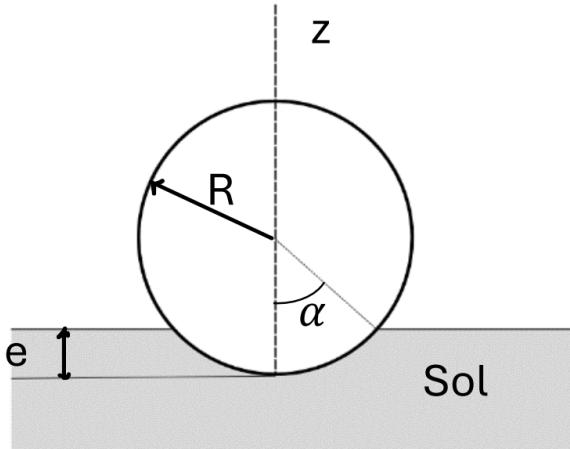


FIGURE 20 – Schéma simplifié de la roue

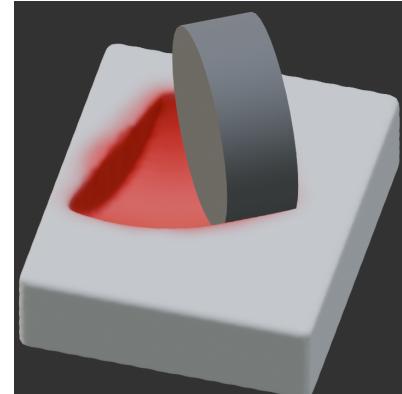


FIGURE 21 – Schéma du tracé de la roue après braquage

Le schéma cinématique du système de suspension est donné par la figure suivante :

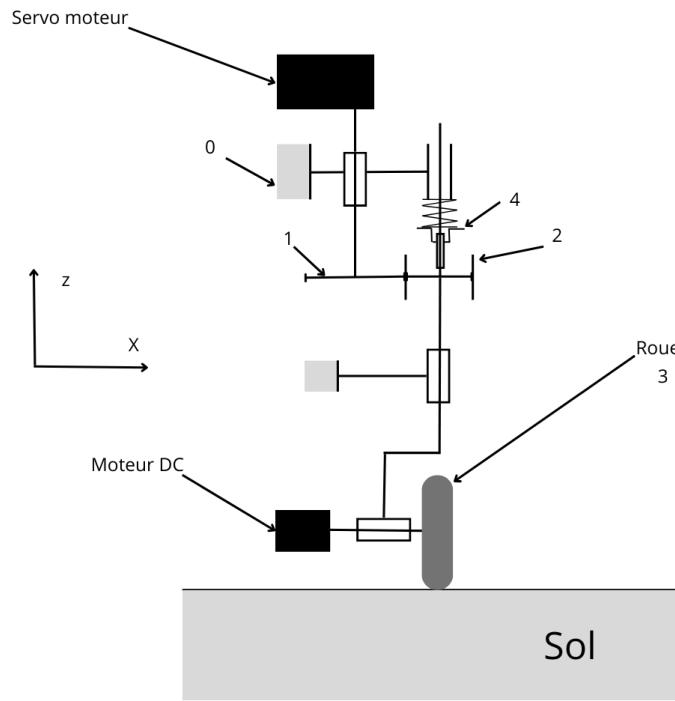


FIGURE 22 – Schéma cinématique du système de suspension

Pour le braquage des roues, nous proposons d'utiliser un servo moteur commandé par une carte Arduino. La transmission de puissance se fait par un engrenage à dentures droites (1) et (2) de nombre de dents respectivement Z_1 et Z_2 .

La puissance de sortie est donnée par :

$$P_s = \eta P_e \quad (7)$$

où,

— P_e : Puissance fournie par le servo moteur.

— η : Rendement de l'engrenage.

Or on a,

$$P_s = \omega_s C_s \quad \text{et} \quad P_e = \omega_e C_e \quad (8)$$

et

$$\frac{\omega_s}{\omega_e} = K \quad \text{avec} \quad K = \frac{Z_1}{Z_2} \quad (9)$$

On obtient alors :

$$C_s = C_e \eta K \quad (10)$$

En faisant l'égalité entre les deux expressions de couple trouvées, on peut déterminer le couple minimal requis pour le braquage de la roue :

$$C_e = \frac{2R\tau \sin\left(\frac{\alpha}{2}\right) (2R^2 - Re)\psi}{\eta K} \quad (11)$$

4.2.2 Description du fonctionnement du système

Pour garantir le bon fonctionnement du robot lors de la navigation sur des terrains difficiles, nous avons développé un système de suspension simplifié basé sur des ressorts d'amortissement. La direction des roues est contrôlée par l'un des servo-moteurs via un engrenage droit. Les roues sont entraînées séparément pour permettre une commande plus flexible et diversifiée, facilitant ainsi plusieurs types de déplacements tels que la translation latérale, la rotation du robot sur lui-même, les mouvements en diagonale, etc. Cette approche, contrairement aux robots conventionnels, facilite la navigation rapide du robot en évitant les mouvements standard (avant, arrière, rotation avec un centre éloigné) qui pourraient entraîner une perte de temps.

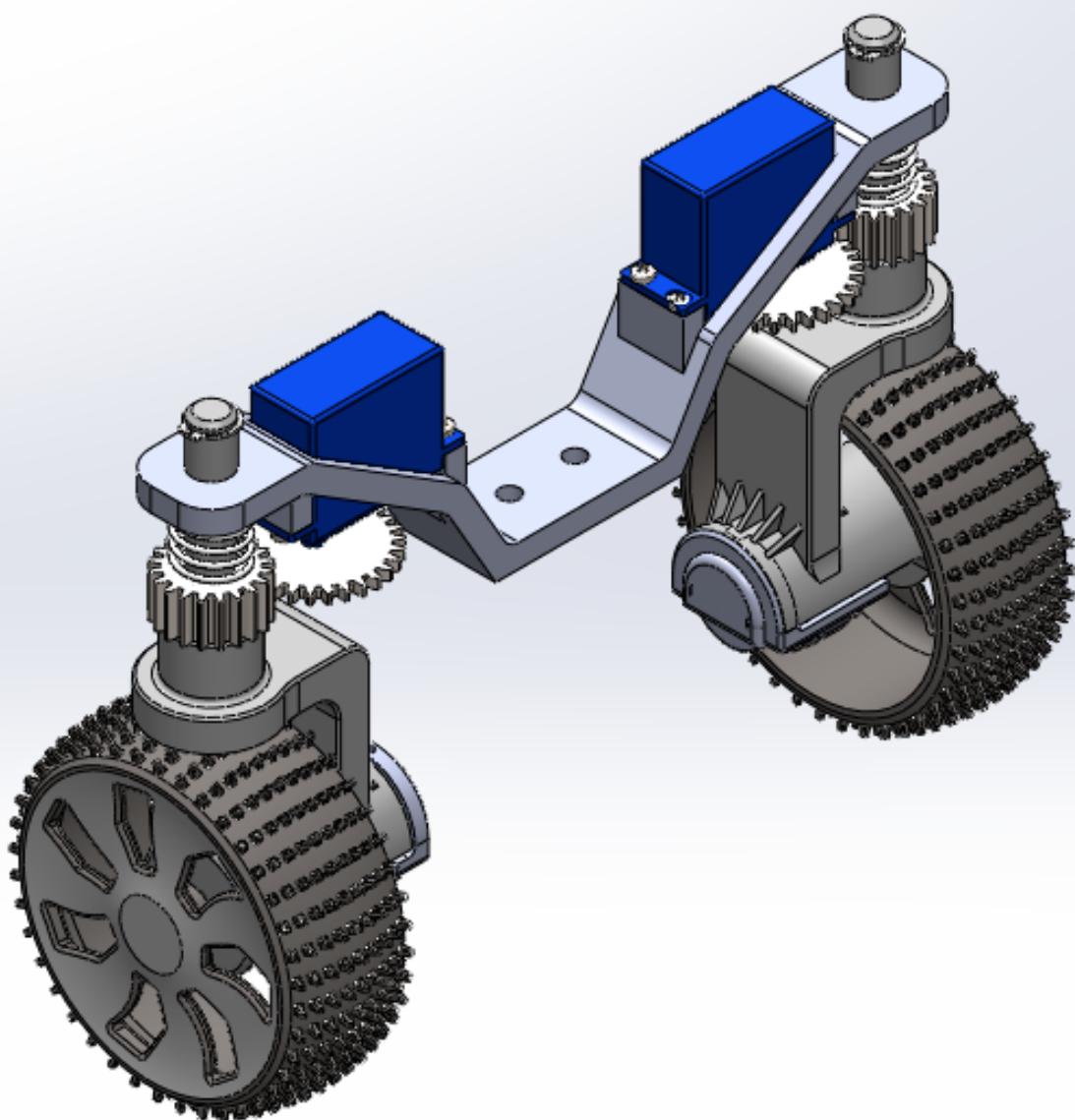


FIGURE 23 – Modèle du système de suspension réalisé avec le logiciel SolidWorks

4.3 Conception du système de découpage

L'énergie nécessaire pour couper une tige d'herbe (gazon) est $W_{tige} = 20.6mJ$.

$$N_{tige/lame} = \left[\frac{l}{d} \right] + 1 \Rightarrow w_{tige/lame} = \left(\left[\frac{l}{d} \right] + 1 \right) \times w_{tige}$$

L'énergie fournie par une lame $w_{tige/lame} = F_{lame} \times d$ Or on a

$$\begin{aligned} C &= \frac{30 \times P_{moteur}}{\pi N} \\ \Rightarrow w_{lame} &= \frac{30 \times P_{moteur} \times d}{n\pi N(R + \frac{l}{2})} \end{aligned}$$

Determination de la puissance moteur minimale :

$$P_{moteur} = \frac{np \times \pi \times (R + \frac{l}{2}) \times (\frac{l}{d} + 1)}{30 \times d}$$

Pour un moteur électrique de rendement μ on aura :

$$P_{electriqueMin} = \frac{np \times \pi \times (R + \frac{l}{2}) \times (\frac{l}{d} + 1)}{30 \times d \times \mu}$$

En appliquant cette formule aux valeurs suivantes :

- n=3
- l=30 mm
- R=80 mm
- d=2.6 mm
- $\mu = 0.85$
- N= 1000

On a besoin donc d'un moteur avec une puissance minimale :

$$P_{electriqueMin} = 483.179W$$

4.3.1 Description du fonctionnement du système

Le modèle 3D présenté ci-dessous représente la tondeuse (Système de découpe). Nous avons développé un mécanisme permettant de régler la hauteur de coupe de l'herbe à l'aide d'un système vis-écrou, contrôlé par un moteur pas à pas (non représenté). Ce mécanisme est guidé en translation par une colonne.

Dans un souci de sécurité, un couvercle de protection a été conçu. Les lames tranchantes sont libres de pivoter suivant la vis de fixation afin de prévenir tout choc accidentel. En cas de collision, les lames absorberont le choc, évitant ainsi toute usure excessive.

Le disque porte-lame est entraîné par un moteur brushless, offrant une vitesse de coupe élevée.

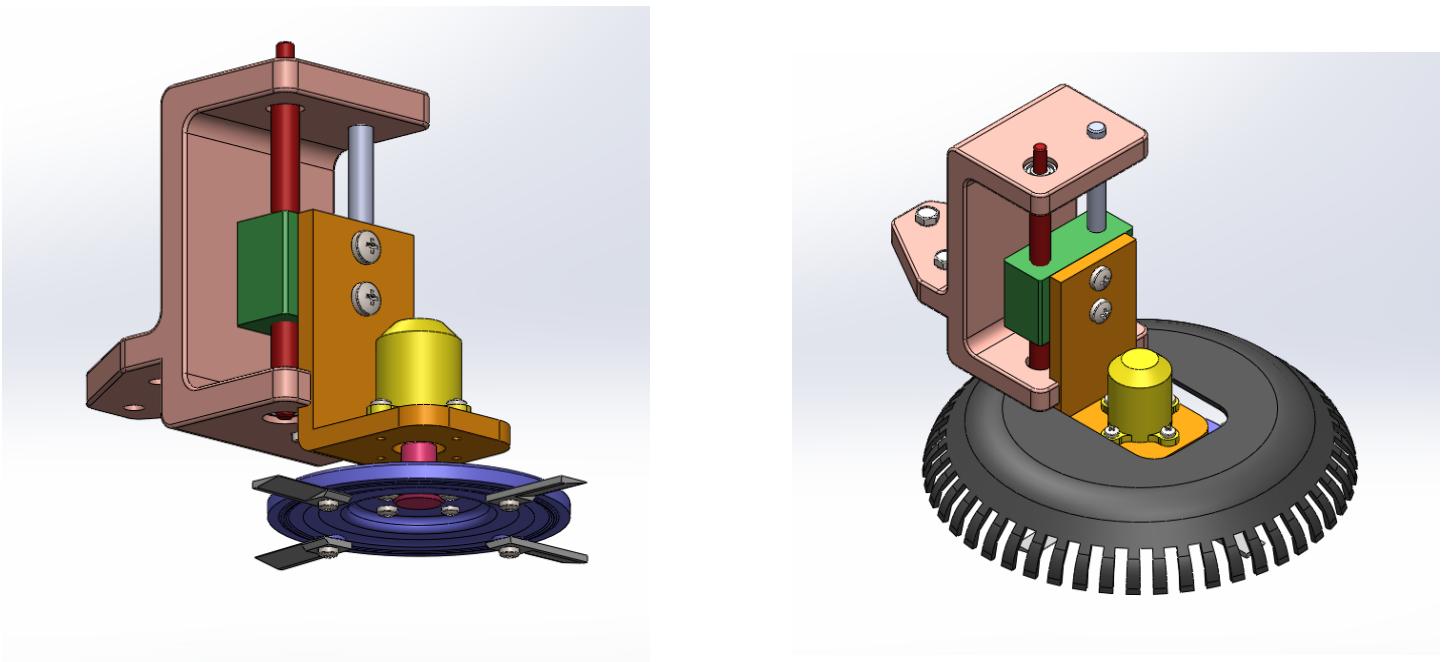


FIGURE 24 – Système de découpage avec régulation du niveau d'herbe

4.4 Conception de le châssis

Nous avons opté pour un design simple mais élégant. Les maquettes approximatives ont été conçues dans le logiciel Blender avant d'avoir la version finale conçue dans le logiciel SolidWorks.

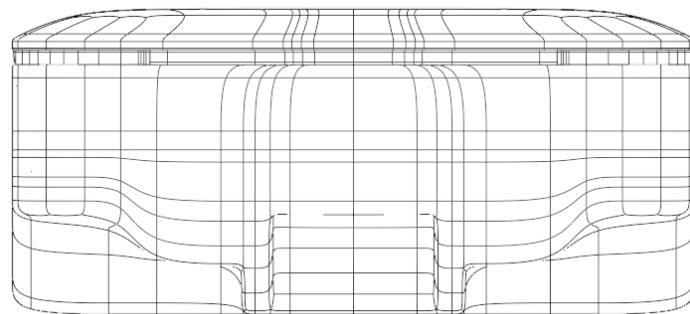


FIGURE 25 – Vue de face de la maquette



FIGURE 26 – Vue de droite de la maquette

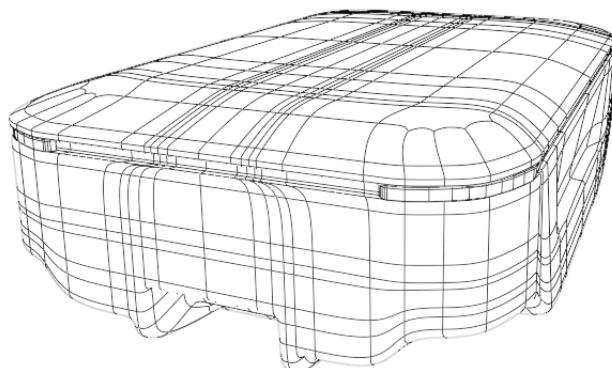


FIGURE 27 – Vue de perspective

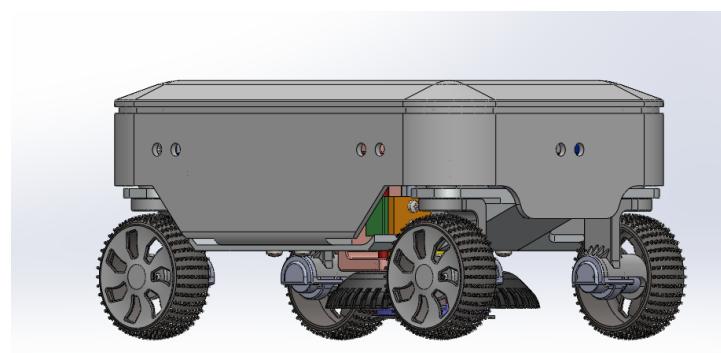


FIGURE 28 – Modèle 3D du robot réalisé avec Solidworks

5 Conception et analyse : Vision par ordinateur

Dans cette partie, nous présentons les décisions et les étapes clés concernant la conception du système de vision pour notre robot mobile, en mettant en avant le positionnement de la caméra et le choix de l'unité de calcul. Le robot doit être capable de détecter et d'interpréter son environnement pour naviguer de manière autonome. Deux options principales ont été considérées : effectuer les calculs sur un autre dispositif via une communication réseau, ou réaliser les calculs localement sur le robot.

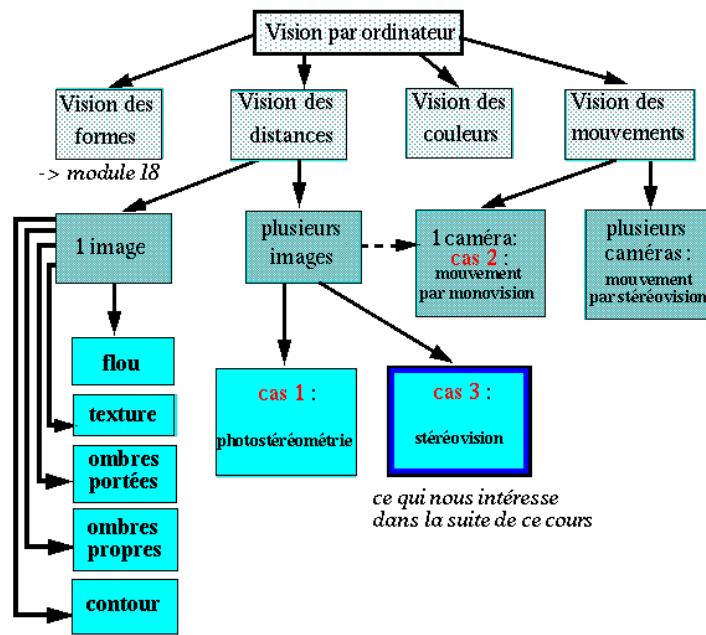


FIGURE 29 – Fonctionnement de la vision par ordinateur

5.1 Choix de l'Architecture de Calcul

Nous avons opté pour la première méthode, c'est-à-dire externaliser les calculs à un autre dispositif, car les cartes électroniques disponibles dans notre robot ne sont pas suffisamment puissantes pour gérer la complexité des calculs et la rapidité requise pour un fonctionnement efficace en temps réel.

5.2 Acquisition de Données

Pour préparer un modèle fonctionnel, nous avons collecté des données en prenant des vidéos et des photos sur les terrains de l'école. Les vidéos et images doivent être similaires à celles que le robot capturera pendant son fonctionnement. Pour ce faire, nous avons filmé à des angles de vue spécifiques, proches du sol, afin de simuler la perspective du robot.

5.3 Collaboration avec le Groupe de Commande et d'Asservissement

Nous avons discuté avec le groupe de commande et d'asservissement du robot sur la nature des données nécessaires. Il existe principalement deux types de données : 1. La détection d'obstacles proches du robot (détection d'objets). 2. La détermination des zones nécessitant l'intervention du robot (segmentation).

Nous avons également prévu de fournir des informations sur les distances par rapport aux obstacles.

5.4 Pré-Traitements des Données

Les étapes de pré-traitement des données incluent : 1. L'extraction des frames à partir des vidéos. 2. La préparation des photos au format approprié pour une utilisation avec le modèle.

5.5 Choix du Modèle de Vision par Ordinateur

Pour le modèle de vision par ordinateur, nous avons choisi d'utiliser YOLOv8 dans sa version large. Ce modèle offre une précision de détection acceptable pour notre cas d'utilisation, ainsi qu'une performance et une rapidité adéquates pour un robot nécessitant un traitement des données en temps réel.

5.6 Inference et Traitement

Les vidéos seront capturées par une caméra avec les caractéristiques suivantes : 2MP-5MP, 15-30 FPS. Les vidéos capturées seront envoyées à l'unité de calcul (PC) pour être traitées par le modèle de vision par ordinateur. Les résultats du

modèle seront ensuite transmis au robot.

5.7 Conclusion

Dans cette partie du projet, nous avons compris comment entraîner et utiliser des modèles de vision par ordinateur pour la navigation des robots. Nous avons rencontré plusieurs approches intéressantes et exploré les capacités et les limites de ces modèles. Cette expérience nous a permis de progresser dans la mise en place d'un système de vision efficace pour notre robot mobile, capable de fonctionner en temps réel tout en répondant aux exigences de précision et de performance.



FIGURE 30 – Le modèle sous fonctionnement

6 Experimentation et test

Pour vérifier la fonctionnalité de certaines parties spécifiques de notre code, nous avons utilisé le logiciel Tinkercad pour réaliser des simulations ciblées. Tinkercad, une plateforme en ligne populaire pour la modélisation et la simulation de circuits électroniques, nous a permis de recréer virtuellement quelques parties de notre projet de robotique. Nous avons intégré les composants nécessaires, tels que les microcontrôleurs, les capteurs et les actionneurs, dans l'environnement de simulation de Tinkercad. Ensuite, nous avons téléchargé et testé des portions du code pour observer le comportement du système. Cette approche nous a permis d'identifier et de corriger les erreurs de programmation, ainsi que de vérifier les interactions entre différents composants. Grâce à ces simulations ciblées, nous avons pu affiner et optimiser nos algorithmes, assurant une meilleure préparation avant de passer à l'implémentation physique du robot.

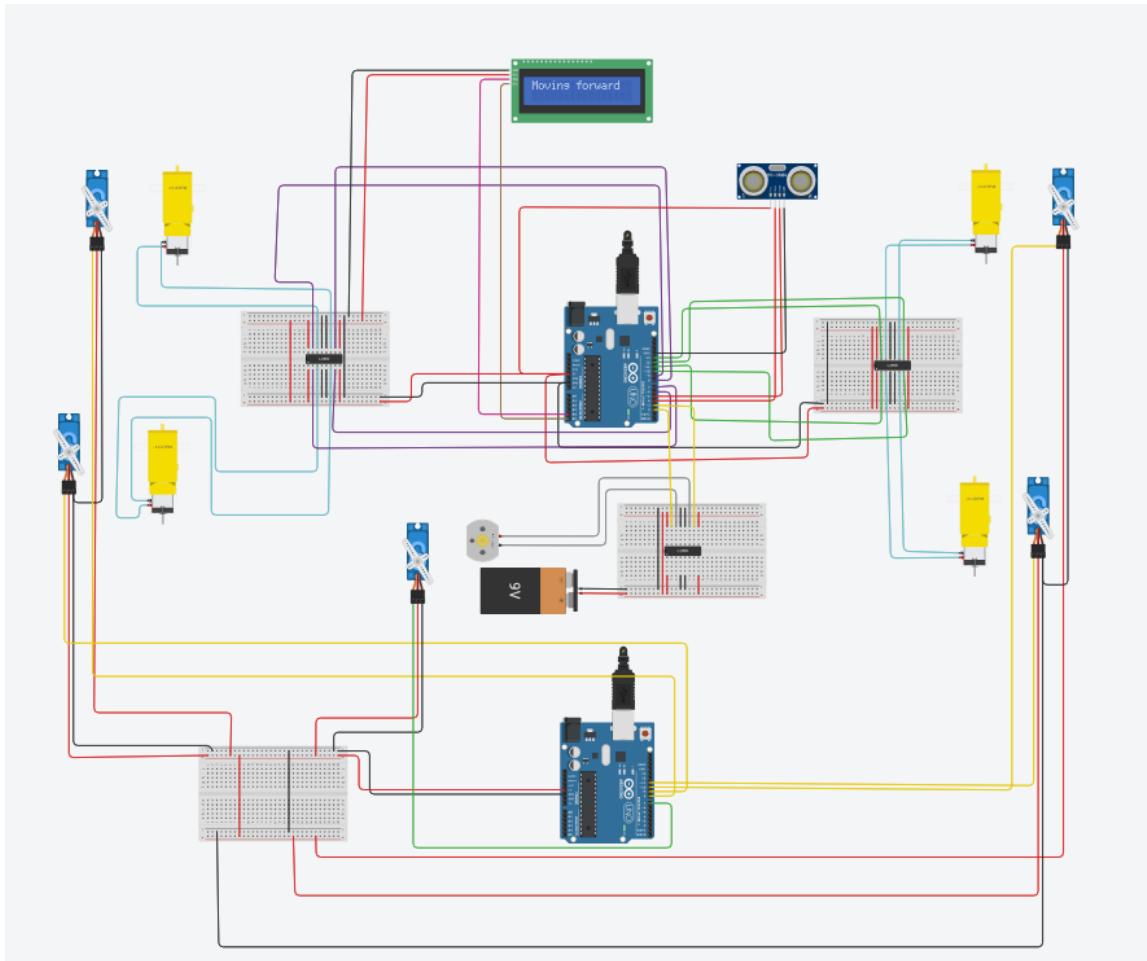


FIGURE 31 – Simulation de quelques parties du code