Labyrinth Solver

Arpudha T K - CB.EN.U4CCE22008
Dhanu Ragavi S - CB.EN.U4CCE22018
Ivana Frederic - CB.EN.U4CCE22024
Computer And Communication Engineering
Amrita Vishwa Vidyapeetham Coimbatore,
Tamil Nadu, India

***Abstract*—A labyrinth in general has a single, nonbranching path leading from the entrance to the exit. This is extended in real-life scenarios where pathfinding is a crucial task. A fully autonomous robot vehicle is designed to achieve it by avoiding the obstacles in front. The Tiva board (EK-TM4C123GXL) is the microcontroller platform employed and its software counterpart, Energia IDE was used. An ultrasonic sensor supported by a servo motor, yields in the detection of the obstacle in front allowing the robot to bypass the shorter distance. Depending on the input signal received, the microcontroller redirects the robot to move in the alternate direction by actuating the motors which are interfaced via the motor driver. An optical encoder is used, to monitor the rotary speed of the wheels. All of these integrated together serve the purpose of manoeuvring the labyrinth.**

Keywords—HC-SR04 Ultrasonic Sensor, L298N Motor Driver, HC-020K Double speed measuring module photoelectric encoder (Optical Encoder), SG90 micro servo motor, EK-TM4C123GXL Evaluation board.

I. Introduction

Navigating through complex and unknown environments is a fundamental challenge in the field of robotics, necessitating advanced techniques in obstacle detection and avoidance. The labyrinth-solving robot design exemplifies a sophisticated system that combines various sensors and control algorithms to achieve autonomous navigation. This robot is equipped with an ultrasonic sensor to detect obstacles, a servo motor for directional control, and encoders to track precise movements, enabling it to traverse a labyrinth effectively. The primary objective is to allow the robot to move forward continuously, avoid collisions, and make intelligent decisions when confronted with obstacles, thereby finding the optimal path through the labyrinth.

The robot's navigation system leverages edge detection and path finding methodologies, crucial for interpreting the environment and steering clear of obstacles. The ultrasonic sensor plays a pivotal role by emitting sound waves and measuring the time it takes for the echoes to return, thus calculating the distance to potential obstacles. This real-time distance measurement enables the robot to detect obstacles accurately and promptly. Upon detecting an obstacle, the robot halts and employs its servo motor to scan the environment by looking left and right, which helps in assessing alternative paths. This scanning mechanism is critical for determining the best direction to proceed, ensuring that the robot does not remain stationary for long periods and can quickly resume its movement toward the goal.

In addition to obstacle detection, the robot incorporates a robust motor control system that facilitates various movements necessary for labyrinth navigation. The system allows the robot to move forward, and backward, and make precise turns to the left or right. When an obstacle is detected, the robot can stop and, based on the servo motor's environmental scan, decide whether to turn left, turn right, or move backward to find a clearer path. This decision-making process is augmented by the feedback from encoders, which provide accurate information about the robot's movements and position. The encoders track the rotation of the wheels, ensuring that the robot's movements are precise and controlled, which is essential for navigating through the narrow and winding paths of a labyrinth.

The integration of these components—ultrasonic sensor, servo motor, motor control system, and encoders—creates a comprehensive solution for autonomous navigation. The ultrasonic sensor's ability to detect obstacles, combined with the servo motor's scanning function and the precise control afforded by the motor control system and encoders, enables the robot to navigate complex environments efficiently. This project demonstrates the application of advanced sensor fusion and control algorithms in mobile robotics, highlighting the potential for robots to operate autonomously in unknown and dynamically changing environments. The successful implementation of this labyrinth-solving robot underscores the advancements in autonomous robotic navigation and sets the stage for further developments in the field.

II. LITERATURE SURVEY

The development of autonomous robots with advanced navigation and obstacle avoidance capabilities has been a focal point of research and innovation. Various projects have explored different methodologies and technologies to enhance the performance and functionality of these robots.

The "Line Follower and Obstacle Avoidance Bot Using Arduino," designed by Aamir Attar, Aadil Ansari, Abhishek Desai, Shahid Khan, and Dipashri Sonawale, is one such project aimed at creating an autonomous robot capable of intelligently detecting obstacles and navigating accordingly. This robot utilizes an Arduino platform to follow a predefined line while also avoiding obstacles that come in its path. The integration of line-following and obstacle avoidance functionalities

demonstrates a significant advancement in robotic navigation systems. This project highlights the potential for robotic machinery to replace skilled labor in various applications, offering benefits such as increased efficiency, better accuracy, and lower costs, particularly in scenarios like patient handling and care where precision and reliability are paramount.

Another important contribution to this field is the "Obstacle-Avoiding Robot with IR and PIR Motion Sensors" by Aniket D. Adhvaryu et al. This project proposed a versatile wheeled autonomous platform that was not limited to a specific task but could be adapted for educational, research, and industrial purposes. The robot employs IR and PIR sensors to detect obstacles and human presence, respectively, showcasing the flexibility of sensor integration in robotic platforms. This project also served an educational purpose, allowing students to gain hands-on experience with microcontroller programming, sensor characteristics, motor driving circuits, and signal conditioning circuit design. The findings from this study emphasized that PIR sensors exhibit higher sensitivity compared to IR sensors in detecting human presence, suggesting their potential for various applications where human detection is crucial.

The "Obstacle Avoidance Robotic Vehicle Using Ultrasonic Sensor, Android, and Bluetooth for Obstacle Detection," developed by Vaghela et al., explored the use of an ultrasonic sensor in conjunction with an Android-based smartphone interface for robot control. This project reviewed various methodologies for wireless gesture control and highlighted the advantages of using Android devices, such as their user-friendly interface, lightweight design, and portability. The study concluded that Android-based control systems offer superior functionality compared to traditional technologies like programmable gloves and static cameras. The research emphasized the need for further exploration in wireless gesture control, particularly for applications in home automation, assistive devices, and interactive displays, showcasing the versatility and potential of integrating smartphone technology with robotic systems.

Paul Kinsky and Quan Zhou's "Obstacle Avoidance Robot" introduced additional mechanical components to the robot, including a laptop holder and a camera holder, to enhance its functionality. The robot utilized an AT89S52 development board to control its motors and employed lowcost cameras for computer vision calibration. The integration of computer vision with motor control allowed the robot to navigate more effectively and avoid obstacles with greater precision. The project demonstrated the successful establishment of serial communication between the laptop and the development board, enabling real-time control and monitoring of the robot's movements. This integration of various technologies underscored the importance of combining computer vision, motor control, and communication systems to develop more capable autonomous robots.

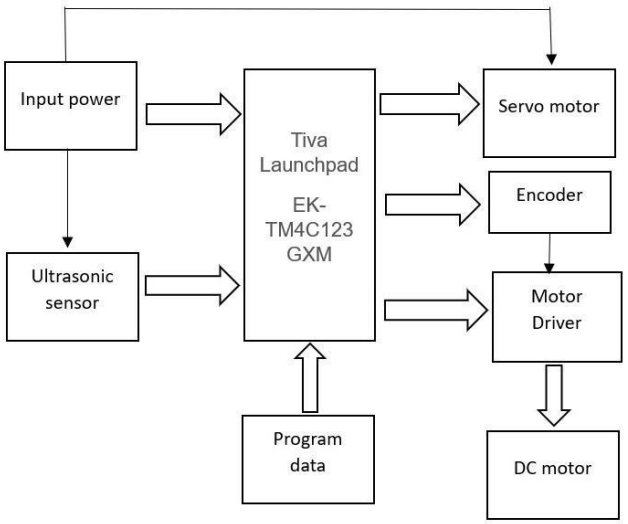
Faiza Tabassum et al. developed an "Obstacle Avoidance Car" that successfully detected and avoided obstacles using a combination of IR sensors and simple steering algorithms. The project focused on reducing the turning radius to navigate the vehicle more effectively. The robot incorporated timer interrupts for IR pulse generation, obstacle detection using IR transceivers, and a steering system using LEGO and servos. The successful implementation of these components highlighted the feasibility of creating efficient obstacle avoidance systems using straightforward and cost-effective techniques. This project demonstrated that even simple algorithms and basic components could be used to achieve reliable obstacle avoidance, making such systems accessible for various applications.

Collectively, these studies and projects illustrate the diverse approaches and technological advancements in the field of autonomous robotics. They highlight the critical role of integrating various sensors, control algorithms, and user interfaces in developing robots capable of navigating complex environments autonomously. The continuous progression in this field is paving the way for more sophisticated and versatile robotic systems, with broad applications ranging from industrial automation to personal assistance, education, and research. These advancements not only improve the functionality and efficiency of robotic systems but also open new avenues for their application in everyday life.

III. METHODOLOGY

The methodology for developing the labyrinth-solving robot involves a detailed approach that integrates both hardware and software components to ensure efficient and autonomous navigation. The system uses various sensors and control mechanisms to detect and avoid obstacles while navigating through a labyrinth. The fundamental block diagram for the system implementation is illustrated in Figure 1.

A. Block Diagram Overview



B. Ultrasonic Sensor for obstacle detection

The robot utilizes an HC-SR04 ultrasonic sensor for obstacle detection. This sensor employs the principle of sonar to achieve

non-contact distance measurement within a range of roughly 2 cm to 400 cm. Unaffected by lighting conditions or dark materials, the HC-SR04 emits high-frequency sound waves that reflect upon encountering an obstacle. The time it takes for the echo to return is used to calculate the distance to the obstacle. The operation sequence involves initialization, trigger pulse generation, echo reception, pulse duration measurement, and distance calculation. During initialization, the Trigger and Echo pins are set to low while the robot moves forward. A 10-microsecond pulse is then sent to the Trigger pin to initiate the measurement. When the sound wave encounters an object and reflects back, the Echo pin transitions high. The pulseIn function is employed to measure the duration of the Echo pin's high state, which is directly proportional to the obstacle distance. This measured duration is subsequently converted into a distance value. If no echo is received within a predefined timeout period, the function terminates, signifying the absence of obstacles within the sensor's maximum range. The ultrasonic sensor consists of a multi-vibrator, which is fixed at its base. The multivibrator is a combination of a resonator and a vibrator the ultrasonic waves generated by the vibration are delivered to the resonator. The ultrasonic sensor actually consists of two parts: the emitter which produces a 40 kHz sound wave and the detector which detects a 40 kHz sound wave and sends an electrical signal back to the microcontroller. HC-SR04 ultrasonic sensors are used which consist of 4 pins VCC, Trigger, Echo, and GND.

Features of Ultrasonic Sensor:

- Compact and lightweight
- High sensitivity and high pressure
- High reliability
- Power consumption of 20mA
- Pulse in/out communication
- Narrow acceptance angle
- Provides exact, non-contact separation
- estimations within 2cm to 3m
- The explosion point LED shows estimations in advancement
- 3-pin header makes it simple to connect utilizing

C. Movement Control

The robot's movement is governed by a logic system that integrates data from both the ultrasonic sensor and the optical encoder, ensuring precise navigation in response to detected obstacles. The logic for movement is as follows:

1. Distance Detection:

- The ultrasonic sensor detects obstacles in the robot's path and provides distance data to the Tiva microcontroller.
- Simultaneously, the optical encoder monitors the rotation of the robot's wheels, providing feedback on its movement.

2. Moderate Distance Handling:

- If the detected distance is moderate, indicating the presence of an obstacle within a certain range, the control algorithm activates.
- The robot reduces its speed, adjusting the motor driver's output to slow down the rotation of the wheels.
- Depending on the proximity of obstacles, the robot prioritizes avoiding obstacles on its left side. If an obstacle is detected on the left, it initiates a right turn to navigate around it while maintaining forward momentum.

3. Short Distance Maneuvering:

- In the event of a short-distance detection, signaling an immediate obstacle, the control algorithm triggers a rapid response.
- The robot significantly slows down, and if necessary, reverses its direction to create space between itself and the obstacle.
- Following the backward movement, the robot evaluates its surroundings once more using the ultrasonic sensor and optical encoder feedback to determine the most viable path.
- It then executes a precise left or right turn, ensuring a safe trajectory around the detected obstacle.

4. Integration with Optical Encoder and Motor Driver:

- The optical encoder continuously monitors the rotation of the robot's wheels, providing real-time feedback to the Tiva microcontroller.
- This feedback is crucial for maintaining accurate control over the robot's movement speed and direction, ensuring smooth navigation around obstacles.
- The Tiva microcontroller processes the encoder data alongside the ultrasonic sensor readings, orchestrating precise adjustments to the motor driver's output signals.
- The motor driver translates these control signals into corresponding adjustments in motor speed and direction, facilitating seamless maneuvering as dictated by the obstacle avoidance logic.

D. Integration with Tiva micro controller:

The seamless integration between the Tiva microcontroller and the robot's components ensures efficient operation and precise control over its movements.

1. Tiva Microcontroller Connectivity:

- The Tiva microcontroller serves as the central processing unit, orchestrating the robot's behaviour through effective interfacing with its various components.
- Establishing connections between the Tiva microcontroller and peripheral devices, including the ultrasonic sensor, motor driver, servo motor, and encoder sensors, forms the backbone of the integration process.

2. Motor Driver Connections:

- Critical to the robot's locomotion, the motor driver board interfaces with the Tiva microcontroller to regulate the DC motors responsible for propulsion.
- Utilizing designated pins (e.g., PA_2, PA_3, PA_4, PA_5), the Tiva microcontroller sends control signals to the motor driver, dictating the direction and speed of the motors for forward, backward, left, and right movements.

3. Servo Motor Interface:

- Precision control over directional adjustments and obstacle detection is achieved through the integration of a servo motor with the Tiva microcontroller.
- By connecting the servo motor's signal pin to the Tiva microcontroller's PD_1 pin, the system enables finetuned manipulation of the servo motor's orientation, enhancing navigational capabilities.

4. Ultrasonic Sensor Connectivity:

- Essential for obstacle detection, the ultrasonic sensor seamlessly interfaces with the Tiva microcontroller, facilitating accurate distance measurements and obstacle avoidance.
- Through connections to the trigger and echo pins (PF_1 and PF_2), the Tiva microcontroller initiates pulse emission and processes echo signals to determine object distances, enabling real-time navigation adjustments.

5. Encoder Sensor Integration:

- Real-time monitoring of wheel rotation and movement feedback is achieved through the integration of optical encoder sensors into the system.
- By connecting encoder outputs (ENCODER_A and ENCODER_B) to the Tiva microcontroller's PD_2 and PD_3 pins, respectively, the system captures precise data on wheel rotation, enabling adaptive navigation strategies.

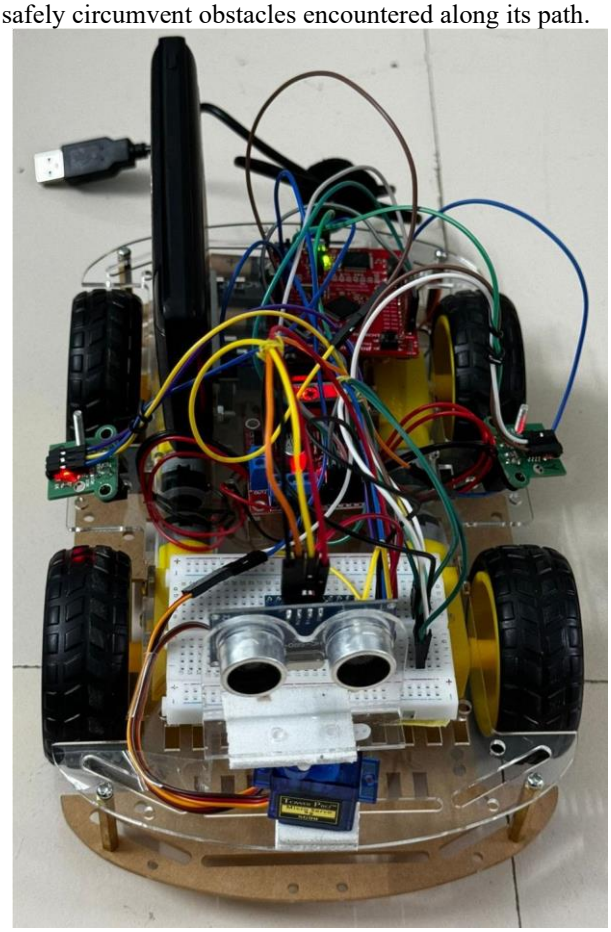
6. Control Algorithm Execution:

- Sensor Data Processing: The algorithm continuously reads distance measurements from the ultrasonic sensor, providing real-time inputs on the proximity of obstacles.
- Decision-Making: Leveraging the distance data, the algorithm determines the appropriate course of action, whether it entails moving forward, turning left, turning right, or reversing to avoid collisions.
- Motor Control: Upon making a decision, the algorithm sends precise signals to the motor driver, dictating the speed and direction of the robot's movements to execute the chosen action seamlessly.

7.System Testing and Calibration:

- Calibration: Initial calibration of the ultrasonic sensor and servo motor ensures accurate distance measurements and precise directional control, laying the foundation for reliable navigation.
- Performance Testing: The robot undergoes extensive testing in diverse labyrinth configurations to assess its obstacle detection and avoidance prowess. Various scenarios are simulated to gauge the robot's adaptability and responsiveness.
- Algorithm Optimization: Feedback from performance testing drives iterative refinements to the control algorithms, aiming to enhance navigation accuracy and efficiency. Algorithmic adjustments are made based on empirical data and observed behaviours.
- Final Validation: Comprehensive validation tests ascertain that the robot meets design specifications and exhibits consistent performance across different labyrinth setups. The culmination of testing affirms the robot's reliability and suitability for autonomous navigation tasks.

This integrated approach ensures that the robot effectively navigates its environment, dynamically adjusting its movement strategy based on real-time sensor data and encoder feedback to safely circumvent obstacles encountered along its path.



E. Pin Configuration

Component	Pin on Component	Pin on Launchpad	Pin Definition in Code
Servo Motor	Signal	PD_1	`servoPin`
Ultrasonic	TRIG	PF_1	`trigPin`
Ultrasonic	ECHO	PF_2	`echoPin`
LED	Anode	RED_LED	`ledPin`
Motor Left	IN1	PA_2	`MOTOR_IN1`
Motor Left	IN2	PA_3	`MOTOR_IN2`
Motor Right	IN3	PA_4	`MOTOR_IN3`
Motor Right	IN4	PA_5	`MOTOR_IN4`
Encoder	A	PD_2	`ENCODER_A`
Encoder	B	PD_3	`ENCODER_B`

IV. Application

1. Warehouse Automation: Automates the movement of goods and materials in warehouses.

2. Healthcare Assistance: Delivers supplies and medications in hospitals and care facilities.

3. Agricultural Automation: Navigates through crops for monitoring, planting, and harvesting.

4. Surveillance and Security: Patrols premises and provides real-time surveillance in security systems.

5. Disaster Management: Navigates through debris in search and rescue missions.

6. Education and Research: Teaches robotics, programming, and automation in educational institutions.

7. Automated Guided Vehicles (AGVs): Transports materials and products in manufacturing and assembly lines.

8. Smart Home Systems: Automates tasks like floor cleaning, pet monitoring, and home security.

V. Result

The obstacle avoidance robot using the Tiva microcontroller demonstrated reliable navigation. The robot moved forward, using an ultrasonic sensor mounted on a servo motor to detect and avoid obstacles. When an obstacle was encountered, the sensor scanned for clear paths, allowing the robot to adjust its direction accordingly. The system's precise distance measurements and responsive control ensured efficient obstacle avoidance, confirming the effectiveness of the design.

VI. Conclusion and Future Scope

The project successfully built a mobile robot prototype for obstacle avoidance using an HC-SR04 ultrasonic sensor and a Tiva microcontroller. The robot effectively navigates its environment, detecting obstacles (2 cm - 400 cm) and adjusting its path through a basic control algorithm. This project lays the groundwork for further exploration in areas like sensor fusion, advanced navigation algorithms, machine learning integration, wireless communication, and improved mechanical design, paving the way for more robust and intelligent autonomous mobile robots.

References

[1] G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)

[2] Aniket D Adhvaryu et al “ Obstacle -avoiding robot with IT and PIR motion sensor “ IOP Conference series: Materials science and engineering, vol. A247, pp. 529-551, April 2005.

[3] Vaghela Ankit1, Patel Jigar2, Vaghela Savan3 “Obstacle Avoidance Robotic Vehicle Using Ultrasonic Sensor, Android And Bluetooth For Obstacle Detection” International Research Journal of Engineering and Technology (IRJET), vol. A247, pp. 29-32, 2005.

[4] Paul Kinsky,Quan Zhou “Obstacle Avoidance Robot” Worcester Polytechnic institute.

[5] Bhagya shree S R , Manoj kollam “Zigbee Wireless Sensor Network For Better Interactive Industrial Automation” , proc.of IEEE ICoAC2011,pp 304-308,2011.

[6] Kirit Bhagat, Sayali Deshmukh, Shraddha Dhonde, Sneha Ghag, : Obstacle Avoidance Robot”, Bachelor of computer engineering , IJSETR, volume 5, issue 2, February 2016.

[7] Jitihsha Agrawal, “Solar Operated low cost Obstacle avoidance Robot” , Department of extc, YMCA university of science and technology (state Government university) Faridabad, IJSRD, volume 3, issue 7 2015 ISSN 2321-0613.

[8] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[9] I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[10] K. Elissa, “Title of paper if known,” unpublished.

[11] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[12] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[13] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

Appendix

```
#include <energia.h>

#include <Servo.h>

// Create a Servo object
Servo myServo;

// Define pin for the servo signal
int servoPin = PD_1;  // PWM capable pin (Check the pin map for the Launchpad)
```

```
// Define pins for ultrasonic sensor
int trigPin = PF_1;       // TRIG pin connected to PF1
int echoPin = PF_2;       // ECHO pin connected to PF2

// Define pin for inbuilt LED
int ledPin = RED_LED;     // Inbuilt LED pin (assuming it's connected to RED_LED pin)
float minDistance_cm = 15.0; // Minimum distance threshold in cm
float minDist_side = 40.0;   // Minimum distance threshold in cm (side)

// Define pins for motor control
#define MOTOR_IN1  PA_2  // Left Int1 (PA2)
#define MOTOR_IN2  PA_3  // Left Int2 (PA3)
#define MOTOR_IN3  PA_4  // Right Int3 (PA4)
#define MOTOR_IN4  PA_5  // Right Int4 (PA5)

// Define pins for encoder
#define ENCODER_A  PD_2  // Encoder output A connected to PD2
#define ENCODER_B  PD_3  // Encoder output B connected to PD3

volatile long encoderCountLeft = 0; // Encoder count for the left motor
volatile long encoderCountRight = 0; // Encoder count for the right motor
int encoderPinAState = LOW;
int encoderPinBState = LOW;

// Timing for task intervals
unsigned long previousMillisTask1 = 0;
const long intervalTask1 = 100;  // Task 1 interval (100 milliseconds)

// PID constants
float Kp = 0.1;  // Proportional gain

// Function prototypes
void task1();
void moveForward();
void stop();
void turnRight();
void turnLeft();
void moveBackward();
void readEncoder();
void adjustMotorSpeed();

void setup() {
  Serial.begin(9600);

  // Initialize pins for ultrasonic sensor
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);

  // Initialize pins for motor control
  pinMode(MOTOR_IN1, OUTPUT);
  pinMode(MOTOR_IN2, OUTPUT);
  pinMode(MOTOR_IN3, OUTPUT);
  pinMode(MOTOR_IN4, OUTPUT);

  // Initialize encoder pins
  pinMode(ENCODER_A, INPUT_PULLUP);
  pinMode(ENCODER_B, INPUT_PULLUP);

  // Ensure motors are off
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, LOW);
  digitalWrite(MOTOR_IN3, LOW);
  digitalWrite(MOTOR_IN4, LOW);

  // Attach the servo to the pin
  myServo.attach(servoPin);

  // Set initial position
  myServo.write(90);  // Middle position (90 degrees)
}

void loop() {
  unsigned long currentMillis = millis();

  // Task 1: Runs every 100 milliseconds
  if (currentMillis - previousMillisTask1 >= intervalTask1) {
    previousMillisTask1 = currentMillis;
    task1();
  }

  // Read the encoder in each loop iteration
  readEncoder();

  // Adjust motor speeds to maintain straight movement
  adjustMotorSpeed();

  // Continue moving forward if no obstacle is detected
  moveForward();
}

// Task to read the ultrasonic sensor and control the motor
void task1() {
  // Generate 10-microsecond pulse to TRIG pin
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure duration of pulse from ECHO pin
```

```
  float duration_us = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  float distance_cm = 0.017 * duration_us;

  // Print the distance to Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");

  // Print encoder counts to Serial Monitor
  Serial.print("Encoder Count Left: ");
  Serial.println(encoderCountLeft);
  Serial.print("Encoder Count Right: ");
  Serial.println(encoderCountRight);

  // Check if the distance is less than the minimum threshold
  if (distance_cm < minDistance_cm) {
    // If obstacle is closer than the threshold, turn on the LED
and stop the robot
    digitalWrite(ledPin, HIGH);
    stop();
    delay(1000);  // Stop for 1 second

    while (true) {
      // Check the right direction
      myServo.write(0); // look right
      Serial.println("Servo position: 0 degrees");
      delay(1000);  // Wait for 1 second
      digitalWrite(trigPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(trigPin, LOW);
      float durationRight_us = pulseIn(echoPin, HIGH);
      float distanceRight_cm = 0.017 * durationRight_us;

      if (distanceRight_cm >= minDist_side) {
        // Turn right if the path is clear on the right
        turnRight();
        delay(750);  // Turn right for 1 second
        stop();
        delay(1000);  // Stop for 1 second
        // Move servo to 90 degrees
        myServo.write(90); // center
        Serial.println("Servo position: 90 degrees");
        delay(1000);  // Wait for 1 second
        moveForward();
        return;  // Exit the loop
      } else {
        // Check the left direction
        myServo.write(180); // look left
        Serial.println("Servo position: 180 degrees");
        delay(1000);  // Wait for 1 second
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        float durationLeft_us = pulseIn(echoPin, HIGH);
        float distanceLeft_cm = 0.017 * durationLeft_us;

        if (distanceLeft_cm >= minDist_side) {
          // Turn left if the path is clear on the left
          turnLeft();
          delay(750);  // Turn left for 1 second
          stop();
          delay(1000);  // Stop for 1 second
          // Move servo to 90 degrees
          myServo.write(90); // center
          Serial.println("Servo position: 90 degrees");
          delay(1000);  // Wait for 1 second
          moveForward();
          return;  // Exit the loop
        }
      }

      // If both right and left are blocked, move backward
      Serial.println("Obstacles detected in all directions. Moving
backward.");
      moveBackward();
      delay(500);  // Move backward for 1 second
      stop();
      delay(1000);  // Stop for 1 second

      // After moving backward, recheck the right and left
directions in a loop
    }
  } else {
    // If distance is greater than or equal to the threshold, turn
off the LED
    digitalWrite(ledPin, LOW);
    moveForward();
  }
}

// Function to move forward
void moveForward() {
  Serial.println("Moving Forward");
  digitalWrite(MOTOR_IN1, HIGH);
  digitalWrite(MOTOR_IN2, LOW);
  digitalWrite(MOTOR_IN3, LOW);
  digitalWrite(MOTOR_IN4, HIGH);
}

// Function to stop
void stop() {
  Serial.println("Stopping");
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, LOW);
```

```
  digitalWrite(MOTOR_IN3, LOW);
  digitalWrite(MOTOR_IN4, LOW);
}

// Function to turn right
void turnRight() {
  Serial.println("Turning Right");
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, HIGH);
  digitalWrite(MOTOR_IN3, LOW);
  digitalWrite(MOTOR_IN4, HIGH);
}

// Function to turn left
void turnLeft() {
  Serial.println("Turning Left");
  digitalWrite(MOTOR_IN1, HIGH);
  digitalWrite(MOTOR_IN2, LOW);
  digitalWrite(MOTOR_IN3, HIGH);
  digitalWrite(MOTOR_IN4, LOW);
}

// Function to move backward
void moveBackward() {
  Serial.println("Moving Backward");
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, HIGH);
  digitalWrite(MOTOR_IN3, HIGH);
  digitalWrite(MOTOR_IN4, LOW);
}

// Function to read the encoder state and update the count
void readEncoder() {
  int newEncoderPinAState = digitalRead(ENCODER_A);
  int newEncoderPinBState = digitalRead(ENCODER_B);

  if (newEncoderPinAState != encoderPinAState) {
    if (newEncoderPinAState == encoderPinBState) {
      encoderCountLeft--;
    } else {
      encoderCountLeft++;
    }
    encoderPinAState = newEncoderPinAState;
  }

  if (newEncoderPinBState != encoderPinBState) {
    if (newEncoderPinAState != encoderPinBState) {
      encoderCountRight--;
    } else {
      encoderCountRight++;
    }
    encoderPinBState = newEncoderPinBState;
  }
}

// Function to adjust motor speed based on encoder counts
void adjustMotorSpeed() {
  // Calculate the difference between left and right encoder
counts
  long error = encoderCountLeft - encoderCountRight;

  // Adjust motor speeds proportionally to the error
  int leftMotorSpeed = constrain(255 - Kp * error, 0, 255);
  int rightMotorSpeed = constrain(255 + Kp * error, 0, 255);

  // Set motor speeds
  analogWrite(MOTOR_IN1, leftMotorSpeed);
  analogWrite(MOTOR_IN2, 0);
  analogWrite(MOTOR_IN3, 0);
  analogWrite(MOTOR_IN4, rightMotorSpeed);
}
```