**Rapport de séance n°16**

GRESSARD Josselin

9 avril 2024

Robotique

Objectif de la séance : Création des supports des capteurs à ultrasons + code pour éviter les obstacles aves les capteurs

Dans un premier temps j’ai lancé l’impression des supports pour les capteurs à ultrasons

Une image contenant ordinateur, Appareil électronique, gadget, Composant d’ordinateur

Description générée automatiquementUne image contenant Appareil électronique, gadget, Appareil de saisie, périphérique

Description générée automatiquement

Une image contenant plat chaud, électroménager, projecteur, conception

Description générée automatiquement avec une confiance moyenneUne image contenant arme

Description générée automatiquement Les supports étaient des tests visant à identifier les petits problèmes et les erreurs potentielles. Dans un premier temps, le couvercle conçu pour bloquer la sortie du capteur ne remplit pas correctement sa fonction, car il n'est pas assez robuste pour le contenir. De plus, les parois sont trop fines, ce qui entraîne un fléchissement du support lorsqu'on appuie dessus. Pendant la séance, j'ai donc retravaillé le modèle pour le rendre plus solide, notamment en renforçant le couvercle avec deux vis sur les côtés. J'ai également ajouté des supports pour permettre de fixer les capteurs aux bâtis, afin qu'ils soient orientés vers la partie juste devant les roues avant du robot.

Ensuite dans un deuxième temps, je me suis occupé de la partie codage pour avoir un code qui maintenant prend en compte les 3 capteurs à ultrasons et que le robot puisse se déplacer de façon presque autonome en tout cas dans un premier temps pour éviter les obstacles aux alentours du robot.

// Constantes pour les capteurs à ultrasons

const int trigPin1 = 2; // Broche de déclenchement des ultrasons du capteur 1

const int echoPin1 = 3; // Broche de réception des ultrasons du capteur 1

const int trigPin2 = 4; // Broche de déclenchement des ultrasons du capteur 2

const int echoPin2 = 5; // Broche de réception des ultrasons du capteur 2

const int trigPin3 = 6; // Broche de déclenchement des ultrasons du capteur 3

const int echoPin3 = 7; // Broche de réception des ultrasons du capteur 3

// Définir la distance maximale à laquelle le robot considère un obstacle (en cm)

const int maxDistance = 20;

// Définir les pins des moteurs

const int motor1PinPWM = 9; // Broche PWM pour contrôler le moteur 1

const int motor1PinDIR = 8; // Broche de direction pour le moteur 1

const int motor2PinPWM = 10; // Broche PWM pour contrôler le moteur 2

const int motor2PinDIR = 11; // Broche de direction pour le moteur 2

void setup() {

  // Définir les broches des capteurs comme entrée

  pinMode(trigPin1, OUTPUT);

  pinMode(echoPin1, INPUT);

  pinMode(trigPin2, OUTPUT);

  pinMode(echoPin2, INPUT);

  pinMode(trigPin3, OUTPUT);

  pinMode(echoPin3, INPUT);

  // Définition des pins des moteurs comme sortie

  pinMode(motor1PinPWM, OUTPUT);

  pinMode(motor1PinDIR, OUTPUT);

  pinMode(motor2PinPWM, OUTPUT);

  pinMode(motor2PinDIR, OUTPUT);

}

// Fonction pour faire avancer le robot

void avancer() {

  digitalWrite(motor1PinDIR, LOW); // Mettre la direction du moteur 1 en arrière

  digitalWrite(motor2PinDIR, LOW); // Mettre la direction du moteur 2 en arrière

  analogWrite(motor1PinPWM, 255); // Faire tourner le moteur 1 à pleine vitesse

  analogWrite(motor2PinPWM, 255); // Faire tourner le moteur 2 à pleine vitesse

}

// Fonction pour tourner à gauche

void tournerGauche() {

  digitalWrite(motor1PinDIR, HIGH); // Mettre la direction du moteur 1 en avant

  digitalWrite(motor2PinDIR, LOW); // Mettre la direction du moteur 2 en arrière

  analogWrite(motor1PinPWM, 255); // Faire tourner le moteur 1 à pleine vitesse

  analogWrite(motor2PinPWM, 255); // Faire tourner le moteur 2 à pleine vitesse

}

// Fonction pour tourner à droite

void tournerDroite() {

  digitalWrite(motor1PinDIR, LOW); // Mettre la direction du moteur 1 en arrière

  digitalWrite(motor2PinDIR, HIGH); // Mettre la direction du moteur 2 en avant

  analogWrite(motor1PinPWM, 255); // Faire tourner le moteur 1 à pleine vitesse

  analogWrite(motor2PinPWM, 255); // Faire tourner le moteur 2 à pleine vitesse

}

// Fonction pour éviter les obstacles

void eviterObstacles() {

  // Faire tourner le robot à gauche

  tournerGauche();

  delay(1000); // Attendre un peu pour tourner

  // Si l'obstacle est toujours présent à gauche, tourner à droite

  if (detecterObstacle()) {

    tournerDroite();

    delay(2000); // Attendre un peu plus longtemps pour tourner à droite

  }

}

// Fonction pour détecter un obstacle

bool detecterObstacle() {

  long duree;

  int distance;

  // Envoyer une impulsion de 10 µs sur la broche de déclenchement

  digitalWrite(trigPin1, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin1, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin1, LOW);

  // Mesurer le temps écoulé pour recevoir le signal réfléchi

  duree = pulseIn(echoPin1, HIGH);

  // Calculer la distance en cm

  distance = duree \* 0.034 / 2;

  // Vérifier si la distance est inférieure à la distance maximale

  if (distance < maxDistance) {

    return true; // Obstacle détecté

  } else {

    return false; // Pas d'obstacle

  }

}

void loop() {

  // Avancer jusqu'à ce qu'un obstacle soit détecté

  while (!detecterObstacle()) {

    avancer();

  }

  // Arrêter le mouvement

  analogWrite(motor1PinPWM, 0);

  analogWrite(motor2PinPWM, 0);

  delay(1000); // Attendre 1 seconde avant de continuer

  // Éviter l'obstacle

  eviterObstacles();

}