TIRUMALA INSTITUTE OF TECHNOLOGY AND SCIENCES

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-(AI)

2024-2025

“ OBSTACLE DETECTION AND AVOIDANCE ROBOT ”

PRESENTED BY

1. Thanuja Thota (21AR1A4354)
2. Pasupuleti Mithin (21AR1A4343)
3. Somu Suchitra Devi (21AR1A4351)
4. Kondepi Ramu (21AR1A4332)

PROJECT GUIDE

N.Nageswara Rao

M.Tech,(Ph.D)

HEAD OF THE DEPARTMENT

Dr.K.Prasada Rao

M.Tech,Ph.D

OBSTACLE DETECTION & AVOIDANCE ROBOT

CONTENTS

ABSTRACT

The **Obstacle Detection and Avoidance Robot** is designed for autonomous navigation by detecting and avoiding obstacles in real time. It ensures smooth and safe movement without human intervention.

The system uses **ultrasonic sensors** to detect obstacles and measure distances. A **microcontroller** processes this data to determine the best path. When an obstacle is detected, the robot dynamically changes its direction using microcontroller modes.

This project is useful for **automation, service, and mobility applications**. Future improvements can include machine learning for advanced decision-making and obstacle recognition.

INTRODUCTION

Robots play a crucial role in automation, reducing human effort, and improving efficiency in various fields like **healthcare, industry, transportation, and security**. For a robot to function independently, it must navigate safely without collisions. **Obstacle detection and avoidance** is a key feature that enables robots to move efficiently in dynamic environments. This feature ensures **safety, accuracy, and real-time decision-making**, making it essential for applications like **autonomous vehicles, smart surveillance, and industrial automation**. Hence, our project focuses on developing an **Obstacle Detection and Avoidance Robot** to enhance navigation and operational efficiency.

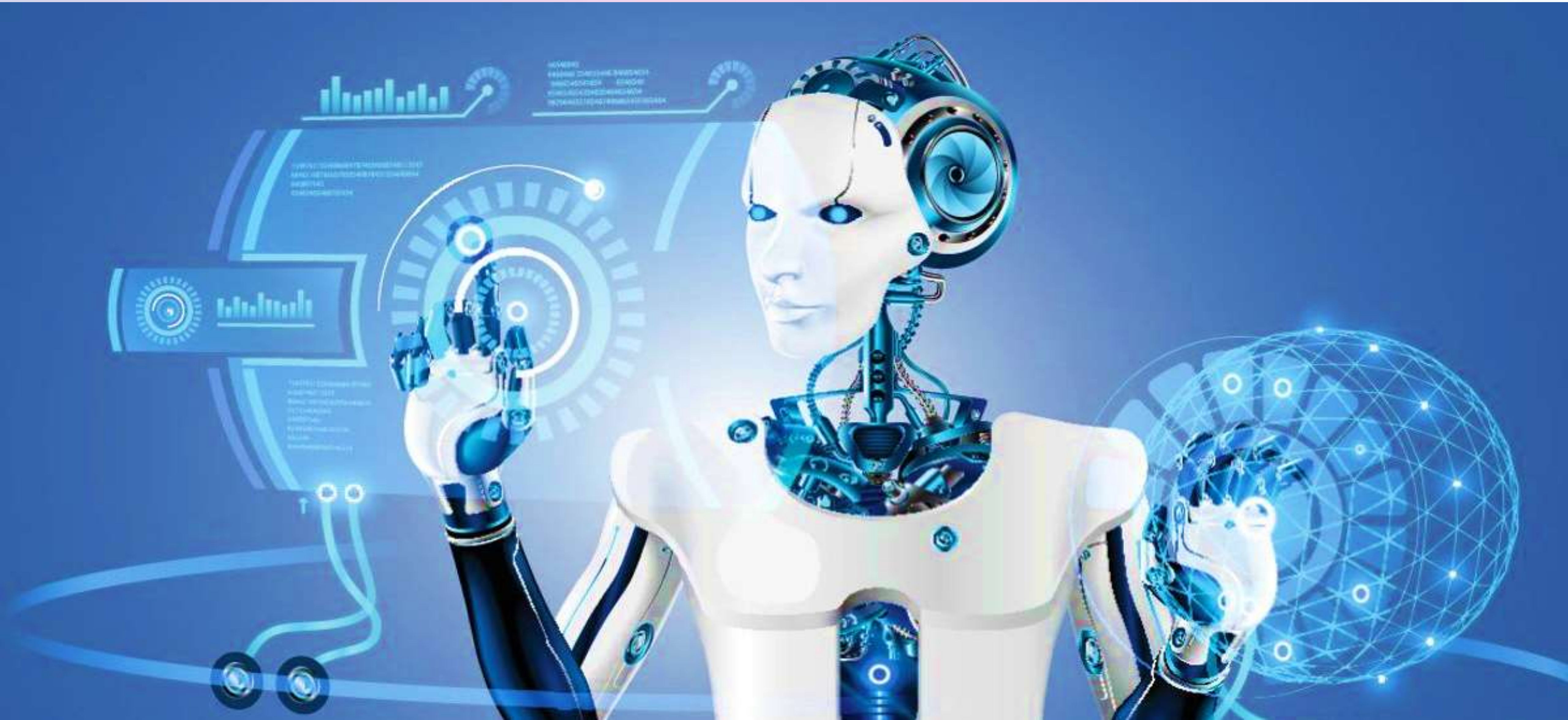
PROBLEM STATEMENT

Many robots today need human control or simple sensors to detect obstacles. These methods do not work well in fast-changing situations. Our goal is to make a smart robot that can sense obstacles and avoid them on its own.

LITERATURE SURVEY

1. "Autonomous Obstacle Detection and Avoidance for Mobile Robots" - IEEE Xplore
2. "Intelligent Path Planning and Obstacle Avoidance for Robotics" - Springer
3. "A Comparative Study of Obstacle Avoidance Algorithms for Autonomous Navigation" - Elsevier
4. "Ultrasonic Sensor-Based Real-Time Obstacle Avoidance System" - ResearchGate
5. "Implementation of an AI-Powered Obstacle Avoidance System in Robotics" - ACM Digital Library

SYSTEM ANALYSIS



EXISTING SYSTEM

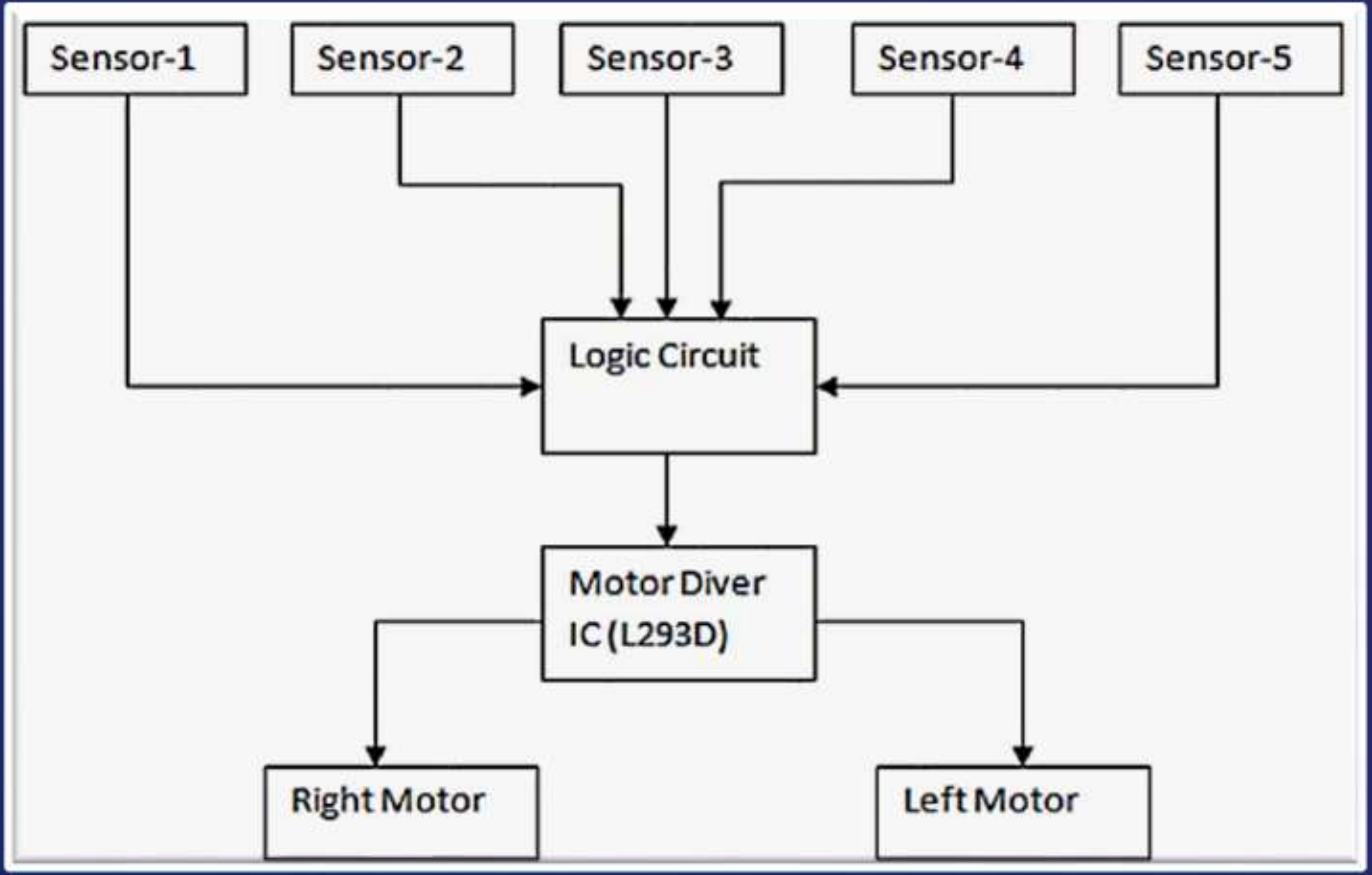


Fig. Block Diagram of Object detection and Avoidance Robot's Existing System

WORKING OF EXISTING SYSTEM

1. The **sensors detect obstacles** and send signals to the logic circuit.
2. The **logic circuit processes the input** and determines the movement strategy:
   - If no obstacle is detected → **Move forward.**
   - If an obstacle is detected in the front → **Stop and decide whether to turn left or right.**
   - If obstacles are detected on both sides → **Stop and move backward or wait.**
3. The **motor driver IC (L293D) receives signals** from the logic circuit and controls the **left and right motors accordingly.**
4. The robot **moves safely while avoiding obstacles.**

DISADVANTAGES OF THE EXISTING SYSTEM

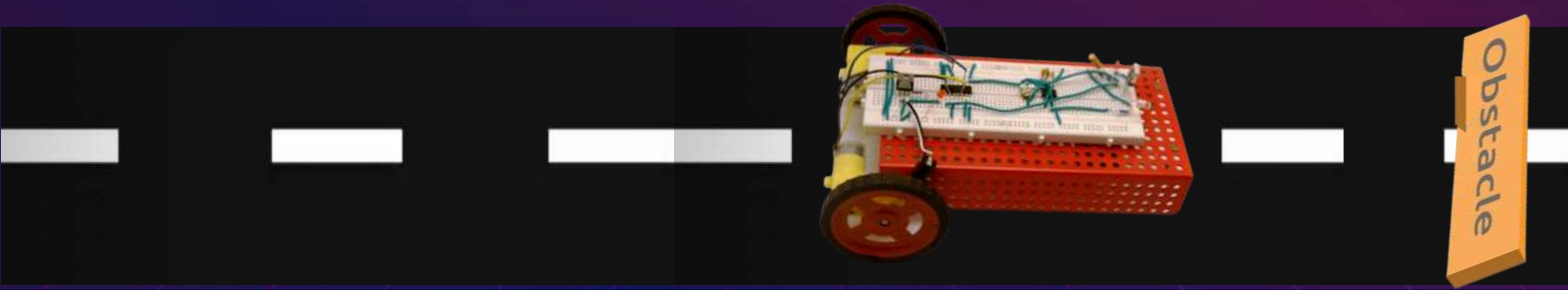
Limited Sensor Range:
Struggles to detect distant obstacles.

Simple Logic:
Can only make basic decisions like turning left or right.

Low Adaptability:
Not effective in changing environments.

No Real-Time Response:
Slow in reacting to sudden obstacles.

High Power Usage:
Sensors consume more energy without optimization.



PROPOSED SYSTEM

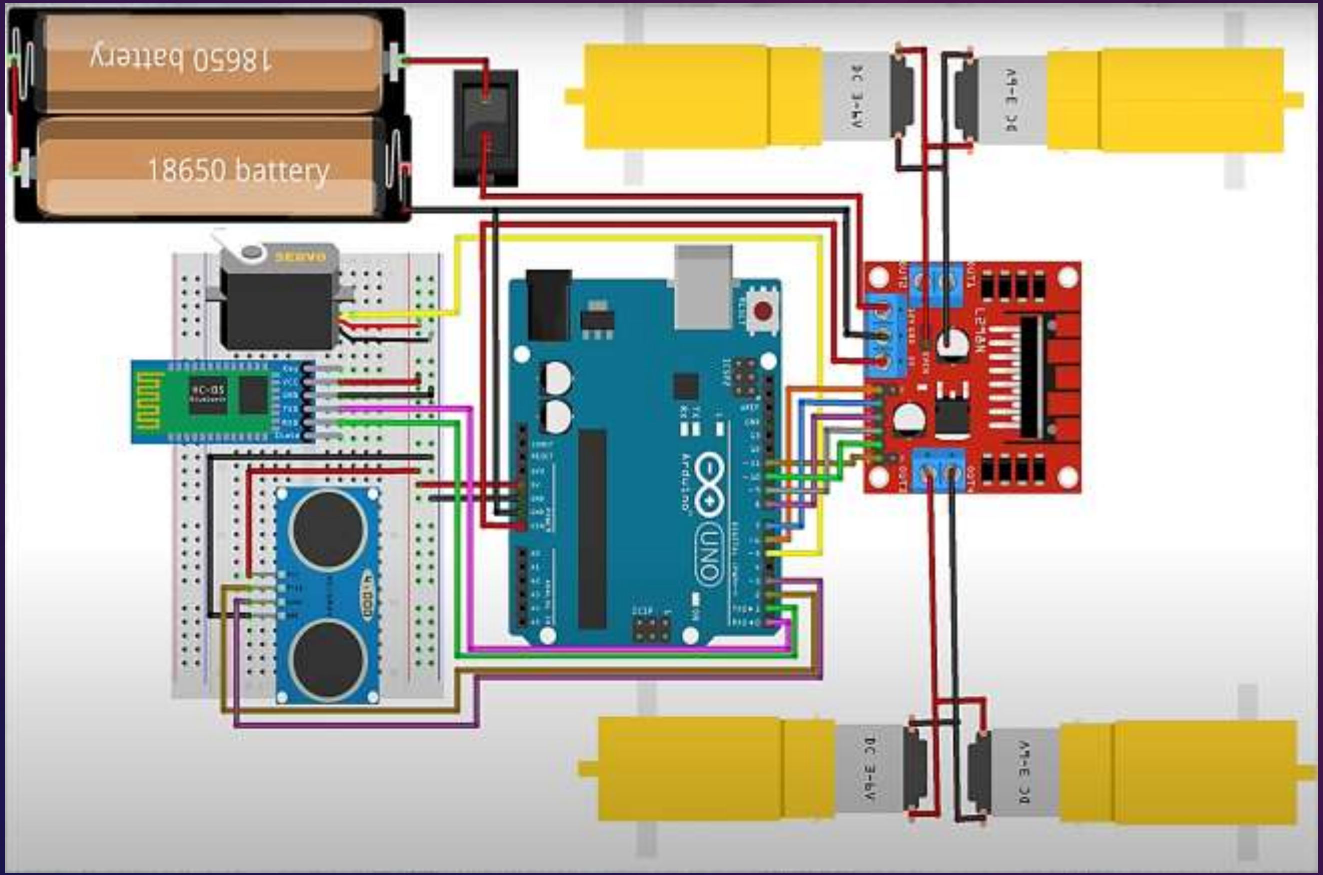


Fig. Block Diagram of Object detection and Avoidance Robot's Proposed System

WORKING OF PROPOSED SYSTEM

1. **Power On** – The robot is powered by 7.4V battery and controlled by Arduino UNO.
2. **Obstacle Detection** – The Ultrasonic Sensor continuously checks for obstacles ahead.
3. **Decision Making**

   If no obstacle, the robot moves forward .

   If obstacle detected, the robot stops and chooses a direction:

   - Turn Left (if the left side is clear)
   - Turn Right (if the right side is clear)
   - Move Backward (if both sides are blocked)

4. **Motor Control** – The Motor Driver Module controls the 4 wheels based on the Arduino's decision.
5. **Additional Feature** – The Bluetooth module allows manual control using a mobile app.
6. **Result** – The robot autonomously avoids obstacles and finds a clear path!

ADVANTAGES OF THE PROPOSED SYSTEM

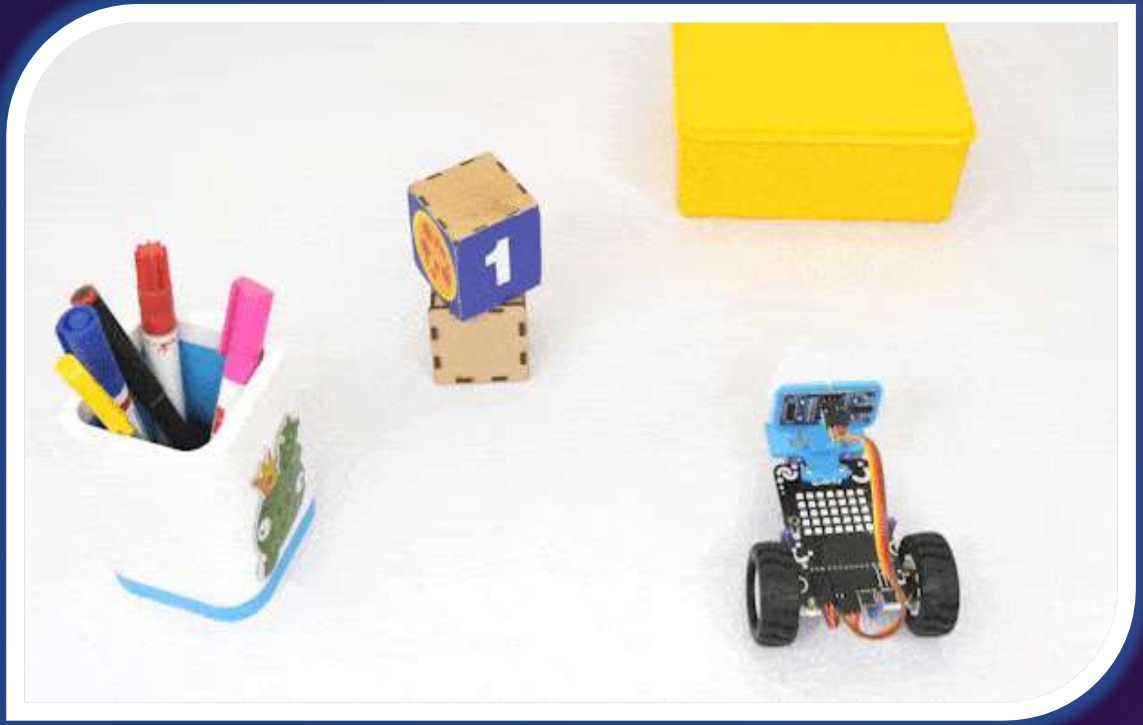


Fig. Real-time working of Object detection and Avoidance Robot

Autonomous Navigation – Moves independently without human control.

Accurate Obstacle Detection – Uses an **ultrasonic sensor** to detect obstacles in real time.

Collision Prevention – Prevents accidents by avoiding obstacles.

Efficient Path Planning – Adjusts speed and direction to find a clear path.

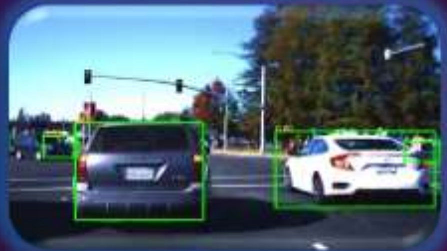
Low Cost & Easy to Build – Uses **Arduino, sensors, and motors**, making it affordable.

APPLICATIONS

Smart Home Automation

Industrial Use

Autonomous Vehicles



Agriculture & Farming

Service Robots

Delivery Robots

SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

Arduino IDE 2.3.4

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the Arduino IDE 2.0 documentation.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on GitHub.

Arduino IDE
For writing and uploading code to the Arduino board.

Embedded C/C++
Programming language for Arduino.

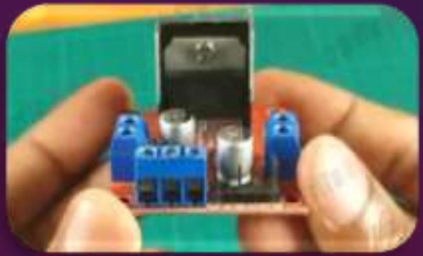
Bluetooth Control App
For remote operation via mobile.

HARDWARE REQUIREMENTS

Arduino UNO

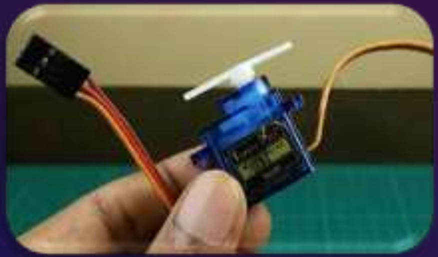
DC Motors (x4)

Battery Holder


Motor Driver Module

Robot wheels

Ultrasonic Sensor


Servo Motor

Jumper Wires

Battery (18650, 3.7V x2)

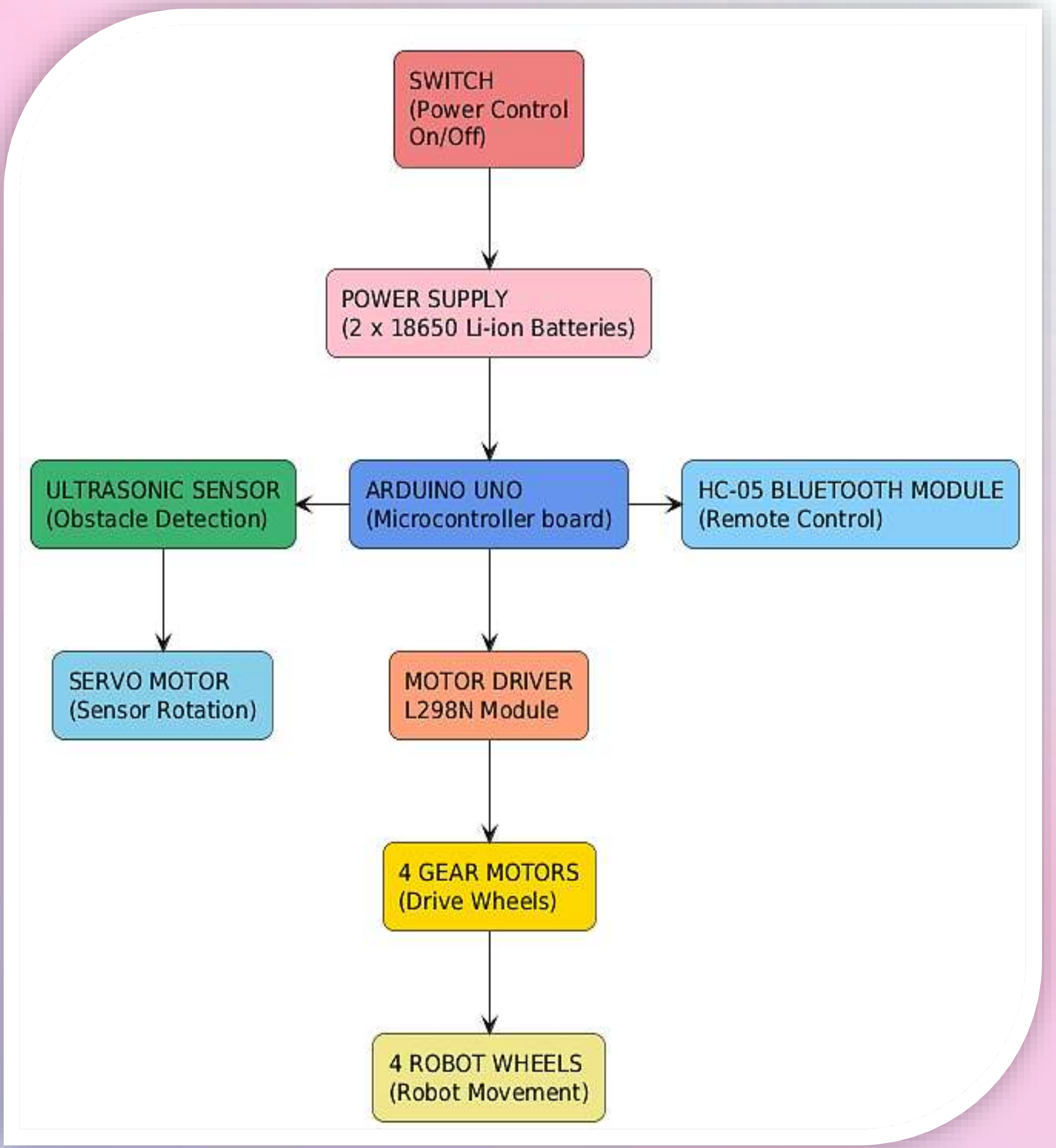
Bluetooth module

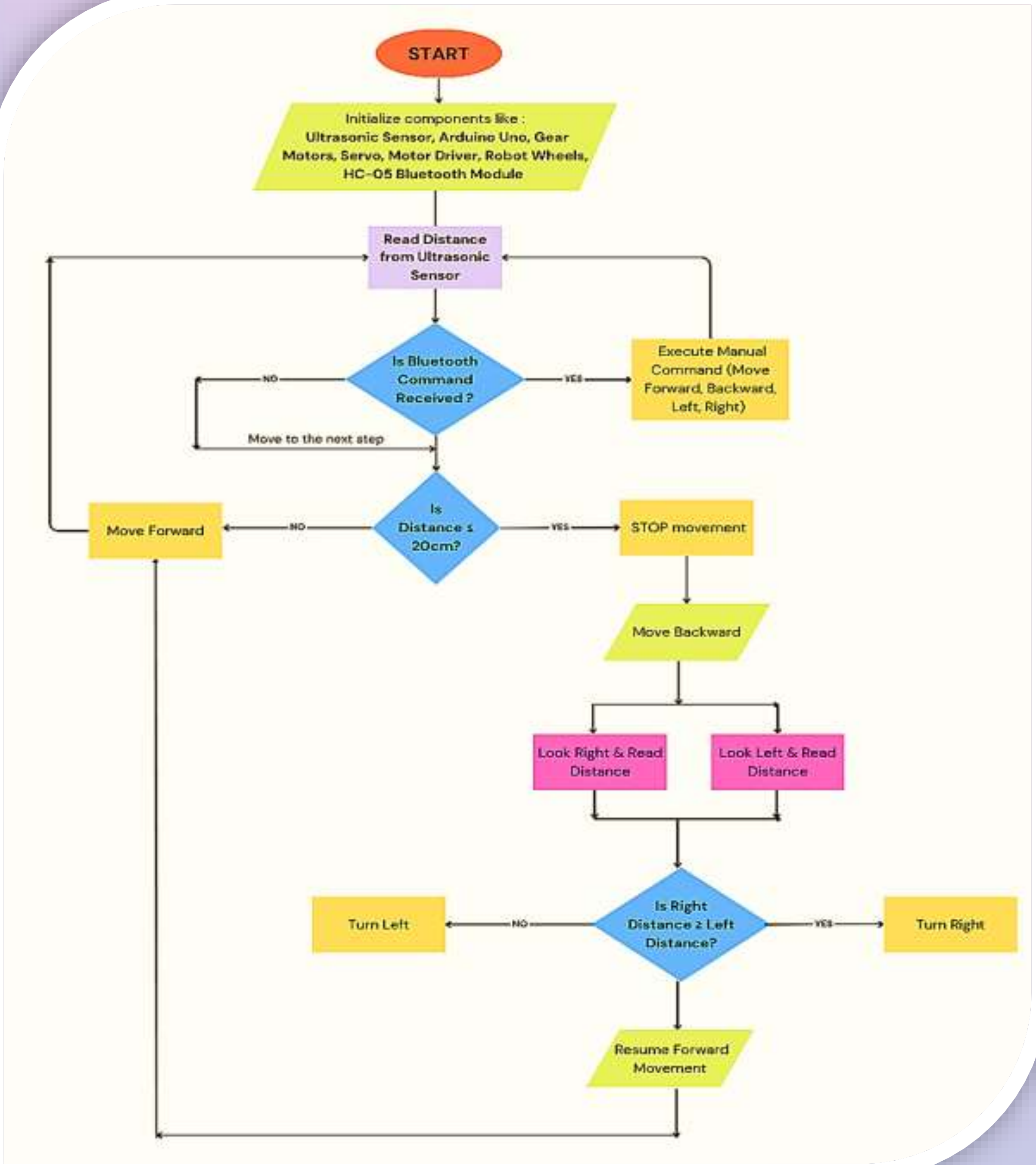
SYSTEM DESIGN

- System Architecture
- Data Flow Diagram
- UML Diagrams

SYSTEM ARCHITECTURE

1. The robot uses an **Arduino Uno** as the main controller.
2. A **switch** and **Li-ion batteries** power the system.
3. An **ultrasonic sensor** detects obstacles, and a **servo motor** rotates the sensor.
4. A **motor driver (L293N)** controls the movement of **4 gear motors** connected to wheels.
5. A **Bluetooth module (HC-05)** allows remote control.
6. **Decision logic** helps the robot move, stop, or turn based on obstacles.


SWITCH (Power Control On/Off)
POWER SUPPLY (2 x 18650 Li-ion Batteries)
ULTRASONIC SENSOR (Obstacle Detection)
ARDUINO UNO (Microcontroller board)
HC-05 BLUETOOTH MODULE (Remote Control)
SERVO MOTOR (Sensor Rotation)
MOTOR DRIVER L298N Module
4 GEAR MOTORS (Drive Wheels)
4 ROBOT WHEELS (Robot Movement)


START
Initialize components like : Ultrasonic Sensor, Arduino Uno, Gear Motors, Servo, Motor Driver, Robot Wheels, HC-05 Bluetooth Module
Read Distance from Ultrasonic Sensor
Is Bluetooth Command Received ?
NO
YES
Execute Manual Command (Move Forward, Backward, Left, Right)
Move to the next step
Is Distance ≤ 20cm?
NO
YES
Move Forward
STOP movement
Move Backward
Look Right & Read Distance
Look Left & Read Distance
Is Right Distance ≥ Left Distance?
NO
YES
Turn Left
Turn Right
Resume Forward Movement

DATA FLOW DIAGRAM

Initialization: The system starts by setting up all components.

Distance Measurement: The ultrasonic sensor detects obstacles.

Bluetooth Control :

If a command is received, the robot moves accordingly. Otherwise, it operates autonomously.

Obstacle Detection :

If **distance > 20 cm**, the robot moves forward.

If **distance ≤ 20 cm**, it stops and moves backward.

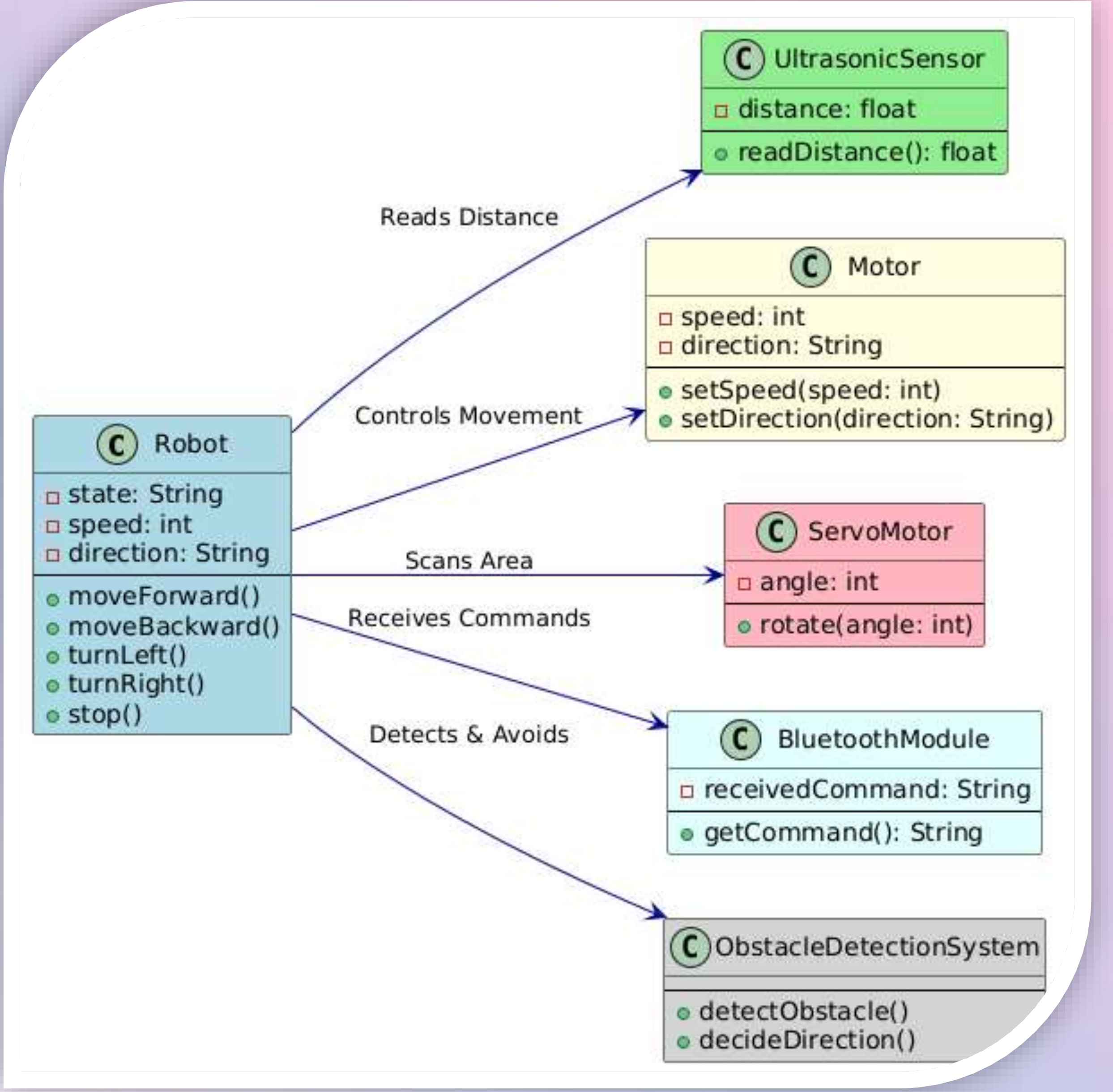
Path Selection:

The servo scans left and right.

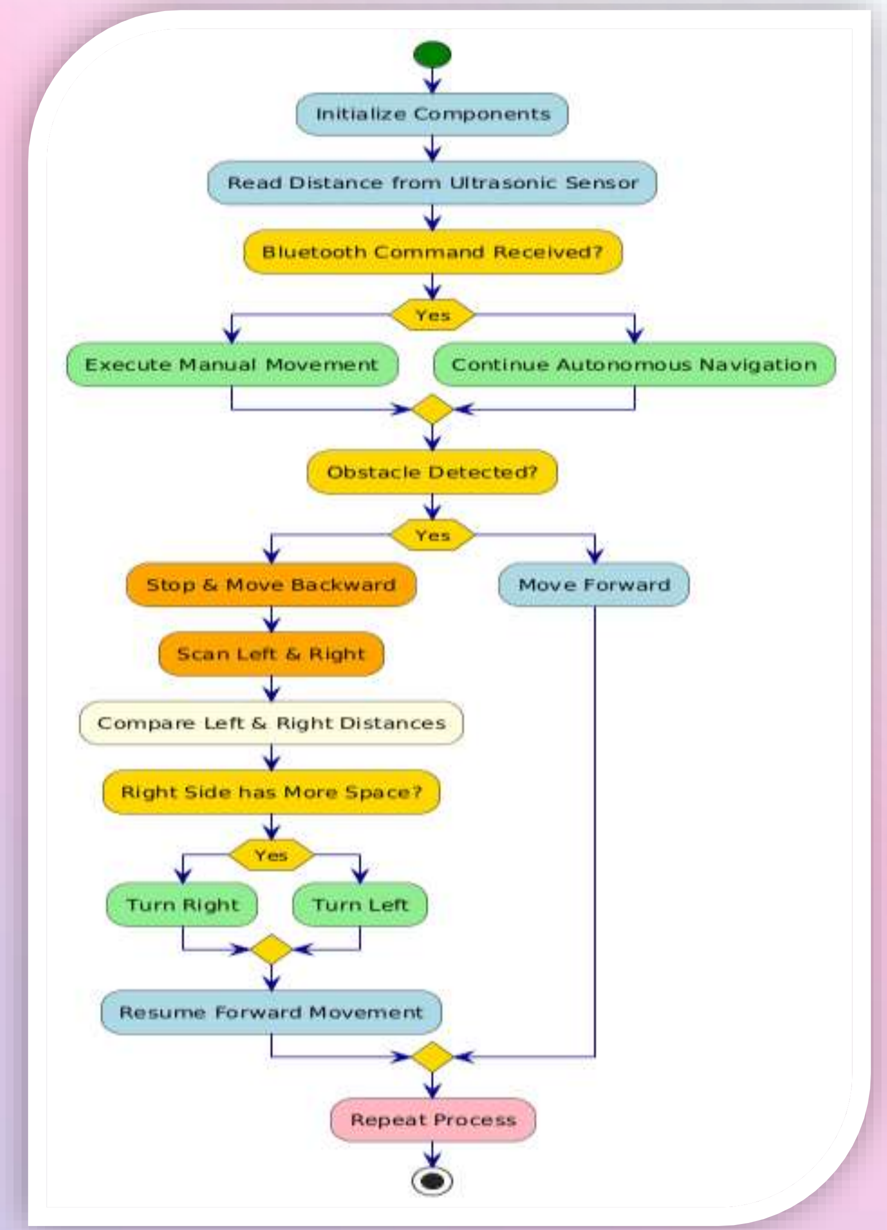
The robot turns toward the side with more space.

Resuming Movement: The robot continues moving forward and repeats the process.

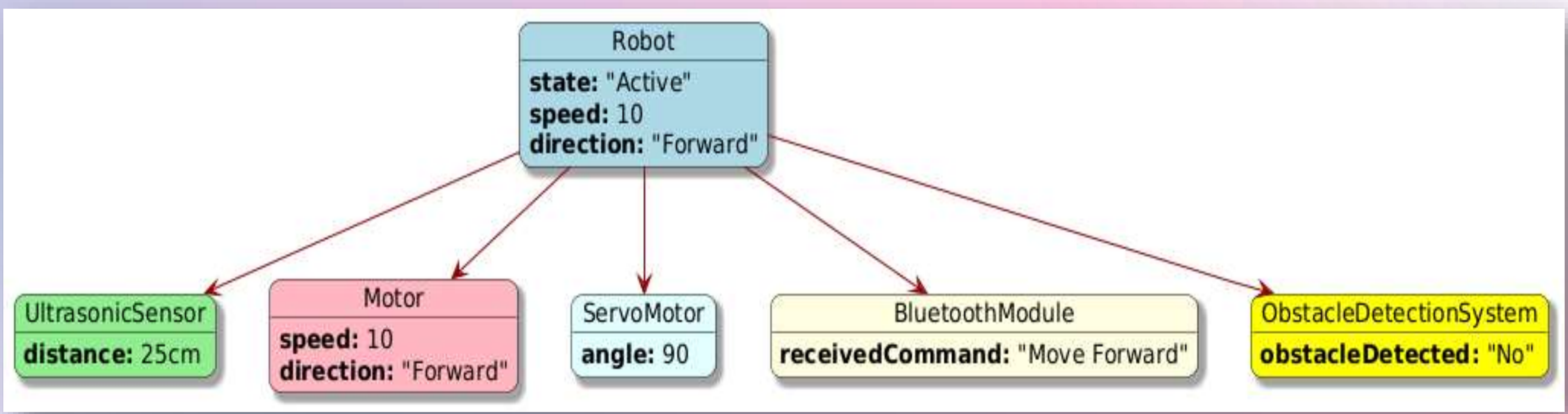
UML CLASS DIAGRAM



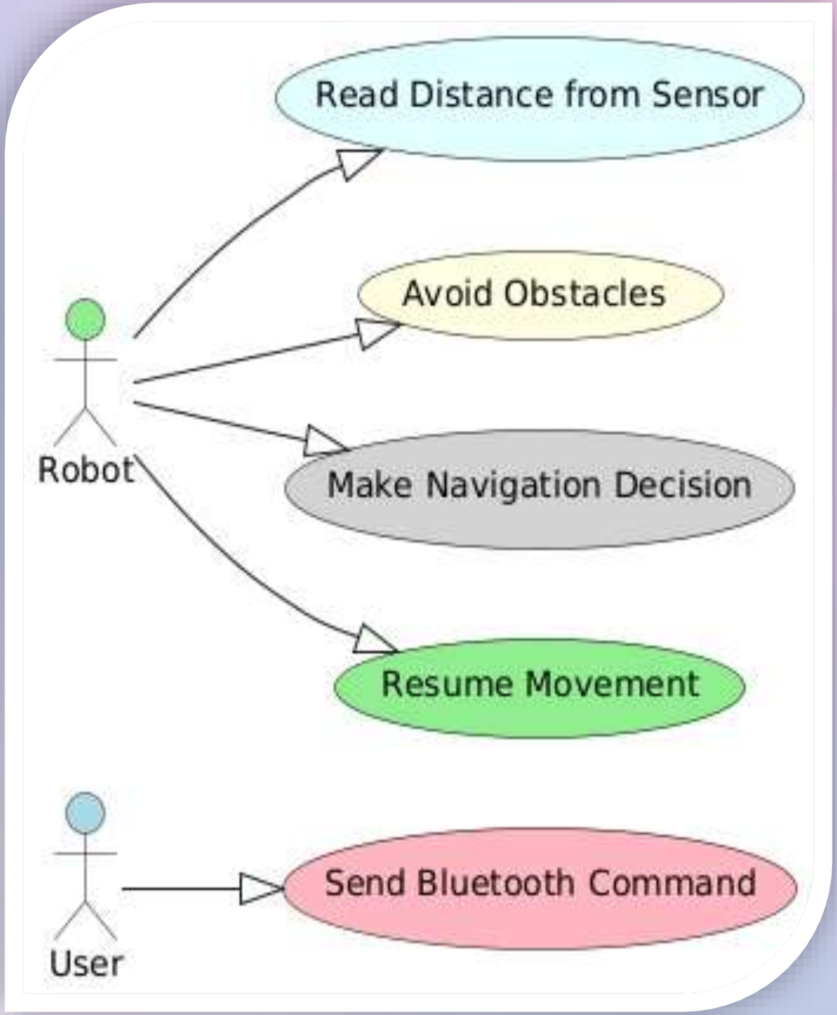
UML ACTIVITY DIAGRAM



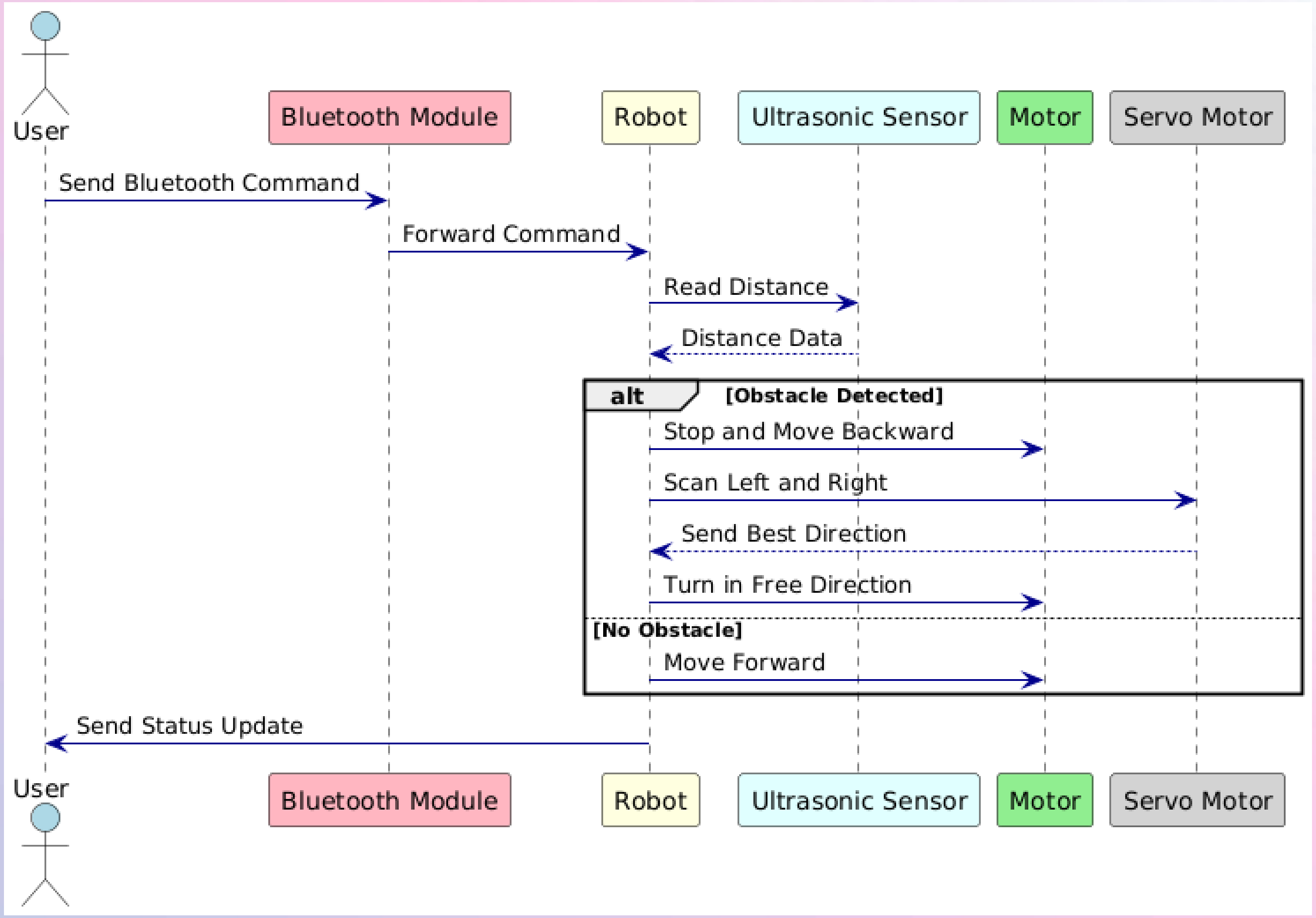
UML OBJECT DIAGRAM



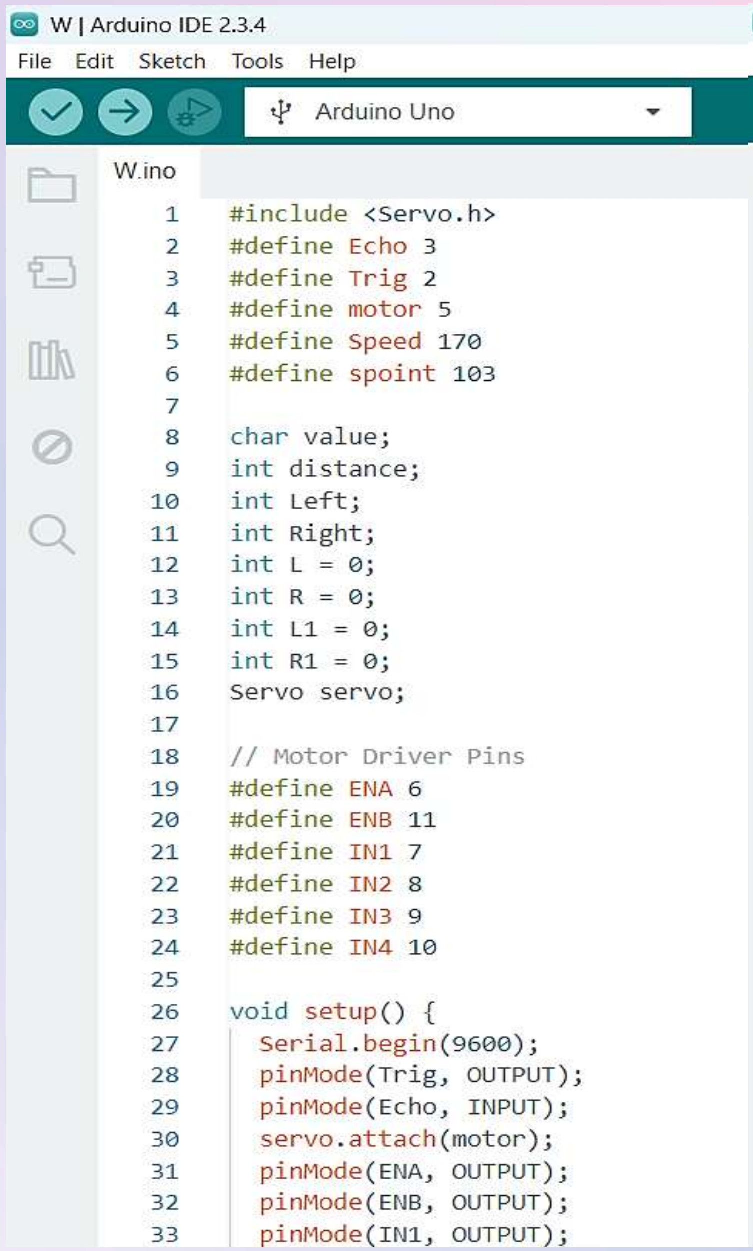
UML USECASE DIAGRAM



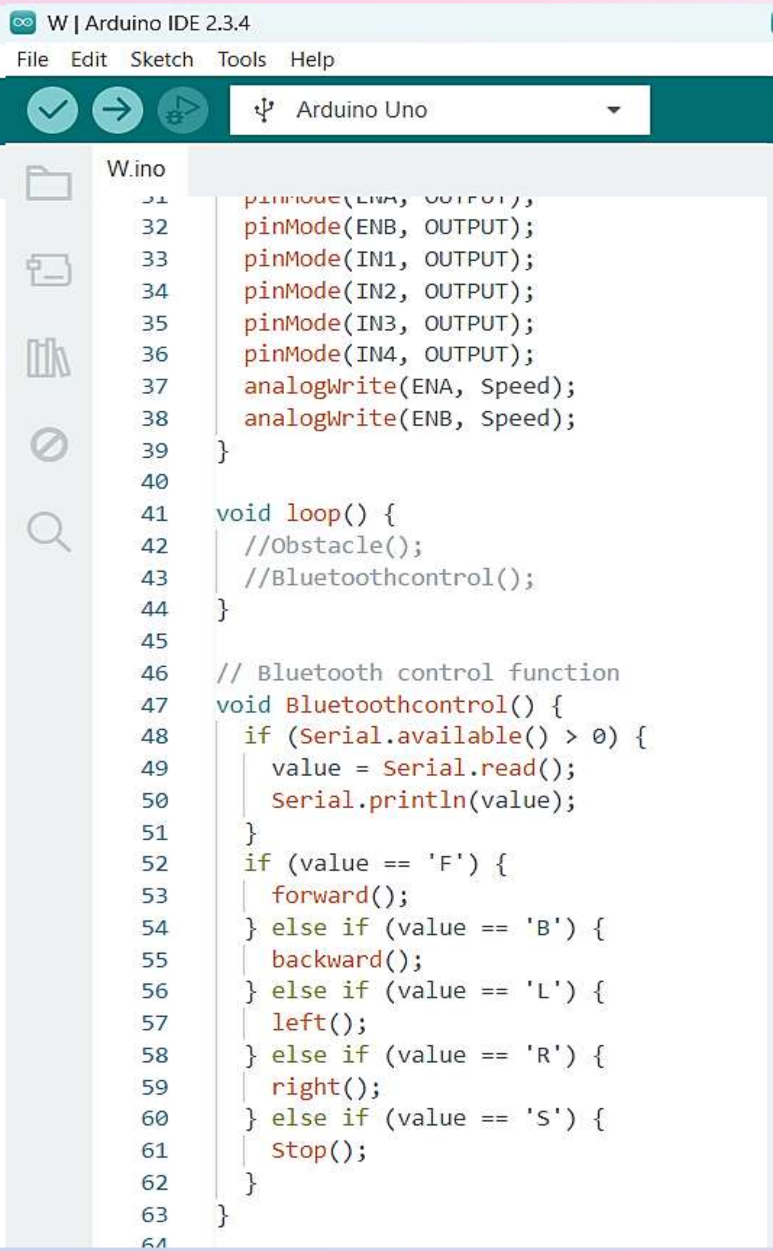
UML SEQUENCE DIAGRAM



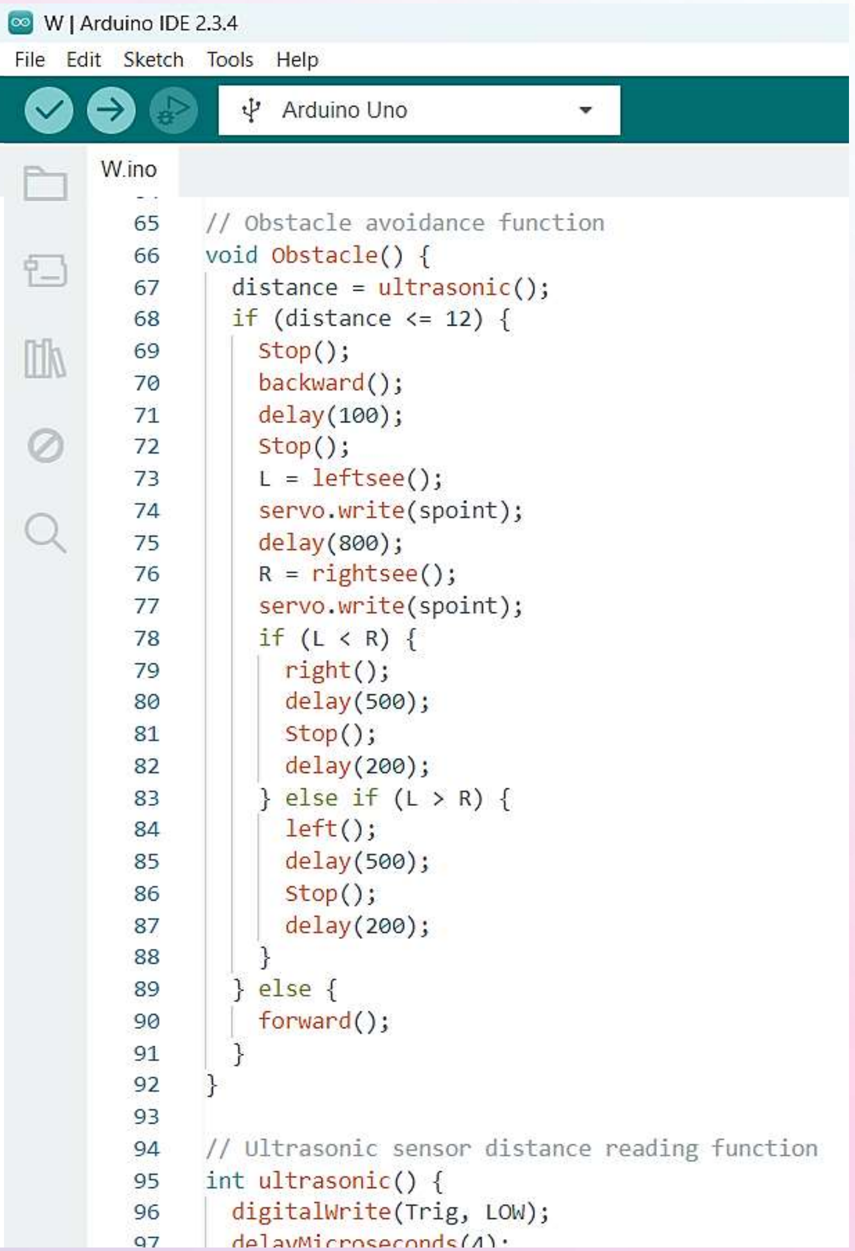
SAMPLE CODE



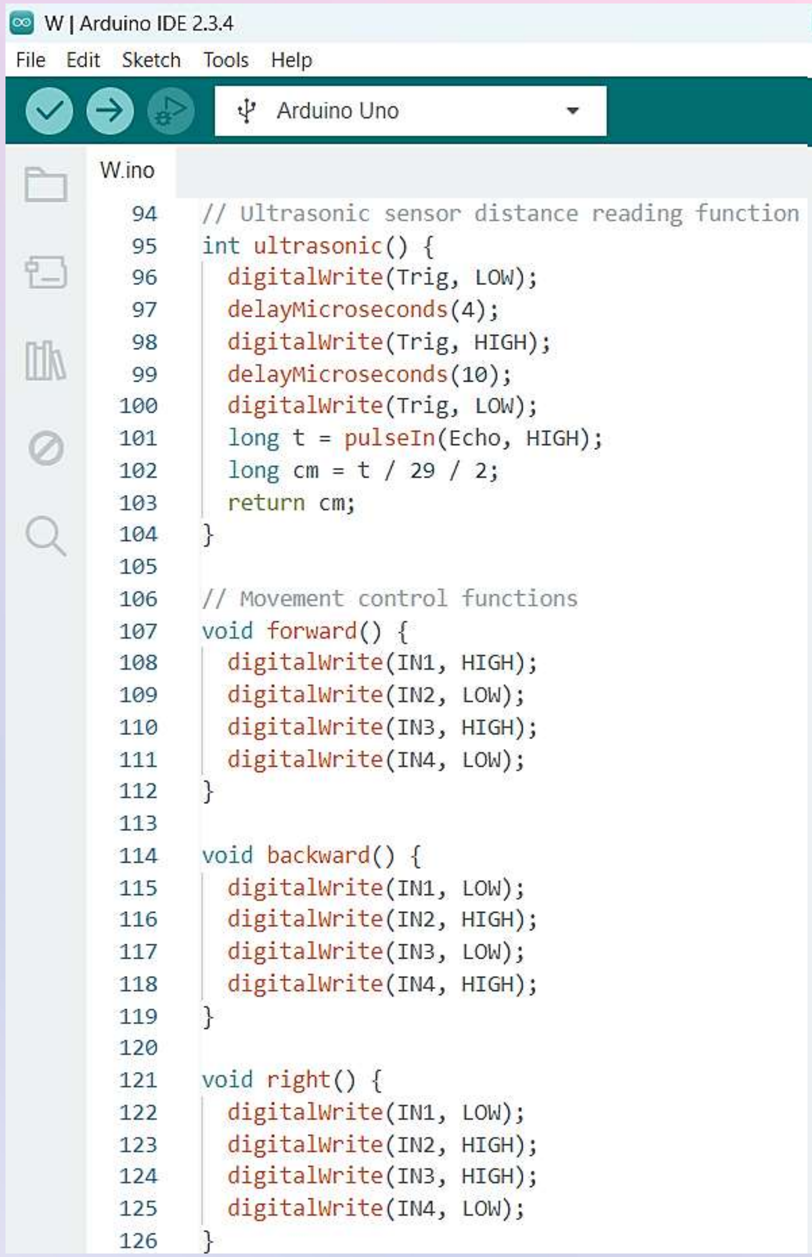
```
#include <Servo.h>
#define Echo 3
#define Trig 2
#define motor 5
#define Speed 170
#define spoint 103

char value;
int distance;
int Left;
int Right;
int L = 0;
int R = 0;
int L1 = 0;
int R1 = 0;
Servo servo;

// Motor Driver Pins
#define ENA 6
#define ENB 11
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 10

void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  servo.attach(motor);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
```



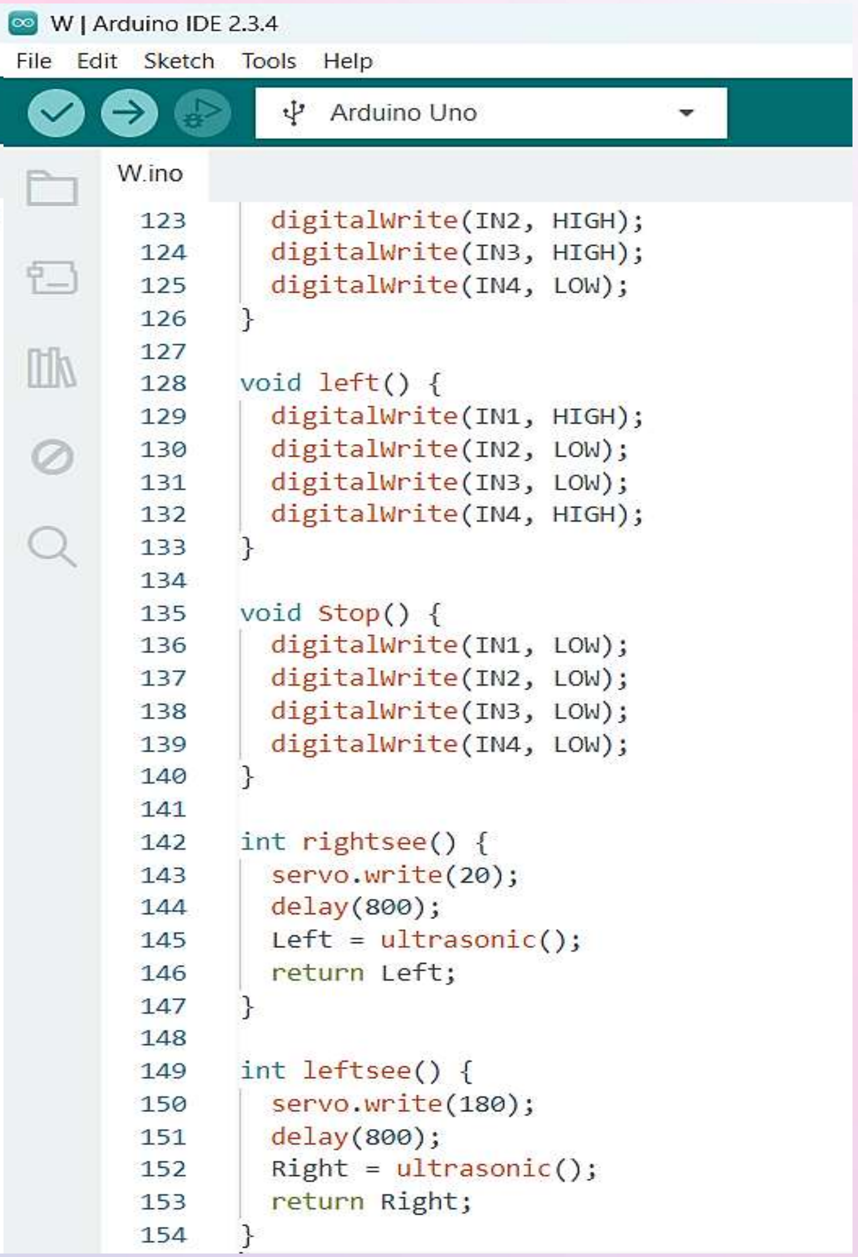
```
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  analogWrite(ENA, Speed);
  analogWrite(ENB, Speed);
}

void loop() {
  //Obstacle();
  //Bluetoothcontrol();
}

// Bluetooth control function
void Bluetoothcontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);
  }
  if (value == 'F') {
    forward();
  } else if (value == 'B') {
    backward();
  } else if (value == 'L') {
    left();
  } else if (value == 'R') {
    right();
  } else if (value == 'S') {
    Stop();
  }
}
```



```
// Obstacle avoidance function
void Obstacle() {
  distance = ultrasonic();
  if (distance <= 12) {
    Stop();
    backward();
    delay(100);
    Stop();
    L = leftsee();
    servo.write(spoint);
    delay(800);
    R = rightsee();
    servo.write(spoint);
    if (L < R) {
      right();
      delay(500);
      Stop();
      delay(200);
    } else if (L > R) {
      left();
      delay(500);
      Stop();
      delay(200);
    }
  } else {
    forward();
  }
}

// Ultrasonic sensor distance reading function
int ultrasonic() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(4);
```



```
// Ultrasonic sensor distance reading function
int ultrasonic() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(4);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
  long t = pulseIn(Echo, HIGH);
  long cm = t / 29 / 2;
  return cm;
}

// Movement control functions
void forward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void backward() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}

void right() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
```



```
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void left() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}

void Stop() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}

int rightsee() {
  servo.write(20);
  delay(800);
  Left = ultrasonic();
  return Left;
}

int leftsee() {
  servo.write(180);
  delay(800);
  Right = ultrasonic();
  return Right;
}
```

MAIN FUNCTIONS

setup( )

1. Runs **once when Arduino starts.**
2. Sets pin modes for:
   - Ultrasonic sensor (Trig, Echo)
   - Motor driver control pins
   - Servo motor
3. Starts Serial communication.
4. Sets initial motor speed.

loop( )

1. Continuously runs.
2. Calls control functions like:
   - Bluetoothcontrol()
   - (Optionally: Obstacle() or
   - voicecontrol())

Bluetoothcontrol( )

1. Reads data from the Bluetooth module.
2. Executes a movement based on the received command:
    - F → Forward
    - B → Backward
    - L → Left
    - R → Right
    - S → Stop

Obstacle( )

1. Uses ultrasonic sensor to measure distance.
2. If obstacle is within **12 cm**:
    - Stops
    - Moves backward
    - Looks left & right using servo
    - Moves in the direction with more space
3. Else → moves forward.

ultrasonic( )

- Measures the distance to an obstacle using the ultrasonic sensor.
- Returns distance in **centimeters (cm).**

Movement Functions

- **forward()** → Moves car forward
- **backward()** → Moves car backward
- **left()** → Turns car left
- **right()** → Turns car right
- **Stop()** → Stops all motors

leftsee() & rightsee()

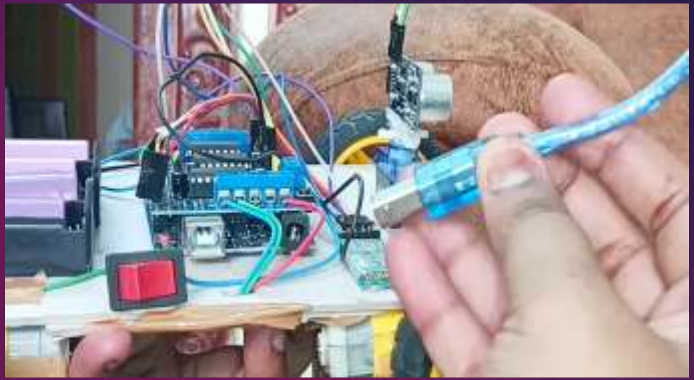
- Moves servo to left/right positions.
- Measures distance in that direction.
- Returns the measured distance.

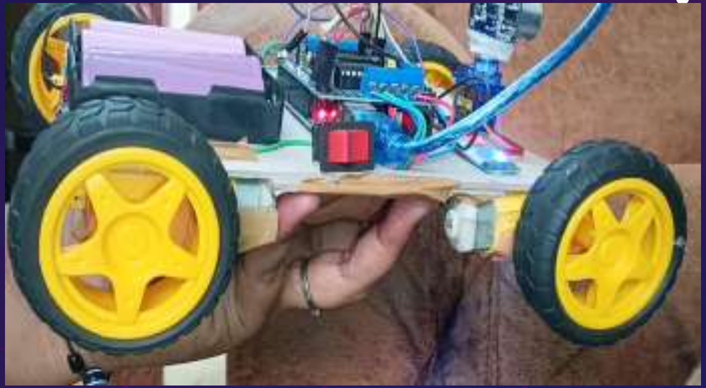
TESTING

Code Execution Output (Serial Monitor)

The serial monitor output displays real-time sensor readings and movement commands. It helps in debugging by showing how the ultrasonic sensor detects obstacles and how the robot responds. Any issues with sensor data or Bluetooth commands can be identified through this output.

Step 1: Connecting the USB Cable

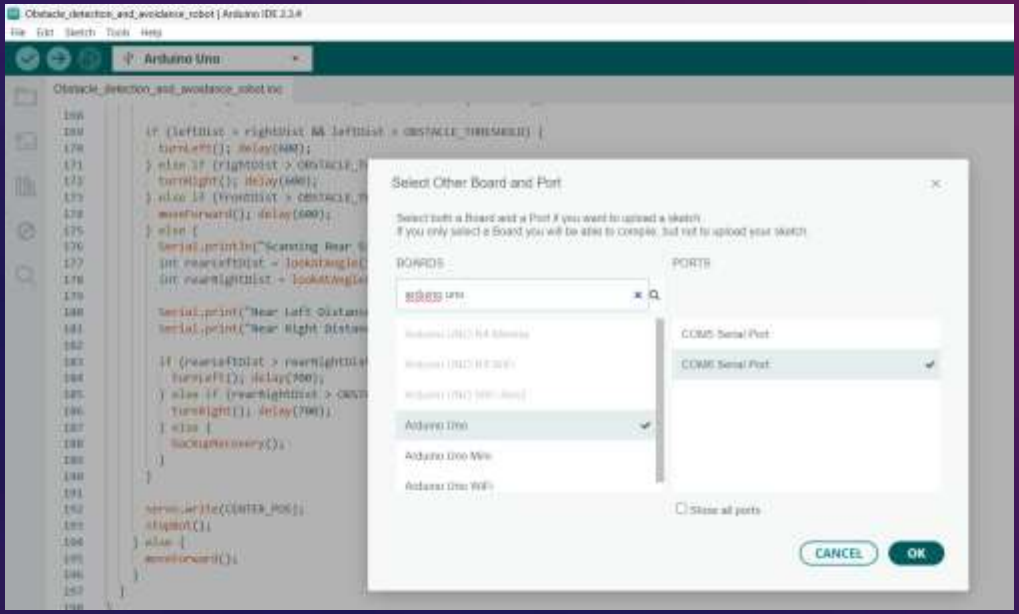




- Take a **USB Type-B cable** and insert one end into your **Arduino Uno** board.

  Connect the other end to your **laptop's USB port** for programming and power supply.

Step 2: Uploading the Code to Arduino


Obstacle_detection_and_avoidance_robot | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Uno
Obstacle_detection_and_avoidance_robot.ino

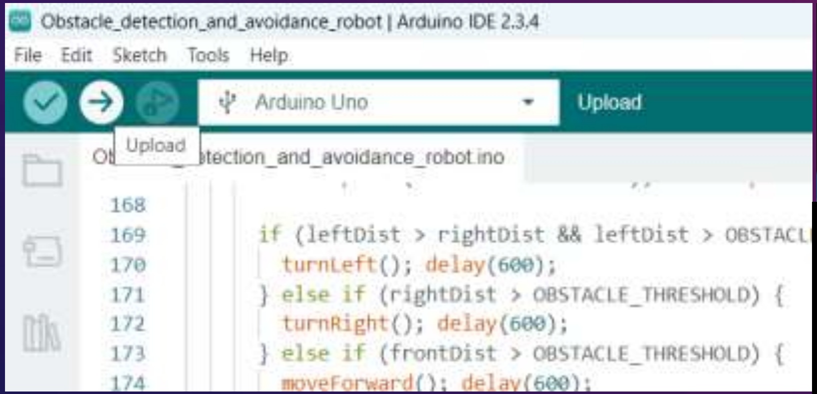
```
        if (leftDist > rightDist && leftDist > OBSTACLE_THRESHOLD) {
          turnLeft(); delay(600);
        } else if (rightDist > OBSTACLE_T
          turnRight(); delay(600);
        } else if (frontDist > OBSTACLE_T
          moveForward(); delay(600);
        } else {
          Serial.println("Scanning Rear S
          int rearLeftDist = lookAtAngle(
          int rearRightDist = lookAtAngle(

          Serial.print("Rear Left Distan
          Serial.print("Rear Right Distan

          if (rearLeftDist > rearRightDis
            turnLeft(); delay(700);
          } else if (rearRightDist > OBST
            turnRight(); delay(700);
          } else {
            backupRecovery();
          }
        }

      servo.write(CENTER_POS);
      stopBot();
    } else {
      moveForward();
    }
  }
}
```

Select Other Board and Port

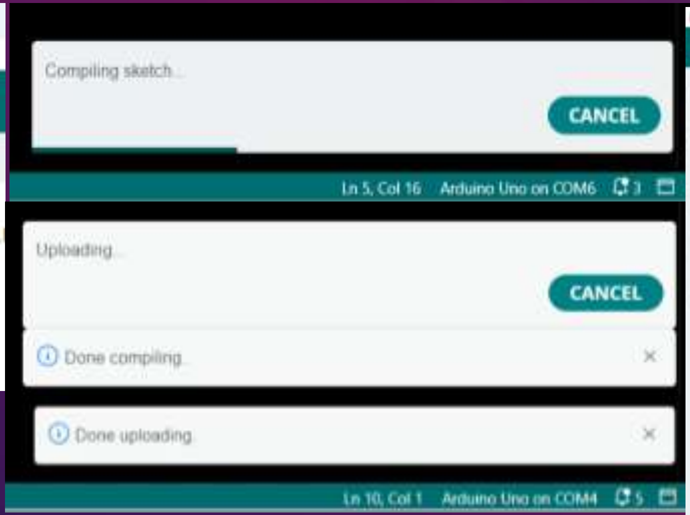
Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.

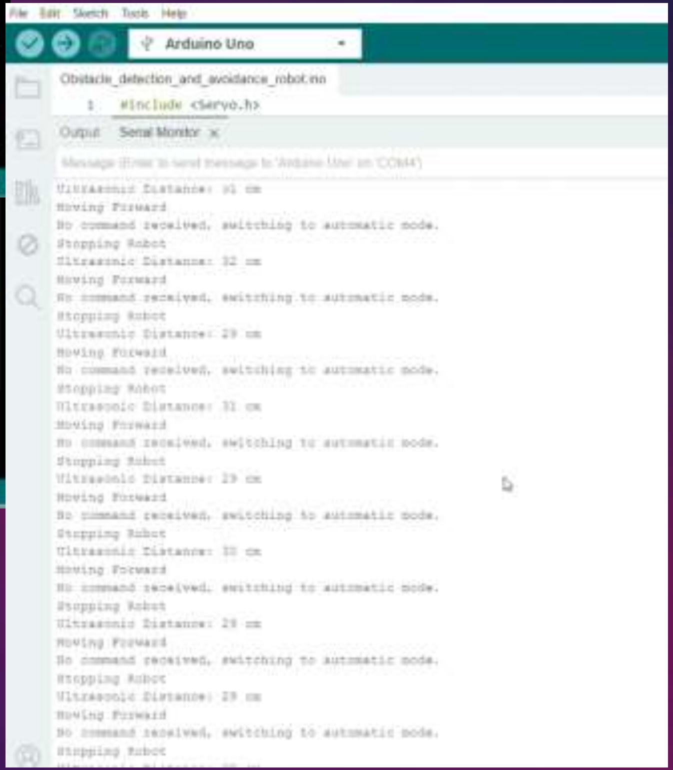
- Open **Arduino IDE** on your laptop.
- Select the correct **Board** (Arduino Uno) and **Port**.

Step 2: Uploading the Code to Arduino


Obstacle_detection_and_avoidance_robot | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Uno
Upload
Upload
Obstacle_detection_and_avoidance_robot.ino

```
    if (leftDist > rightDist && leftDist > OBSTACL
      turnLeft(); delay(600);
    } else if (rightDist > OBSTACLE_THRESHOLD) {
      turnRight(); delay(600);
    } else if (frontDist > OBSTACLE_THRESHOLD) {
      moveForward(); delay(600);
```

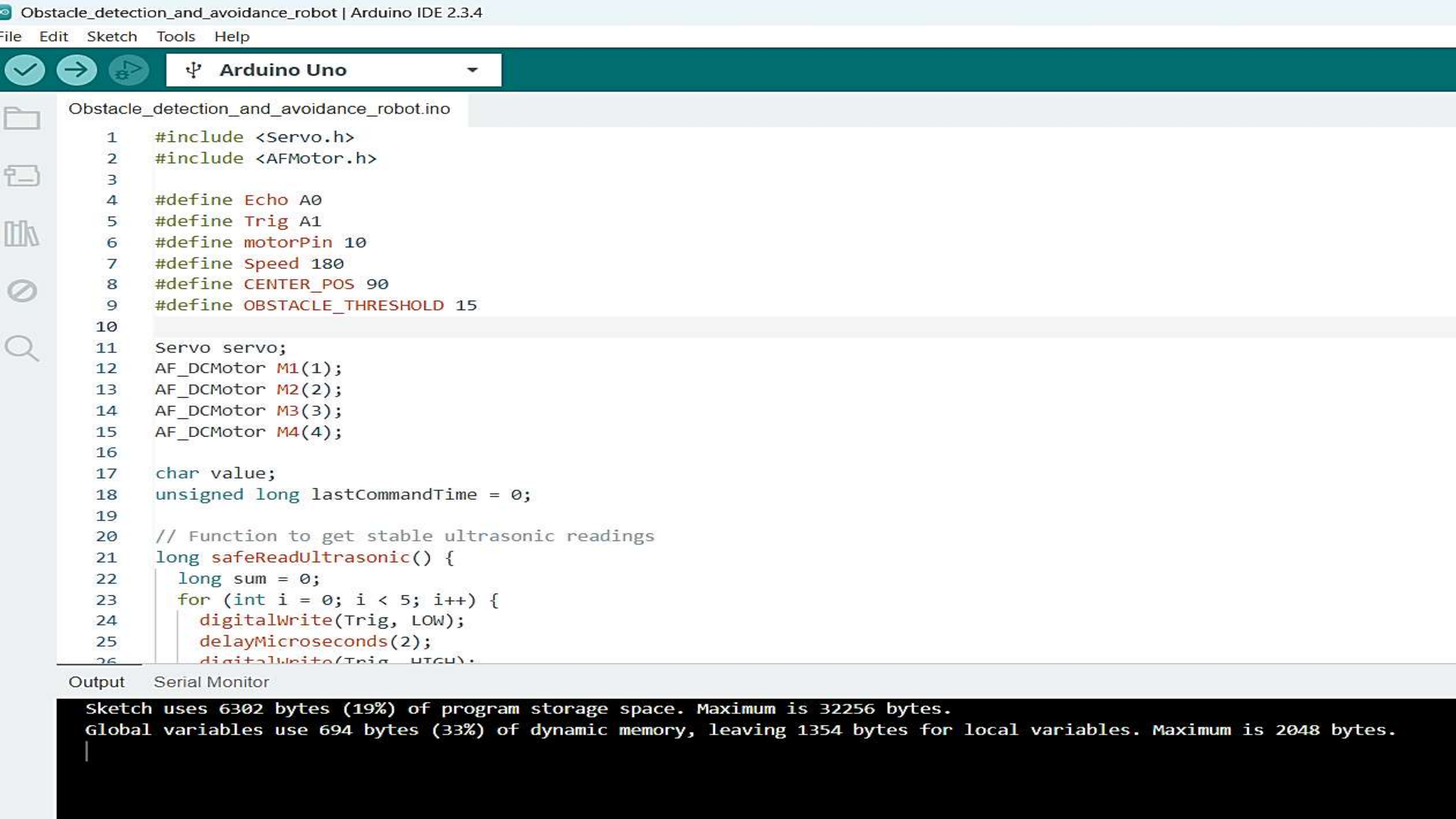



File Edit Sketch Tools Help
Arduino Uno
Obstacle_detection_and_avoidance_robot.ino
Output Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM4')

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

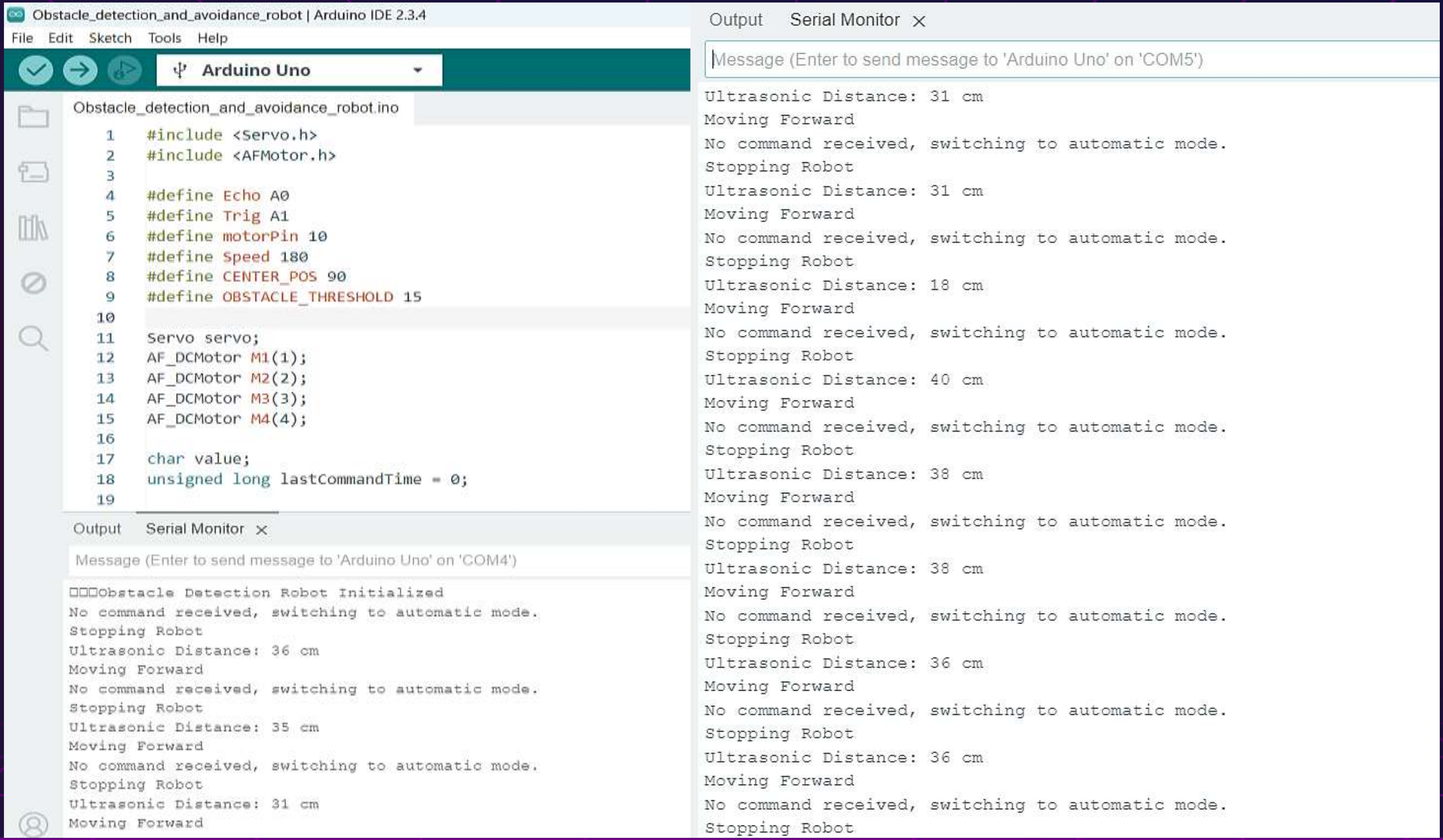
```
Ultrasonic Distance: 31 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 32 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 29 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 31 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 29 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 33 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 29 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 29 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
```

- Write or load the obstacle detection and avoidance **code** into the IDE.
- Click **Upload** to transfer the code to the Arduino.


Obstacle_detection_and_avoidance_robot | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Uno
Obstacle_detection_and_avoidance_robot.ino
Output Serial Monitor

```
#include <Servo.h>
#include <AFMotor.h>

#define Echo A0
#define Trig A1
#define motorPin 10
#define Speed 180
#define CENTER_POS 90
#define OBSTACLE_THRESHOLD 15

Servo servo;
AF_DCMotor M1(1);
AF_DCMotor M2(2);
AF_DCMotor M3(3);
AF_DCMotor M4(4);

char value;
unsigned long lastCommandTime = 0;

// Function to get stable ultrasonic readings
long safeReadUltrasonic() {
  long sum = 0;
  for (int i = 0; i < 5; i++) {
    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
```

```
Sketch uses 6302 bytes (19%) of program storage space. Maximum is 32256 bytes.
Global variables use 694 bytes (33%) of dynamic memory, leaving 1354 bytes for local variables. Maximum is 2048 bytes.
```


Obstacle_detection_and_avoidance_robot | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Uno
Obstacle_detection_and_avoidance_robot.ino
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM4')
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM5')

```
#include <Servo.h>
#include <AFMotor.h>

#define Echo A0
#define Trig A1
#define motorPin 10
#define Speed 180
#define CENTER_POS 90
#define OBSTACLE_THRESHOLD 15

Servo servo;
AF_DCMotor M1(1);
AF_DCMotor M2(2);
AF_DCMotor M3(3);
AF_DCMotor M4(4);

char value;
unsigned long lastCommandTime = 0;

```

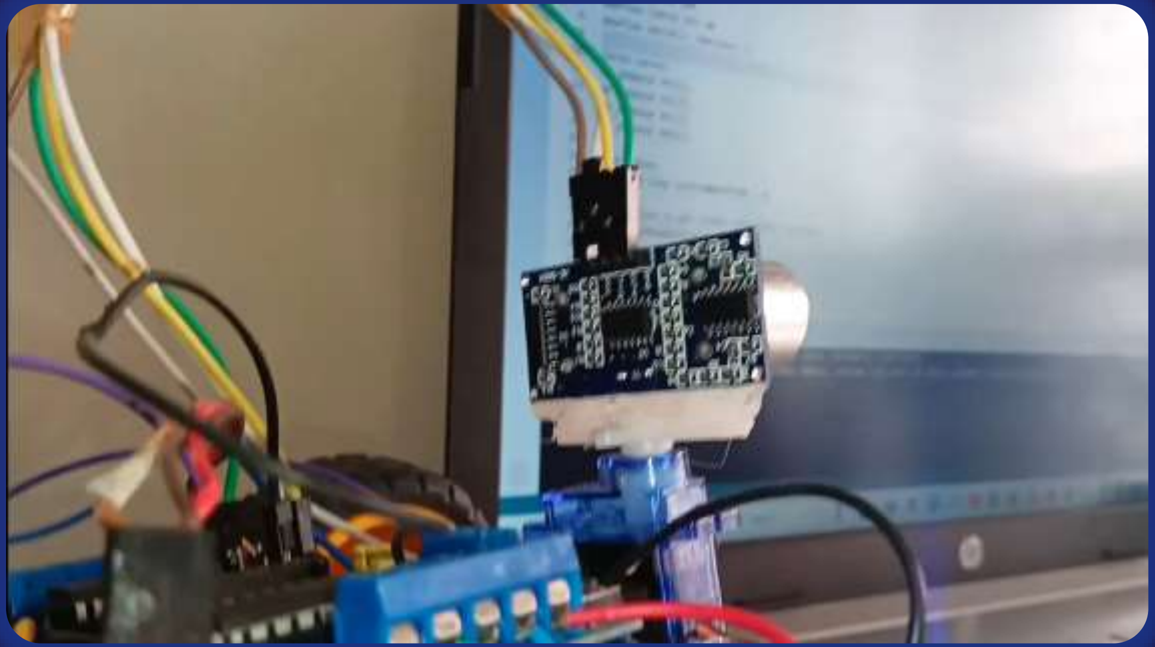
```
□□□Obstacle Detection Robot Initialized
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 36 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 35 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 31 cm
Moving Forward
```

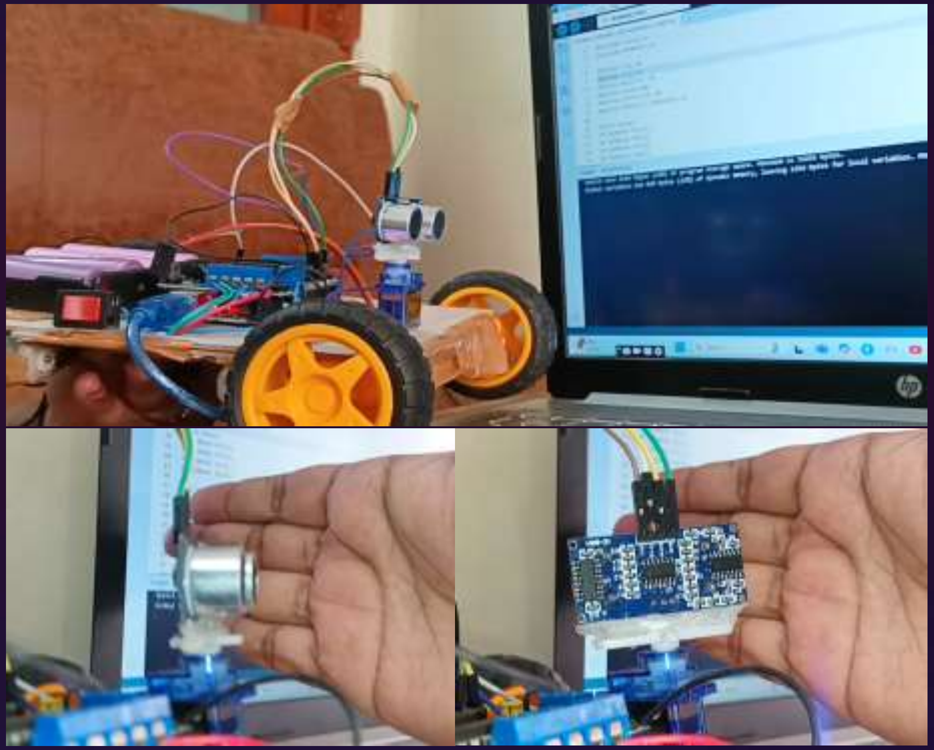
```
Ultrasonic Distance: 31 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 31 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 18 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 40 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 38 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 38 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 36 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
Ultrasonic Distance: 36 cm
Moving Forward
No command received, switching to automatic mode.
Stopping Robot
```

Unit Testing (Component-Level Testing)

Component	Function Name	Purpose
Ultrasonic Sensor	ultrasonic()	Measures distance to obstacles accurately.
Servo Motor	servo.write(angle) in leftsee(), rightsee()	Moves the servo motor to scan left and right for obstacle detection.
Motor Controls	forward(), backward(), left(), right(), Stop()	Moves the robot in respective directions and stops it.
Bluetooth Commands	Bluetoothcontrol()	Receives and executes commands ('F', 'B', 'L', 'R', 'S') via Bluetooth.

Unit Testing (Component-Level Testing)





Integration Testing (Module Interaction)

Components Involved	Functions Involved	Purpose
Ultrasonic Sensor + Servo Motor + Motor Driver	Obstacle(), leftsee(), rightsee(), ultrasonic(), Movement functions	To detect obstacles, decide direction, and navigate accordingly.
Bluetooth Module + Motor Driver	Bluetoothcontrol(), Movement functions	To receive commands and control motor movements.
Servo + Obstacle Avoidance Logic	leftsee(), rightsee() within Obstacle()	To scan left and right and choose safer direction.

System Testing (Complete Robot)

Objective :

Ensure the robot performs as expected when all modules work together in various real-world conditions.

Test Scenarios :

- **Without Obstacles :** Robot moves continuously forward.
- **Single Front Obstacle :** Robot stops, moves backward, scans left & right, and picks a clearer path.
- **Obstacles on Both Sides :** Robot performs backup maneuver by choosing less obstructed side.
- **Bluetooth Manual Control :** Robot responds instantly to F, B, L, R, S commands via serial input.

System Testing (Complete Robot)

Alpha Testing (Internal Team Testing)

Objective : Identify and fix early-stage bugs.

Performed By :

Thanuja Thota - 21AR1A4354

Mithin Pasupuleti - 21AR1A4343

Suchitra devi Somu - 21AR1A4351

Ramu Kondepi - 21AR1A4332

Key Checks :

- Test in **various lighting conditions** (affects ultrasonic sensor).
- Run **continuous operation tests** to check **battery drainage and overheating.**
- Test **manual override (Bluetooth Control) reliability**.

Functional Testing

Objective : Verify that each function works as intended in different scenarios.

Tested Functions :

ultrasonic( ) – Ensures accurate obstacle distance measurement.

leftsee( ) / rightsee( ) – Checks servo motor scanning and distance reading at specific angles.

forward( ) / backward( ) – Validates forward and backward movements.

left( ) / right( ) – Ensures smooth left and right turns.

Stop( ) – Confirms immediate stop functionality.

Obstacle( ) – Tests obstacle detection and automatic path decision.

Bluetoothcontrol( ) – Ensures correct response to Bluetooth commands (F, B, L, R, S).

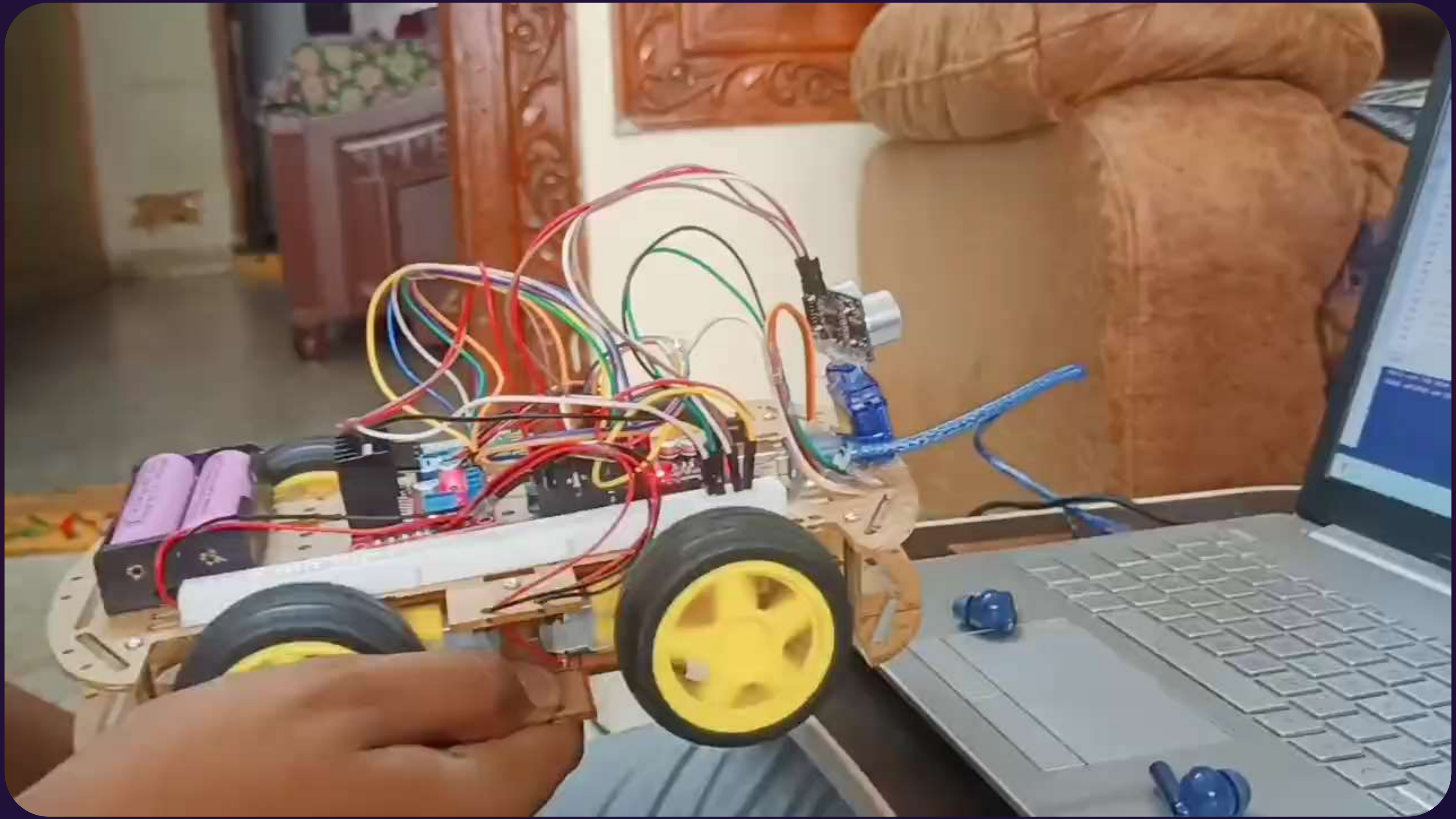
Performance & Stress Testing

Objective : Check how well the robot performs under continuous use and tough conditions.

Tests Performed :

- **Continuous Movement :** Run forward, backward, left, and right for a long time to see if motors heat up or slow down.
- **Fast Obstacle Detection :** Check if the ultrasonic sensor can quickly and correctly measure distances when obstacles appear one after another.
- **Bluetooth Command Spam :** Send many commands rapidly from Bluetooth to see if the robot responds properly without delays or errors.
- **Obstacle Crowd Test:** Place multiple obstacles around and check if the robot avoids them smartly without getting stuck.

Performance & Stress Testing



CONCLUSION

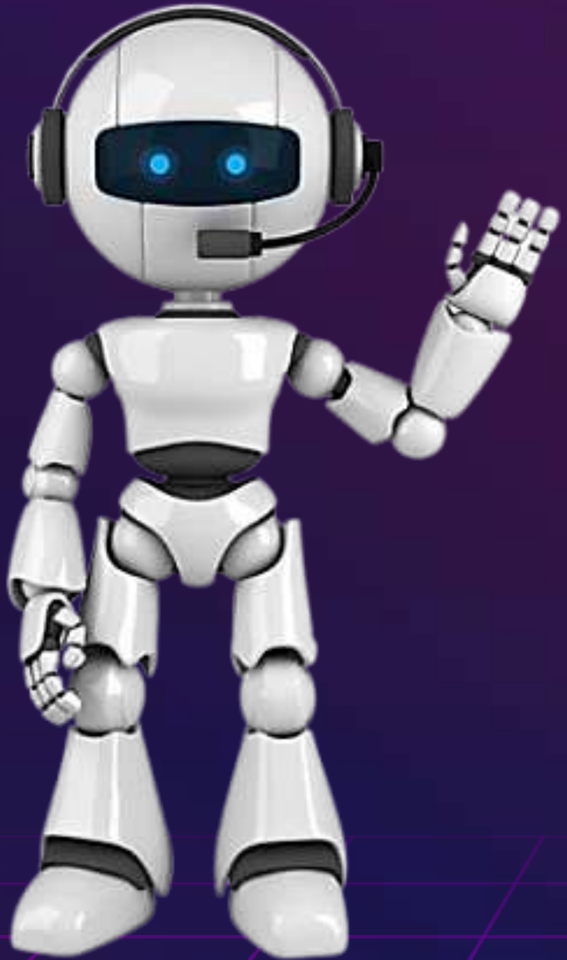
- Developed a **fault-tolerant** robot with **obstacle avoidance + Bluetooth + voice control**.
- Implemented a **smart scanning mechanism** for adaptive movement.
- Ensured **seamless switching** between manual and autonomous modes.
- Optimized **backward movement** based on obstacle distance.
- **Key Takeaway :** The robot successfully navigates its environment **without getting stuck**, making it suitable for **autonomous mobility applications**.

FUTURE ENHANCEMENTS

1. **AI-Based Decision Making** – Implement ML models for **smarter obstacle handling.**
2. **Enhanced Navigation** – Integrate GPS for outdoor navigation.
3. **IoT Connectivity** – Enable remote monitoring & control via the internet.
4. **Battery Optimization** – Implement power-saving techniques for **longer operation.**
5. **Advanced Sensors** – Use **LIDAR or camera vision** for **more accurate object detection.**

Bibliography

- Arduino Official Documentation – www.arduino.cc
- Adafruit Motor Shield Library Guide – https://learn.adafruit.com/
- Servo Motor Control using Arduino – www.arduino.cc/reference
- HC-05 Bluetooth Module Guide – randomnerdtutorials.com



THANK YOU!