OBSTACLE DETECTION AND AVOIDANCE ROBOT

A Project report submitted in the partial fulfilment of the Academic requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By

T. THANUJA THOTA	**21AR1A4354**
P. MITHIN	**21AR1A4343**
S. SUCHITRADEVI	**21AR1A4351**
K. RAMU	**21AR1A4332**

Under the esteemed guidance of

Mr. N. NAGESWARARAO B.E, M. Tech

Assistant Professor

Department of Computer Science & Engineering

TIRUMALA INSTITUTE OF TECHNOLOGY & SCIENCES

(Affiliated to J.N.T. University, KAKINADA, Approved by AICTE, Accredited by NAAC)

Satuluru, Nadendla (MD), Palnadu (DT), Andhra Pradesh 522549

2024 - 2025

TIRUMALA INSTITUTE OF TECHNOLOGY & SCIENCES

(Affiliated to J.N.T. University, KAKINADA, Approved by AICTE, Accredited by NAAC)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled " **OBSTACLE DETECTION AND AVOIDANCE ROBOT** " submitted by T. Thanuja (21AR1A4354), P. Mithin (21AR1A4343), S. Suchitradevi (21AR1A4351), K. Ramu (21AR1A4332), in partial fulfilment for the award of the degree of " **Bachelor of Technology** " in **Computer Science & Engineering** is a record Bonafide work carried out by us under the guidance and supervision during the academic year 2024-2025 and found worthy of acceptance to requirements of the university.

Internal Guide

Mr.N.NAGESWARARAO B.E, M.Tech

Assistant Professor

Head of the Department

Dr.K.PRASADARAO B.Tech, M.Tech, Ph.D

Professor

Project coordinator

Ms.B.JAYAVARDHANI B.Tech , M.Tech

Assistant professor

External Examiner

TIRUMALA INSTITUTE OF TECHNOLOGY & SCIENCES

(Affiliated to J.N.T. University, KAKINADA, Approved by AICTE**,** Accredited by NAAC)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that this dissertation entitled **"OBSTACLE DETECTION AND AVOIDANCE ROBOT"** being submitted to the Department of **COMPUTER SCIENCE AND ENGINEERING,** Accredited by AICTE, affiliated to **JNTU, KAKINADA** for the award of **Bachelor of Technology,** is a record of Bonafide work done by us it has not been submitted to any other Institute or University of the award other degree.

Place:

Date:

SUBMITTED BY

T. THANUJA THOTA	**21AR1A4354**
P. MITHIN	**21AR1A4343**
S. SUCHITRADEVI	**21AR1A4351**
K. RAMU	**21AR1A4332**

ACKNOWLEDGEMENT

We express our deep sense of gratitude to **Dr. NALABOTHU VENKATA RAO B.V.SC.,** Chairman of **TIRUMALA INSTITUTE OF TECHNOLOGY & SCIENCES** Narasaraopet, for creating excellent academic atmosphere and providing good infrastructural facilities to us.

We express our deep-felt gratitude to the management **of TIRUMALA INSTITUTE OF TECHNOLOGY & SCIENCES** for helping us in successful completion of the project.

We are grateful to our Director Prof. **Dr. Y.V. NARAYANA B.Tech,M.Tech,Ph.D.** for providing us all the necessary facilities for the completion of this project in specified time.

We are grateful to our Principal Prof. **Dr. K.PRASADA RAO B.Tech,M.Tech,Ph.D.** for providing us all the necessary facilities for the completion of this project in specified time.

We express to our sincere gratitude to **Dr. K.PRASADA RAO B.Tech,M.Tech,Ph.D.** Professor and Head of the Department of Computer Science & Engineering , for his support and encouragement during the period of project work.

We express our sincere thanks to **Ms. B.JAYAVARDHANI B.Tech,M.Tech.,** Assistant Professor and project coordinator Department of Computer Science & Engineering , for her support and encouragement during the period of project work.

We express our sincere thanks to my guide **Mr. N.NAGESWARARAO B.Tech,M.Tech.,** Assistant Professor for his valuable guidance and constant encouragement which enabled us to accomplish our project successfully in time. His vast experience, profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We express our earnest thanks to other faculty members of department of Computer Science and Engineering for extending their helping hands and valuable suggestions when in need.

T. THANUJA THOTA	**21AR1A4354**
P. MITHIN	**21AR1A4343**
S. SUCHITRADEVI	**21AR1A4351**
K. RAMU	**21AR1A4332**

ABSTRACT

Autonomous navigation and obstacle avoidance are critical aspects of modern robotics, with applications in industries such as surveillance, automation, and smart mobility. This project focuses on designing and implementing an **Obstacle Detection and Avoidance Robot** that autonomously navigates its environment while avoiding obstacles in real time. The system is built using an **Arduino Uno** microcontroller, an **L293D motor driver**, **ultrasonic sensors**, **gear motors with robot wheels**, "and additional components such as a servo motor for directional scanning and an HC-05 Bluetooth module for remote control. The obstacle detection mechanism is powered by **ultrasonic sensors**, which continuously scan the environment and relay distance measurements to the Arduino. The microcontroller processes this data and, based on predefined conditions, commands the motor driver to adjust the robot's movement accordingly. The robot employs a combination of **forward, backward, and sideward motion**, facilitated by the robot wheels, to efficiently navigate around obstacles. An **ultrasonic sensor** mounted on a **servo motor** enhances obstacle detection by scanning multiple directions. Additionally, a **Bluetooth module** allows for manual operation when required. This project demonstrates a practical application of **sensor integration, real-time data processing, and autonomous control** in robotics. It offers potential use cases in autonomous delivery systems, smart transportation, and security patrolling. Future enhancements could include the integration of **computer vision and machine learning** to enable more sophisticated navigation and decision-making. Overall, this robot provides an efficient and reliable solution for autonomous obstacle avoidance, contributing to advancements in intelligent robotic systems.

INDEX

CHAPTER 1
INTRODUCTION

1. INTRODUCTION

1.1 Motivation

With advancements in robotics and automation, there is a growing need for intelligent systems that can operate independently and adapt to their surroundings. The motivation behind this project is to develop an autonomous robot capable of detecting and avoiding obstacles in real time, making it suitable for various applications such as surveillance, smart mobility, and automation. Autonomous robots are widely used in industries, logistics, and security, reducing human effort and increasing efficiency. The development of such intelligent systems is essential for modern technological progress and has a vast potential for future enhancements with AI and machine learning integration.

1.2 Problem Statement

In many environments, obstacles pose a challenge to autonomous navigation. Traditional robots often lack efficient obstacle detection and avoidance mechanisms, leading to collisions and operational inefficiencies. Without proper navigation, these robots may fail to perform their intended tasks, resulting in safety hazards and mechanical damage. This project aims to design a robot that can detect obstacles and navigate safely without human intervention, ensuring smooth movement and reliable automation in complex environments.

1.3 Objective of the Project

The main objectives of this project are:

- To develop an autonomous robot capable of detecting and avoiding obstacles.
- To integrate sensors and control systems for real-time decision-making.
- To enable smooth navigation using omnidirectional wheels for better Adaptability.
- To provide remote control functionality via Bluetooth for manual operation if required.
- To enhance safety and efficiency in automation systems by reducing collision risks.

1.4 Scope

This project focuses on the design and implementation of an obstacle detection and avoidance robot using Arduino. It covers hardware components like ultrasonic sensors, servo motors, and motor drivers, along with software algorithms for real-time navigation. The system will be

designed to work in indoor environments, such as warehouses, offices, and homes, where avoiding obstacles is crucial. Future enhancements may include AI-based decision-making, camera-based obstacle detection, and GPS integration for outdoor navigation, making the robot even more versatile. The project also lays the groundwork for further research in autonomous robotics and automation systems.

1.5 Project Introduction

The Obstacle Detection and Avoidance Robot is an autonomous system designed to navigate through environments while avoiding obstacles. It uses ultrasonic sensors to detect objects in its path, a servo motor to scan surroundings, and a motor driver to control movement. The robot can operate independently and also be controlled remotely via Bluetooth, making it versatile for various applications such as security, automation, and smart mobility. With the rise of intelligent automation, such systems are becoming crucial for enhancing productivity and safety. This project provides a cost-effective and efficient solution for various real-world applications where autonomous navigation is required. It represents a step towards the future of smart robotic systems in industries and everyday life.

CHAPTER 2
LITERATURE SURVEY

2. LITERATURE SURVEY

A literature survey is essential to understand the existing research, technologies, and methodologies related to obstacle detection and avoidance robots. This section reviews various studies, innovations, and technical advancements that contribute to the development of autonomous robots with efficient navigation capabilities.

2.1 Overview of Obstacle Avoidance in Robotics

Obstacle avoidance is a critical functionality in autonomous mobile robots, enabling them to navigate safely without collisions. Various research studies and technological advancements have focused on improving real-time object detection and navigation techniques. The use of ultrasonic sensors, infrared sensors, LiDAR, and AI-based models has been explored in different robotic systems to ensure accurate and dynamic movement.

2.2 Existing Technologies and Approaches

Several methodologies have been explored for obstacle detection and avoidance:

- **Sensor-Based Navigation:** Ultrasonic, infrared, and LiDAR sensors are widely used to detect obstacles and adjust movement accordingly.
- **Arduino-Based Navigation:** Low-cost microcontrollers like Arduino are widely implemented due to their ease of use and affordability.

2.3 Research Papers on Obstacle Avoidance Robots

Several research papers have focused on designing and improving obstacle avoidance robots. Below are some notable works:

1. **"An IoT-Based Obstacle Avoidance Robot Using Ultrasonic Sensor and Arduino" – Mothe et al., 2020**

   This paper presents an IoT-integrated autonomous robot that utilizes ultrasonic sensors, Arduino Uno, and an L293D motor driver for real-time obstacle avoidance. The study highlights the efficiency of wireless communication in ensuring safe movement and remote monitoring. The integration of IoT allows for real-time data analysis and improved decision-making in navigation. The research also discusses the advantages of

using low-cost and energy-efficient components, making it a viable solution for smart automation applications.

2. **"Obstacle Avoidance Robot Using Arduino" – Pavithra A C and Subramanya Goutham V, 2018**

   The study discusses an Arduino-based robotic vehicle equipped with ultrasonic sensors, an HC-05 Bluetooth module, servo motors, and an L293D motor driver. The research focuses on the robot's capability to autonomously navigate and avoid obstacles while maintaining efficient communication with external devices via Bluetooth. The low-cost and efficient design make it a suitable solution for applications in education, surveillance, and automation. The study also explores the role of microcontrollers in processing sensor data for real-time decision-making.

3. **"Obstacle Avoiding Robotic Vehicle with Arduino and Ultrasonic Sensor" – BalaKrishna K, Rachananjali K, and CH.N. Narasimha Rao, 2022**

   This study presents a real-time obstacle detection and avoidance system using ultrasonic and infrared sensors. The research emphasizes the importance of low power consumption and efficient processing to enhance the robot's performance. The paper discusses the integration of various sensors and their role in providing accurate distance measurements to prevent collisions. Additionally, the study highlights challenges in sensor accuracy and ways to optimize algorithmic processing for real-world applications.

2.4 Applications of Obstacle Avoidance Robots

Autonomous robots with obstacle avoidance capabilities are used in:

- **Industrial Automation:** Robots navigate and transport goods in warehouses.
- **Security and Surveillance:** Autonomous patrolling robots detect intrusions and navigate safely.
- **Smart Transportation:** Self-driving vehicles utilize obstacle avoidance for safe driving.
- **Healthcare Assistance:** Robots help elderly and disabled individuals navigate safely.

2.5 Challenges in Obstacle Avoidance Robotics

- **Accuracy in Detection:** Ultrasonic sensors may struggle with transparent or very thin objects.
- **Real-Time Processing:** Efficient algorithms are required for high-speed decision-making.
- **Energy Consumption:** Continuous sensor operation demands optimized power management.

2.6 Future Enhancements and Research Directions

- **AI and Machine Learning:** Integrating AI for improved decision-making in dynamic environments.
- **Advanced Sensors:** Using LiDAR and computer vision for precise object detection.
- **IoT and 5G Connectivity:** Enabling real-time monitoring and remote control.

CHAPTER 3
SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Existing System

The existing obstacle avoidance robotic systems primarily rely on basic sensor-based navigation with limited intelligence. These systems use ultrasonic sensors, infrared sensors, and Arduino microcontrollers to detect obstacles and adjust the robot's movement accordingly. However, they have certain limitations, such as lack of advanced decision-making capabilities and real-time remote control.

3.2 Disadvantages

- Limited decision-making ability.
- Inability to detect certain types of obstacles (transparent or very thin objects).
- Lack of real-time remote control or monitoring features.

3.3 Proposed System

The proposed obstacle avoidance robot integrates advanced technologies such as IoT, Bluetooth communication, and improved sensor networks to enhance navigation capabilities. The system utilizes an Arduino Uno microcontroller, an HC-05 Bluetooth module for remote monitoring, L293D motor driver for efficient movement control, and ultrasonic sensors for precise obstacle detection.

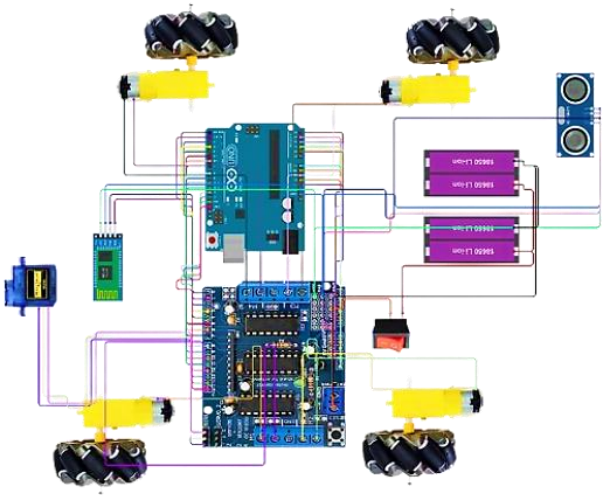


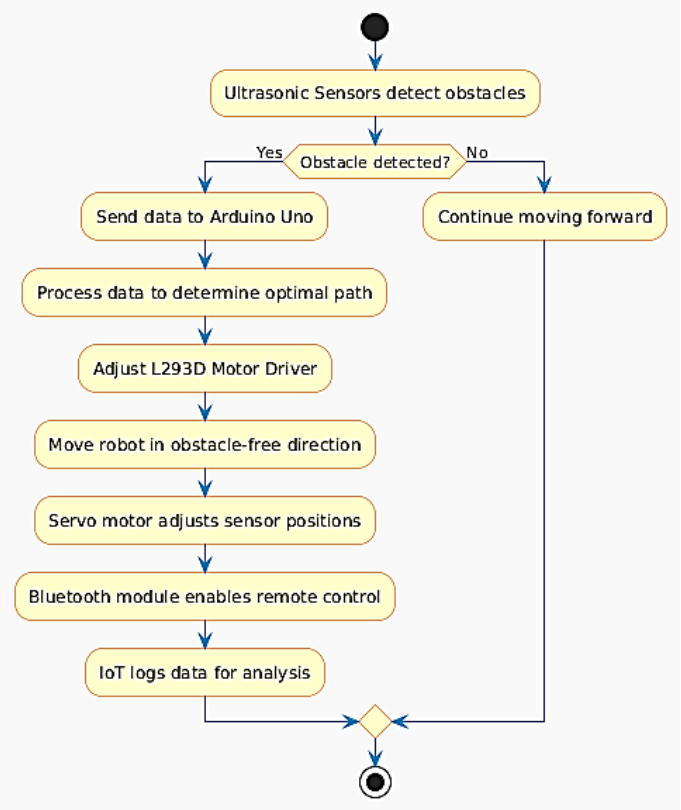
Fig. Circuit diagram of Proposed system architecture

3.4 Advantages

- Improved obstacle detection accuracy using multiple sensors.
- Real-time monitoring and remote-control using Bluetooth.
- Efficient navigation with AI-enhanced decision-making.
- Optimized power consumption for extended operational time.

3.5 Work Flow of Proposed System

- The ultrasonic sensors continuously detect obstacles in the environment.
- The sensor data is processed by the Arduino Uno microcontroller, which determines the optimal path to avoid collisions.
- The L293D motor driver controls the movement of the robot's gear motors.



- The HC-05 Bluetooth module enables remote control and monitoring of the robot's movement.
- A servo motor adjusts sensor positions for enhanced obstacle detection in multiple directions.
- IoT integration allows real-time data logging and decision-making enhancements.

CHAPTER 4
REQUIREMENT ANALYSIS

4. REQUIREMENT ANALYSIS

The requirement analysis involves identifying the essential hardware and software components needed for the successful implementation of the proposed obstacle avoidance robotic system. It ensures that the system meets the functional and non-functional requirements necessary for smooth operation.

4.1 Functional Requirements

- The robot must autonomously detect and avoid obstacles.
- It should support remote control via Bluetooth communication.
- The system must provide real-time feedback on detected obstacles.
- The servo motor should scan the environment for a better navigation path.
- The IoT platform should store and analyze real-time data.

4.2 Non-Functional Requirements

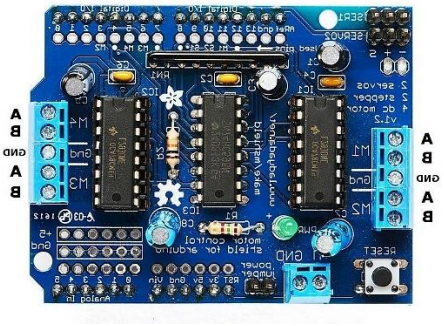
- The system should be power-efficient to extend battery life.
- The robot should have minimal latency in response to obstacles.
- The Bluetooth communication must be stable within a defined range.
- The system should be lightweight and easy to assemble.
- The IoT platform should have a user-friendly interface for data visualization.

4.3 Hardware Requirements

1. Arduino Uno Microcontroller

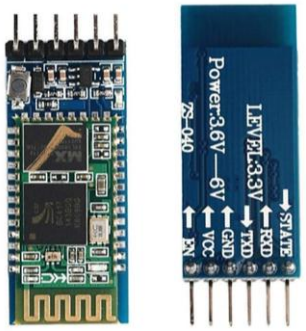
2. Ultrasonic Sensors (HC-SR04)

3. L293D Motor Driver



4. Gear Motors (x4) and Wheels

5. HC-05 Bluetooth Module



6. Servo Motor

7. Li-ion Battery (x2) & Battery Holder

8. Connecting Wires & Jumper Wires

9. Switch

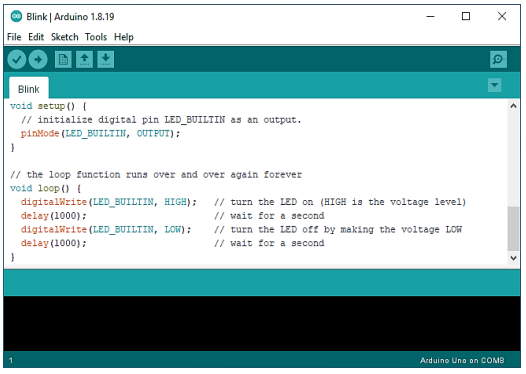
10. Male Berg Strip

11. cardboard Board

12. Soldering Iron & Wire, Hot Glue Gun & Tubes

4.4 Software Requirements

1. **Arduino IDE -** Used for writing, compiling, and uploading the microcontroller code.



2. **Embedded C Programming**

3. **Bluetooth Control Application**

CHAPTER 5
SYSTEM DESIGN

SYSTEM DESIGN

The **System Design** section provides a comprehensive overview of the architectural and functional structure of the **Obstacle Avoidance Robot**. It consists of the following key components:

- **System Architecture** – Represents the high-level structure and interaction between various hardware and software components.
- **UML Diagrams** – Illustrate system behaviour, including use case, sequence, and class diagrams.
- **Data Flow Diagrams (DFD)** – Depict the flow of data between different modules of the system.

5.1 SYSTEM ARCHITECTURE

The **System Architecture Diagram** of the Obstacle Avoidance Robot provides a visual representation of how different hardware and software components interact to ensure obstacle detection and autonomous movement. The diagram is divided into four main subsystems:

1. User Interface

- The **Bluetooth App** serves as the primary interface for the user to send commands to the robot.
- It communicates wirelessly with the **HC-05 Bluetooth Module** integrated into the robot system.

2. Robot System

This section contains the core processing and sensing components:

- **Arduino Uno (MCU)** – The central controller responsible for processing inputs from sensors and executing movement commands.
- **Ultrasonic Sensor** – Detects obstacles in the robot's path and sends signals to the Arduino Uno for processing.
- **HC-05 Bluetooth Module** – Receives commands from the user via the Bluetooth app.

- **L293D Motor Driver** – Controls the movement of the gear motors based on Arduino's instructions.
- **Servo Motor** – Adjusts the direction of the ultrasonic sensor for better obstacle detection.
- **IoT Platform** – Logs data for remote monitoring or future enhancements.

3. Actuation System

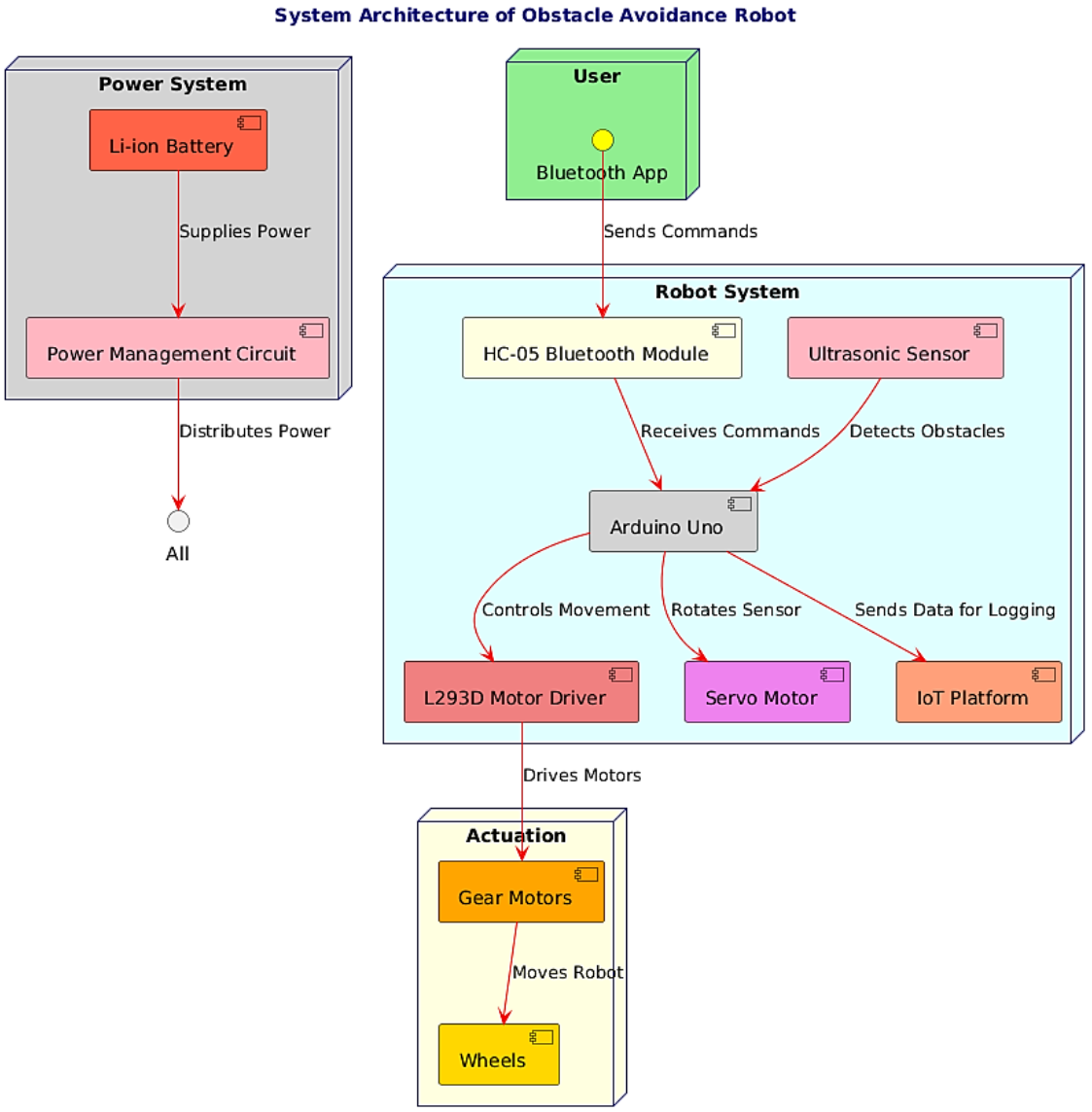
- **Gear Motors** – Receive movement instructions from the motor driver and drive the wheels accordingly.
- **Wheels** – Provide mobility to the robot, allowing it to navigate obstacles effectively.

4. Power System

- **Li-ion Battery** – The main power source supplying energy to all components.
- **Power Management Circuit** – Regulates and distributes power efficiently to various modules.

Data Flow and Interaction

- The **Bluetooth App** sends movement commands to the **HC-05 Bluetooth Module**.
- The **Ultrasonic Sensor** continuously detects obstacles and relays this data to the **Arduino Uno**.
- The **Arduino Uno** processes sensor inputs and user commands to control the **L293D Motor Driver** and **Servo Motor**.
- The **Motor Driver** actuates the **Gear Motors**, which in turn drive the **Wheels** to move the robot.
- The **IoT Platform** stores data, which can be analyzed for performance monitoring.
- The **Power System** ensures a stable power supply to all modules.



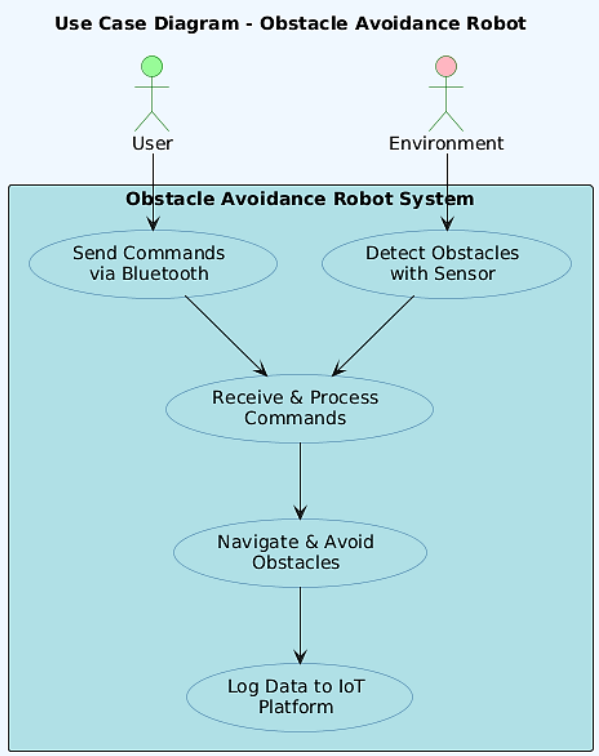
Colour Coding and Visual Representation

- Each subsystem in the **System Architecture Diagram** is visually distinguished using color coding for better clarity.
- The **User Interface (Green)** highlights the interaction between the user and the robot.
- The **Robot System (Blue)** encloses the main electronic components responsible for decision-making.
- The **Actuation System (Yellow-Orange)** represents the mechanical movement of the robot.
- The **Power System (Red-Pink)** ensures energy distribution and stability.
- Arrows with **Red Lines** indicate data/control flow between different modules.

5.2 UML DIAGRAMS

5.2.1 USE CASE DIAGRAM

The **Use Case Diagram** represents how external entities (actors) interact with the Obstacle Avoidance Robot System.



Actors:

1. **User** (Green)
    - The user interacts with the system through a **Bluetooth App**.
    - Sends movement commands to control the robot.
2. **Environment** (Pink)
    - The environment affects the robot's navigation by providing obstacles.
    - The robot detects these obstacles using sensors.

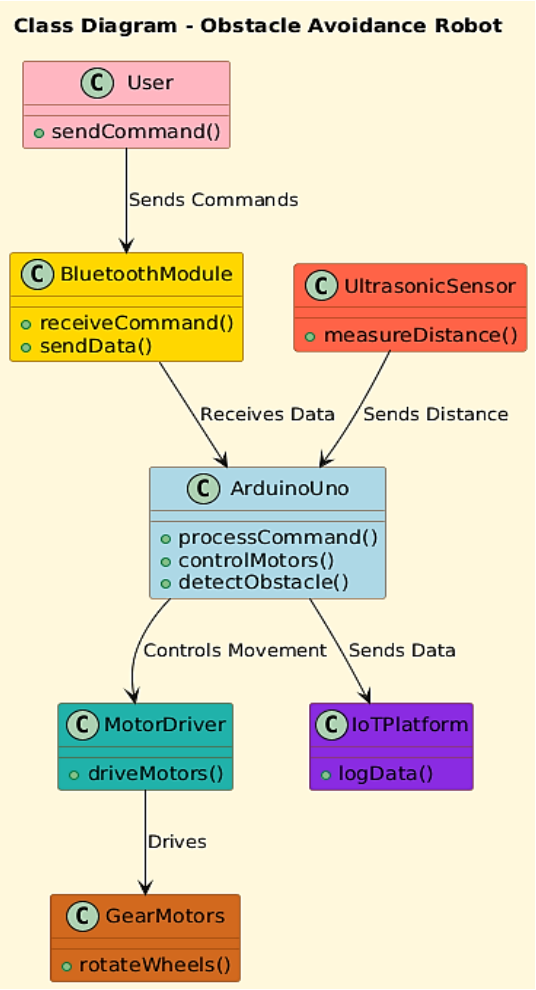
System Functionalities (Use Cases):

1. **Send Commands via Bluetooth:**
    - The user sends movement commands using a **mobile application**.
2. **Detect Obstacles with Sensor:**
    - The **Ultrasonic Sensor** detects nearby obstacles.

3. **Receive & Process Commands:**
    - The robot processes user commands and sensor input to decide the next action.
4. **Navigate & Avoid Obstacles:**
    - The robot turns based on sensor readings to avoid collisions.

5.2.2 CLASS DIAGRAM

This class diagram represents an **Obstacle Avoidance Robot**, illustrating interactions between sensors, controllers, and movement modules. The robot processes user commands, detects obstacles, and navigates autonomously.



Classes and Functions

1. **User**: Sends movement commands via sendCommand().
2. **BluetoothModule**: Receives commands (receiveCommand()) and sends data (sendData()) to **ArduinoUno**.

3. **UltrasonicSensor**: Measures distances via measureDistance().
4. **ArduinoUno**:
    - Processes commands (processCommand()).
    - Controls motors (controlMotors()).
    - Detects obstacles (detectObstacle()).
5. **MotorDriver**: Drives motors through driveMotors().
6. **GearMotors**: Rotates wheels using rotateWheels().
7. **IoTPlatform**: Logs operational data via logData().

Working Mechanism

1. **User** sends commands via **BluetoothModule**.
2. **BluetoothModule** transmits commands to **ArduinoUno**.
3. **UltrasonicSensor** detects obstacles and sends data.
4. **ArduinoUno** processes inputs and controls **MotorDriver**.
5. **MotorDriver** moves the **GearMotors** accordingly.
6. **IoTPlatform** logs data for analysis.

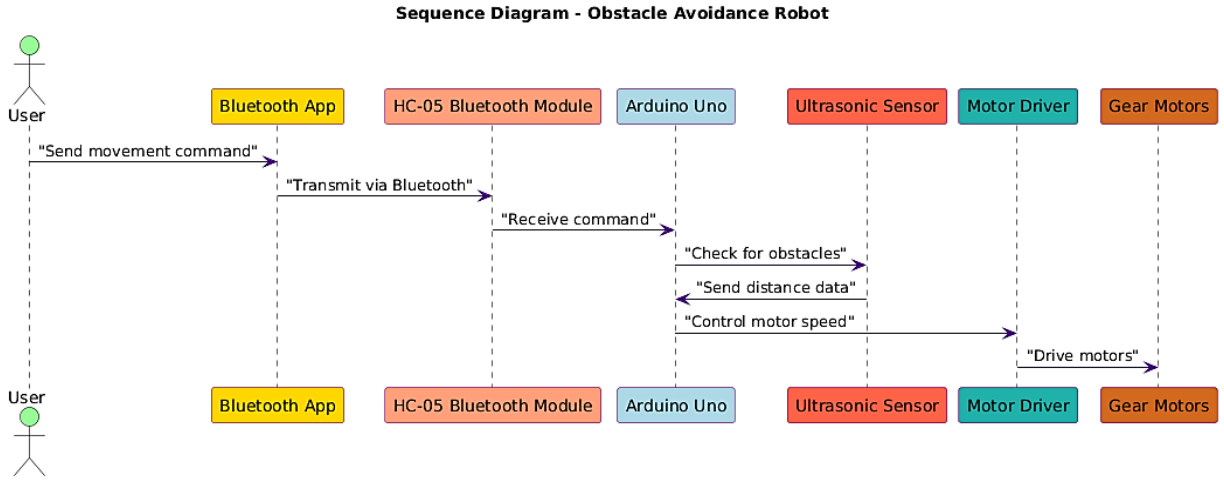
5.2.3 SEQUENCE DIAGRAM

The sequence diagram represents the step-by-step interaction between components of the **Obstacle Avoidance Robot**, ensuring smooth command execution and obstacle detection.

Sequence of Operations

1. **User** sends a movement command via the **Bluetooth App**.
2. **Bluetooth App** transmits the command via the **HC-05 Bluetooth Module**.
3. **Arduino Uno** receives the command and checks for obstacles using the **Ultrasonic Sensor**.

4. **Ultrasonic Sensor** sends distance data back to **Arduino Uno**.
5. **Arduino Uno** processes the data and controls motor speed accordingly.
6. **Motor Driver** receives speed control instructions and drives the **Gear Motors**.
7. The **Gear Motors** rotate, enabling robot movement.



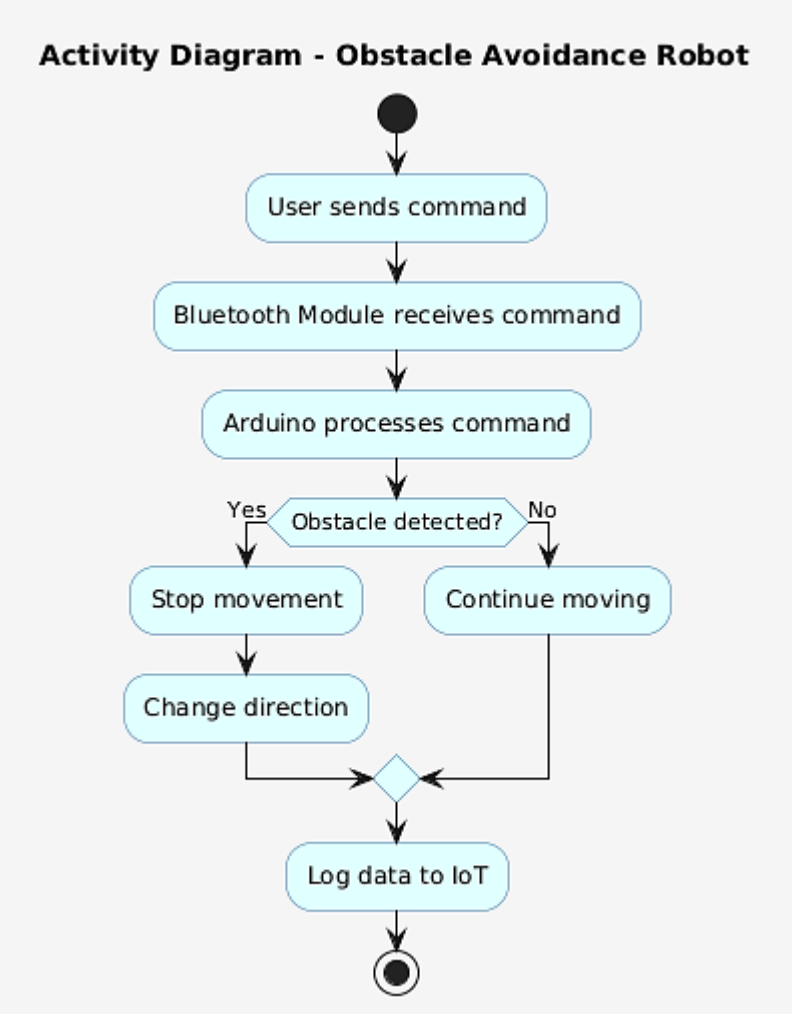
5.2.4 ACTIVITY DIAGRAM

The following activity diagram represents the workflow of an obstacle avoidance robot. The robot follows a sequence of steps to process commands and respond to obstacles efficiently.

Workflow Description:

1. **User sends command**: The user initiates a command to control the robot.
2. **Bluetooth Module receives command**: The command is transmitted via a Bluetooth module.
3. **Arduino processes command**: The Arduino microcontroller processes the received instruction.
4. **Obstacle detection**: The system checks if an obstacle is present.
    - If **Yes**, the robot stops movement and changes direction.
    - If **No**, the robot continues moving forward.

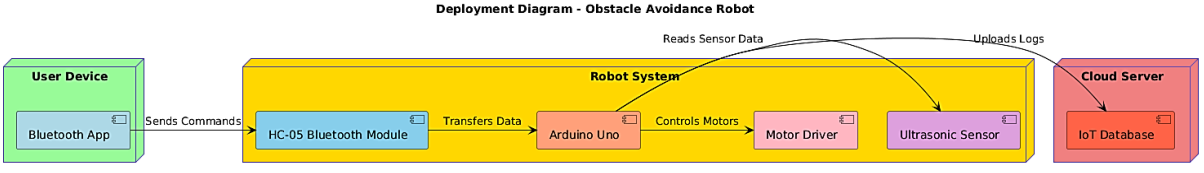
Diagram Representation:



5.2.5 DEPLOYMENT DIAGRAM

The **Deployment Diagram** represents the **hardware and software setup**:

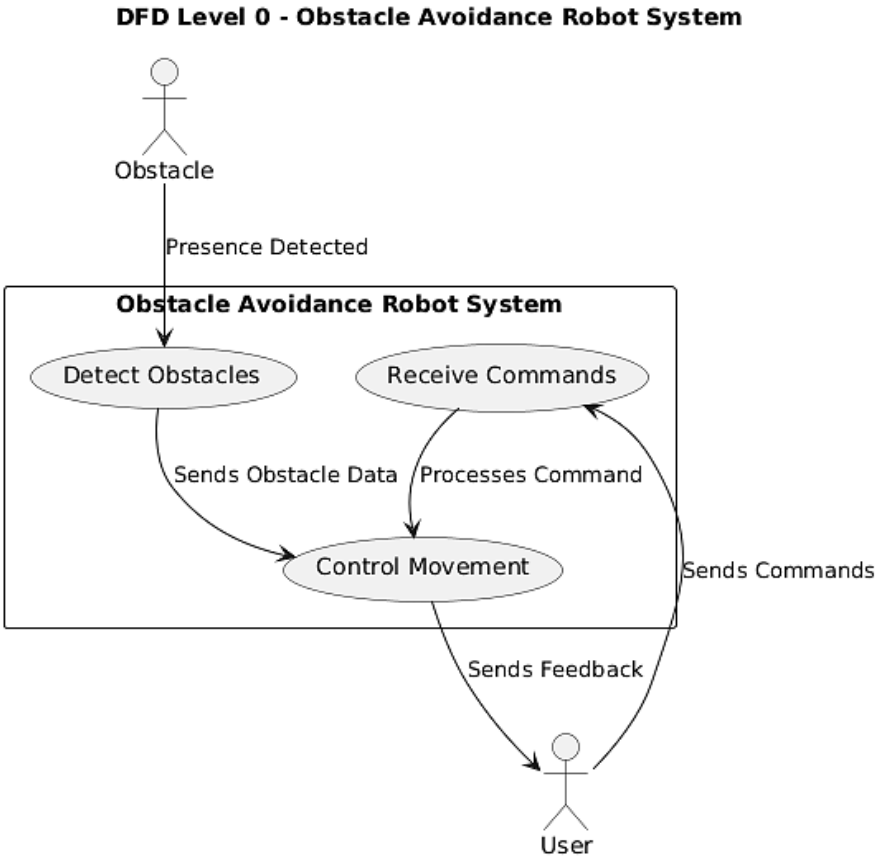
- The **User Device** runs a **Bluetooth App** that communicates with the **Robot System**.
- The **Robot System** consists of an **Arduino Uno**, **Bluetooth Module**, **Sensors**, and **Motor Driver**.
- The **Cloud Server** stores **sensor logs & movement data**.



5.3 DATA FLOW DIAGRAM (DFD)

5.3.1 Level 0: Context Diagram

The context diagram represents the entire system as a single process. It shows how the obstacle avoidance robot interacts with external entities like the user (via a Bluetooth App) and an IoT platform.



Entities:

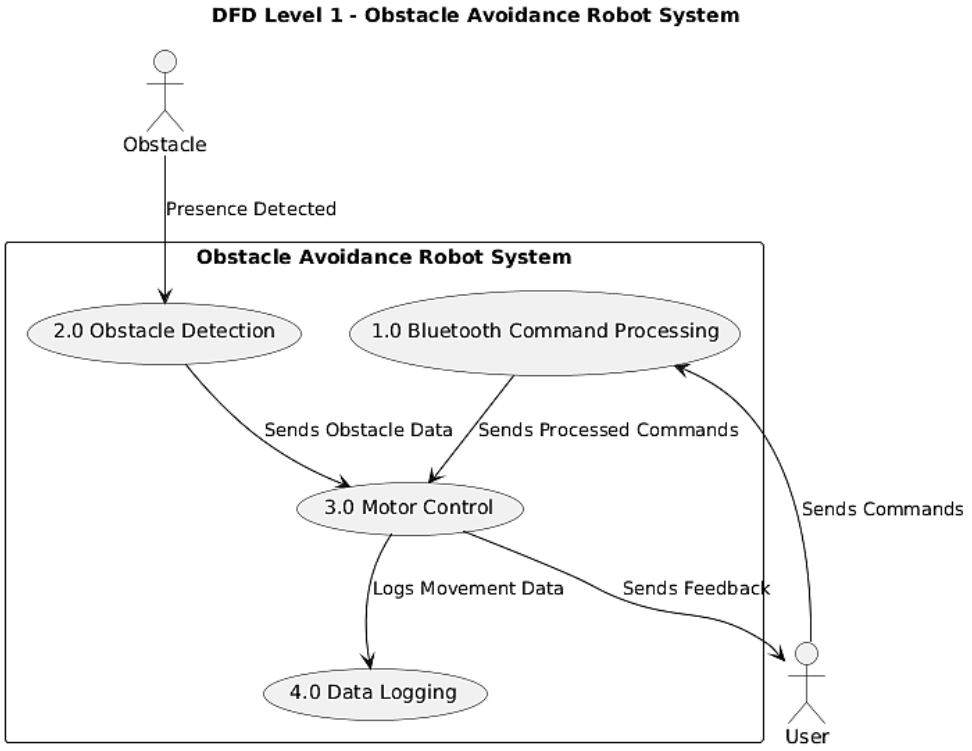
1. **User** – Sends movement commands through the Bluetooth App.
2. **IoT Platform** – Receives data from the robot for logging and analysis.

Process:

- The user sends movement commands to the robot via the Bluetooth App.
- The robot receives sensor data, processes it, and makes movement decisions.
- The robot transmits operational data to the IoT platform.

5.3.2 Level 1: High-Level Data Flow

The Level 1 diagram expands on the context diagram by breaking down the robot's functionality into subsystems.

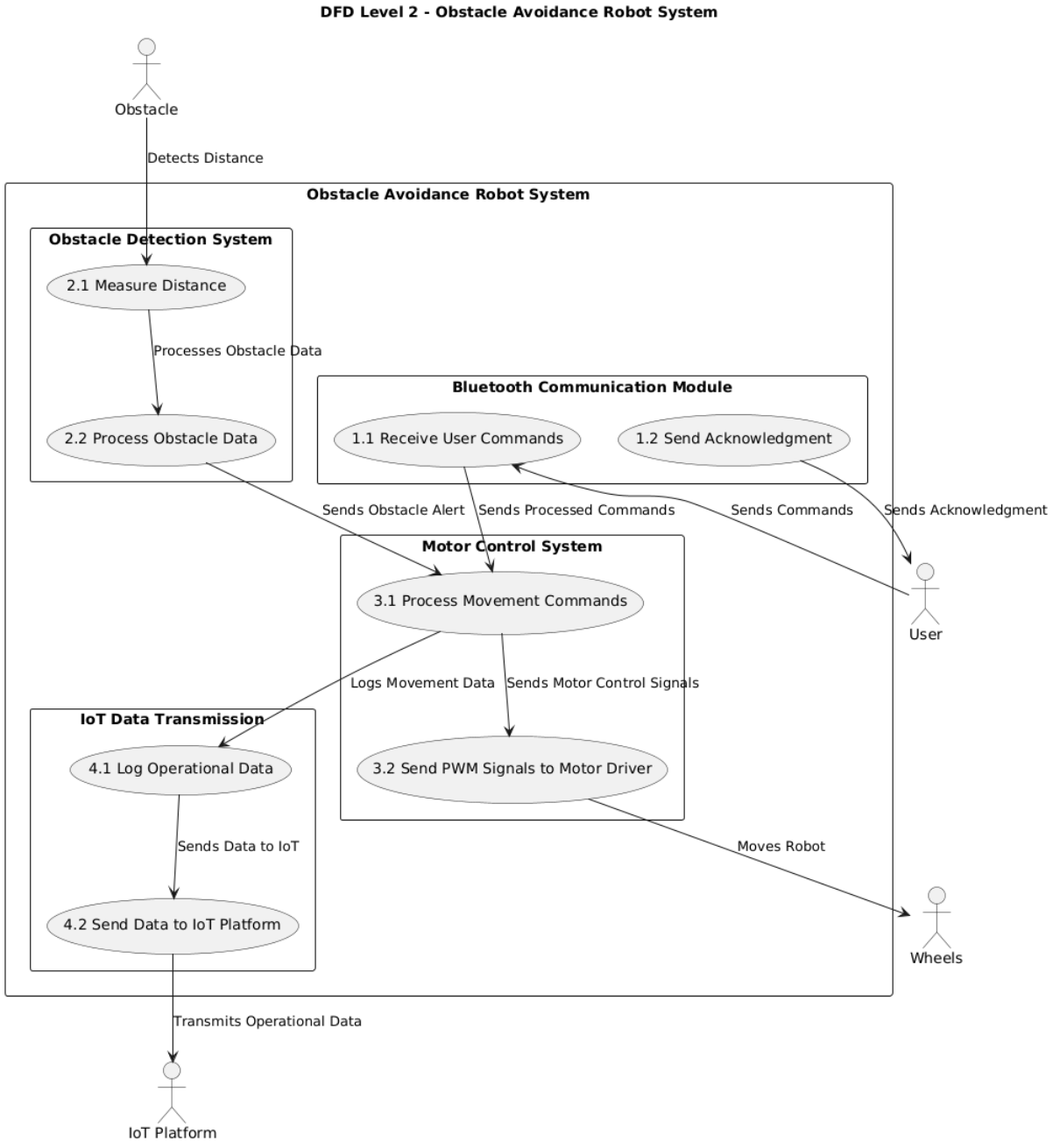


Main Processes:

1. **Bluetooth Communication Module**
    - Receives commands from the user via the Bluetooth App.
    - Sends these commands to the Arduino Uno for processing.
2. **Obstacle Detection System**
    - Uses the ultrasonic sensor to detect obstacles.
    - Sends distance measurements to the Arduino Uno.
3. **Motor Control System**
    - Arduino Uno processes user commands and sensor data.
    - Sends control signals to the motor driver to adjust movement.
4. **IoT Data Transmission**
    - The robot logs operational data (e.g., movements, obstacles detected).
    - Sends this data to the IoT platform for monitoring.

5.3.3 Level 2: Detailed Data Flow

The Level 2 diagram further details the interactions between components.



Subsystem Interactions:

- **User via Bluetooth App → Bluetooth Module**
  - Command Data: Move Forward, Turn Left, Stop, etc.
  - Response: Acknowledgment or status update.

- **Ultrasonic Sensor → Arduino Uno**
    - Obstacle Distance Data.
    - Arduino processes this data to decide whether to stop, turn, or continue moving.
- **Arduino Uno → Motor Driver**
    - Sends PWM signals to adjust speed and direction of motors.
- **Motor Driver → Wheels**
    - Executes movement based on signals received.
- **Arduino Uno → IoT Platform**
    - Sends real-time logs (e.g., battery status, movement logs, obstacle detection data).

CHAPTER 6
IMPLEMENTATION & RESULTS

5. IMPLEMENTATION

6.1 SOURCE CODE

```
#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;

long safeReadUltrasonic() {
  long sum = 0;
  for (int i = 0; i < 5; i++) {
    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);
    sum += pulseIn(Echo, HIGH, 20000);
    delay(20);
  }
  long avg = sum / 5;
  return avg / 29 / 2;
}

int lookAtAngle(int angle) {
  servo.write(angle);
  delay(400);
  return safeReadUltrasonic();
}

void moveForward() { M1.run(FORWARD); M2.run(FORWARD); M3.run(FORWARD);
M4.run(FORWARD); }
void moveBackward() { M1.run(BACKWARD); M2.run(BACKWARD); M3.run(BACKWARD);
M4.run(BACKWARD); }
```

```
void turnLeft() { M1.run(FORWARD); M2.run(FORWARD); M3.run(BACKWARD);
M4.run(BACKWARD); }
void turnRight() { M1.run(BACKWARD); M2.run(BACKWARD); M3.run(FORWARD);
M4.run(FORWARD); }
void stopBot() { M1.run(RELEASE); M2.run(RELEASE); M3.run(RELEASE);
M4.run(RELEASE); }

void dynamicBackward(long frontDistance) {
  int backDelay = map(frontDistance, 0, 50, 200, 500);
  moveBackward();
  delay(backDelay);
  stopBot();
}

void backupRecovery() {
  stopBot();
  moveBackward();
  delay(300);
  stopBot();
  int randomTurn = random(0, 2);
  if (randomTurn == 0) { turnLeft(); delay(600); } else { turnRight();
delay(600); }
  stopBot();
}

void BluetoothAndVoice() {
  if (Serial.available() > 0) {
    value = Serial.read();
    if (value == 'F') moveForward();
    else if (value == 'B') moveBackward();
    else if (value == 'L') turnLeft();
    else if (value == 'R') turnRight();
    else if (value == 'S') stopBot();
  }
}

void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  servo.attach(motorPin);
  servo.write(CENTER_POS);

  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);

  randomSeed(analogRead(A2));
```

```
}

void loop() {
  BluetoothAndVoice();

  long distance = safeReadUltrasonic();

  if (distance <= OBSTACLE_THRESHOLD) {
    stopBot();
    dynamicBackward(distance);

    int leftDist = lookAtAngle(150);
    int rightDist = lookAtAngle(30);
    int frontDist = lookAtAngle(90);

    if (leftDist > rightDist && leftDist > OBSTACLE_THRESHOLD) {
      turnLeft(); delay(600);
    } else if (rightDist > OBSTACLE_THRESHOLD) {
      turnRight(); delay(600);
    } else if (frontDist > OBSTACLE_THRESHOLD) {
      moveForward(); delay(600);
    } else {
      // Rear simulation checks
      int rearLeftDist = lookAtAngle(170);
      int rearRightDist = lookAtAngle(10);

      if (rearLeftDist > rearRightDist && rearLeftDist > OBSTACLE_THRESHOLD) {
        turnLeft(); delay(700);
      } else if (rearRightDist > OBSTACLE_THRESHOLD) {
        turnRight(); delay(700);
      } else {
        backupRecovery();
      }
    }

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

6.2 HARDWARE IMPLEMENTATION

Hardware Implementation Process for Obstacle Detection and Avoidance Robot

6.2.1 Hardware Requirements:

1. Arduino Uno and its Cable
2. L293D
3. Ultrasonic Sensor
4. Servo Motor
5. HC-05 Bluetooth Module
6. Gear Motors x 4
7. Li-ion Battery x 2
8. Battery Holder
9. Connecting Wires
10. Jumper Wires Male to Female
11. Jumper Wires Female to Female
12. Switch
13. Robot Car Wheels
14. Male Berg Strip
15. Foam Board (*18x12cm)
16. Soldering Iron & Wire
17. Hot Glue Gun and Glue tubes
18. Star Screw Driver
19. Cutter

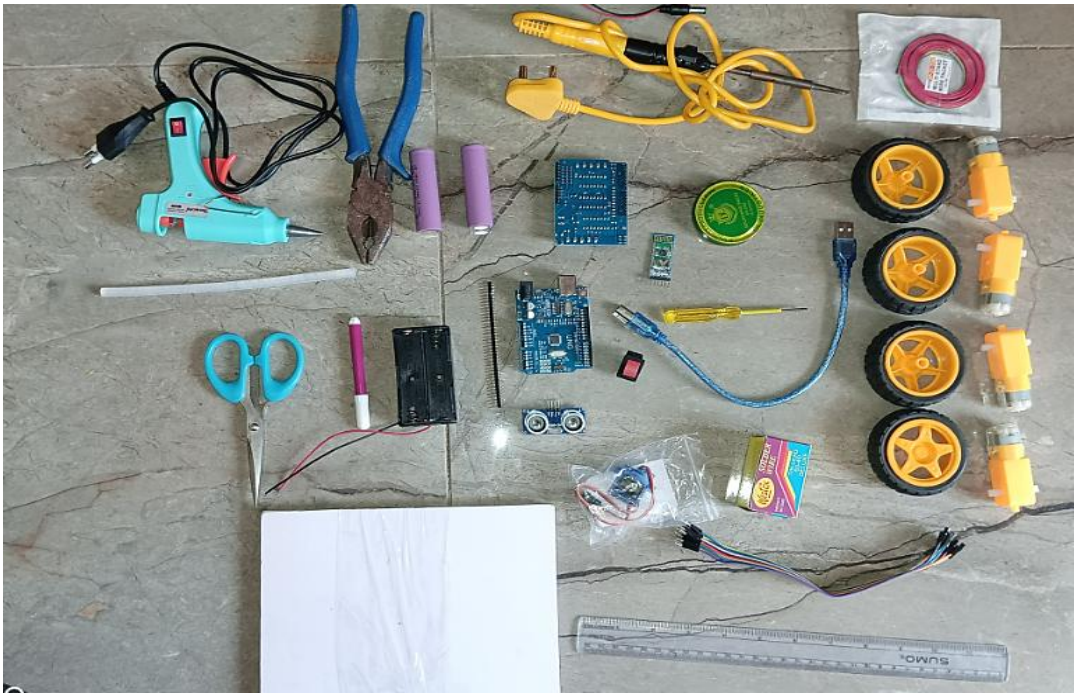
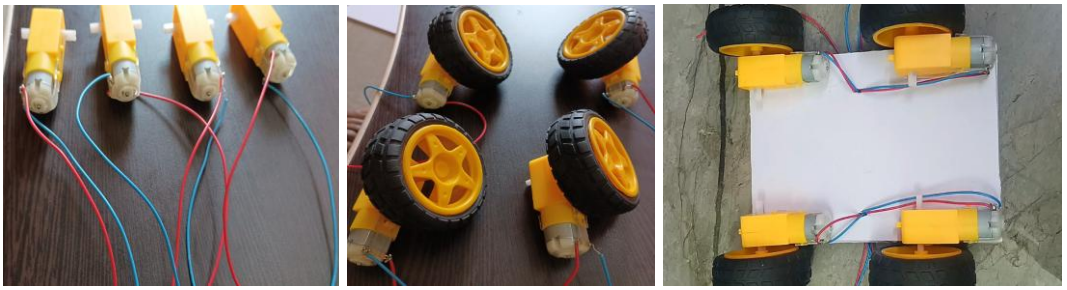


Fig. Hardware Requirements

1. Chassis Assembly

The base of the robot is constructed using a **cardboard plank** of dimensions *18x12 cm*.

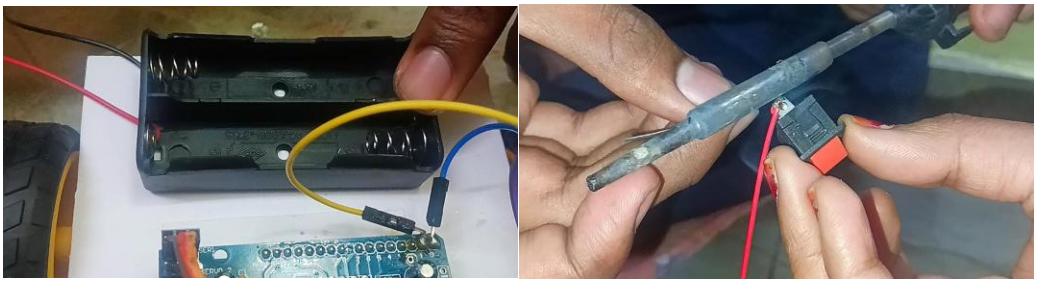
- The **gear motors** (4x) are securely fixed onto the chassis using a **hot glue gun**.
- The **robot car wheels** are attached to the motors, ensuring proper alignment.



2. Power System Setup

To power the robot, **Li-ion batteries** (2x) are connected appropriately:

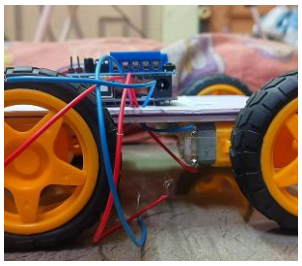
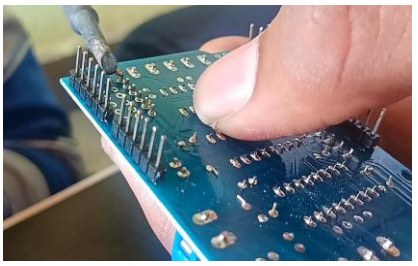
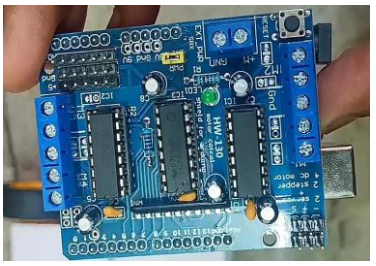
- The batteries are connected in **series or parallel**, depending on the voltage and current requirements.
- A **battery holder** is fixed onto the chassis using **hot glue** or adhesive tape to ensure stability.
- A **switch** is wired in series with the battery to control the power supply effectively.



3. Motor Driver and Control Unit Integration

The **L293D motor driver** is used to control the movement of the gear motors.

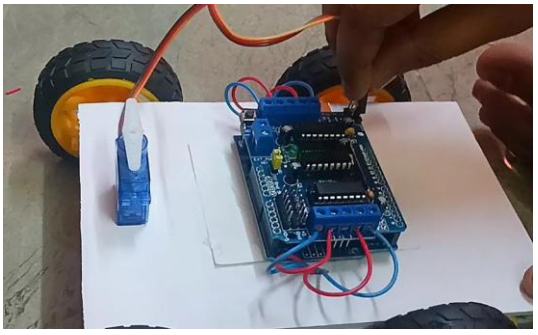
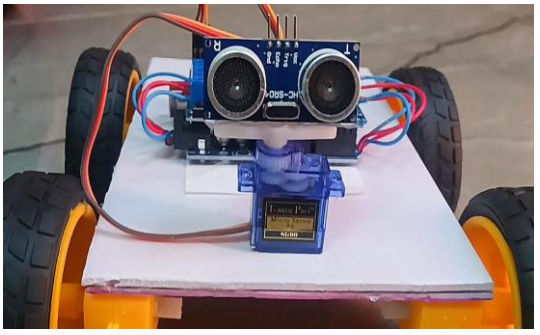
- **Male Berg Strips** are soldered onto the **L293D motor driver** for easy connectivity.
- The **gear motors** are connected to the L293D driver.
- **Jumper wires** are used to establish connections between the L293D and **Arduino Uno**, ensuring proper control signals.

4. Ultrasonic Sensor and Servo Motor Installation

For obstacle detection, an **ultrasonic sensor** is mounted on a **servo motor** to scan the surroundings.

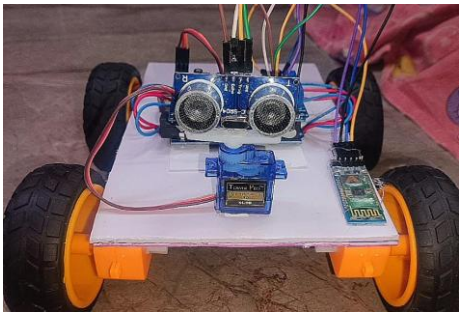
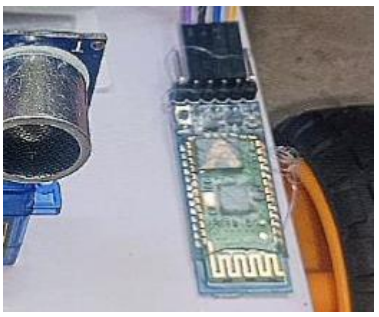
- The **ultrasonic sensor** is attached to the **servo motor** using **hot glue** or adhesive tape.
- The **servo motor** is then securely mounted at the front of the **cardboard chassis**.
- Proper wiring is done to connect both the ultrasonic sensor and the servo motor to the **Arduino Uno** using **jumper wires**.

5. Bluetooth Module Connection

A **HC-05 Bluetooth module** is used for wireless communication.

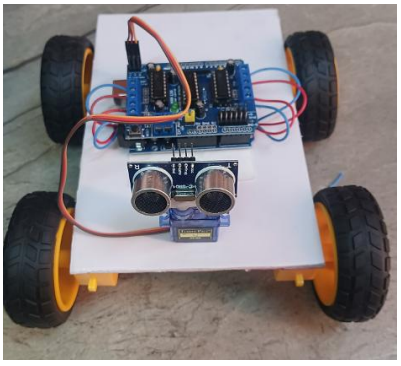
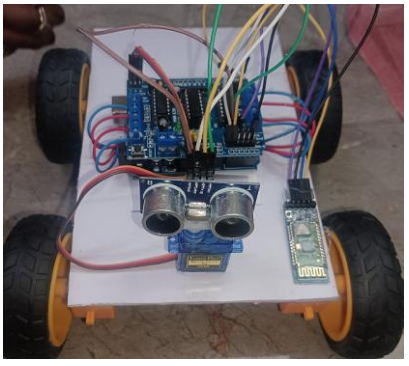
- The **HC-05 module** is connected to the **Arduino Uno** using **jumper wires**.
- Proper TX-RX and RX-TX connections are made to ensure smooth data transmission.
- The module is placed securely to avoid disconnections due to vibrations.

6. Wiring and Soldering

To ensure stable and efficient performance, all electrical connections are made carefully.

- **Connecting wires** are used to establish connections between components.
- **Soldering iron and wire** are used to strengthen loose connections.
- **Electrical tape** is used to insulate the connections, preventing short circuits and ensuring safety.

7. Testing and Calibration

Before finalizing the setup, thorough testing is conducted to ensure all components work correctly:

- The power system is checked, and the **motors' functionality** is tested.
- The **ultrasonic sensor** readings are verified, and the **servo motor's** rotation is calibrated.
- **Bluetooth connectivity** is tested to confirm communication with the mobile app or remote control.
- The **obstacle detection and avoidance behaviour** are observed and fine-tuned for optimal performance.

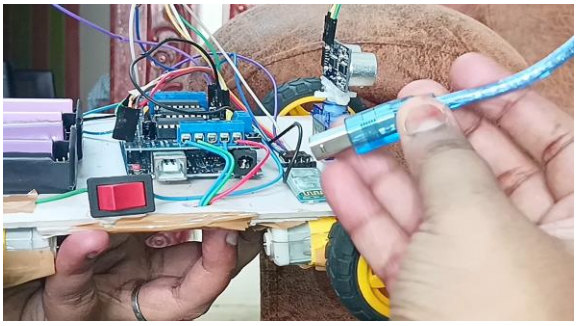
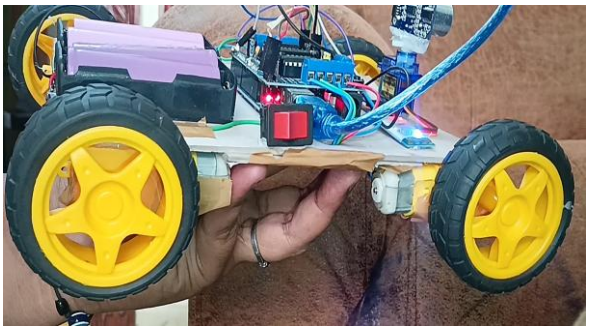


6.3 RESULTS

6.3.1 Code Execution Output (Serial Monitor)

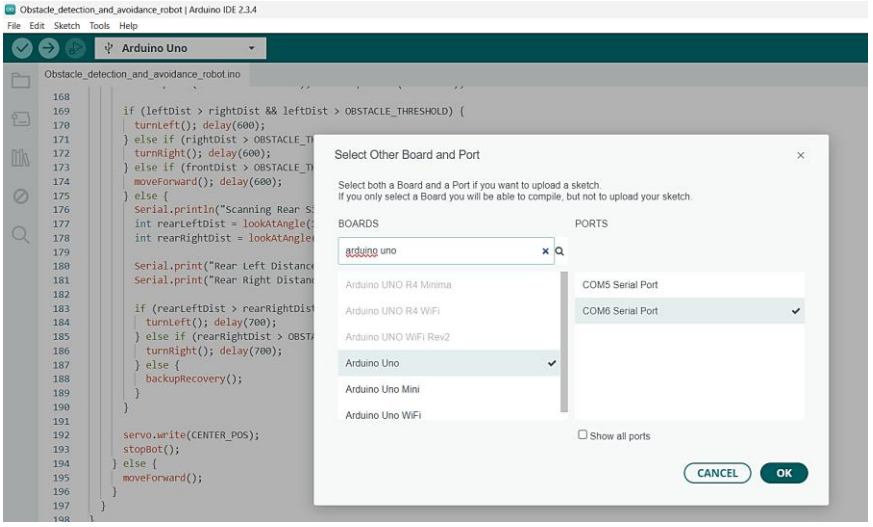
Step 1: Connecting the USB Cable



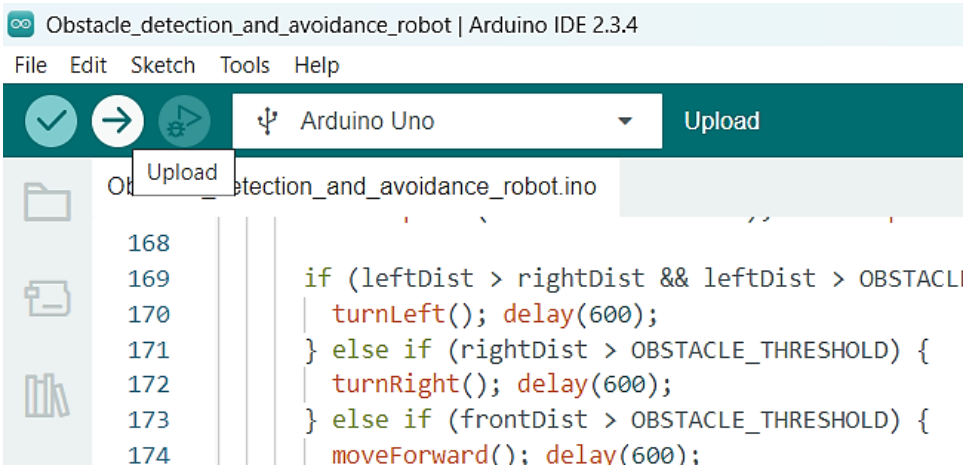



- *Take a **USB Type-B cable** and insert one end into your **Arduino Uno** board.*
- *Connect the other end to your **laptop's USB port** for programming and power supply.*

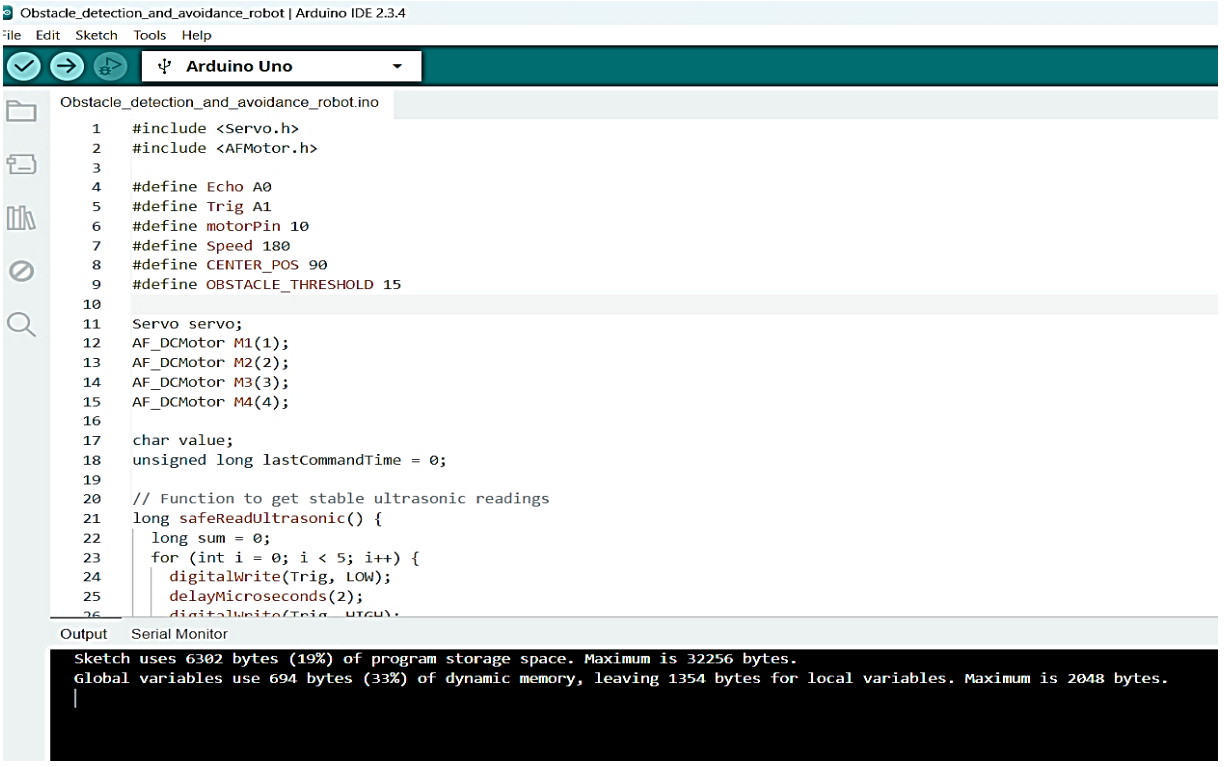
Step 2: Uploading the Code to Arduino

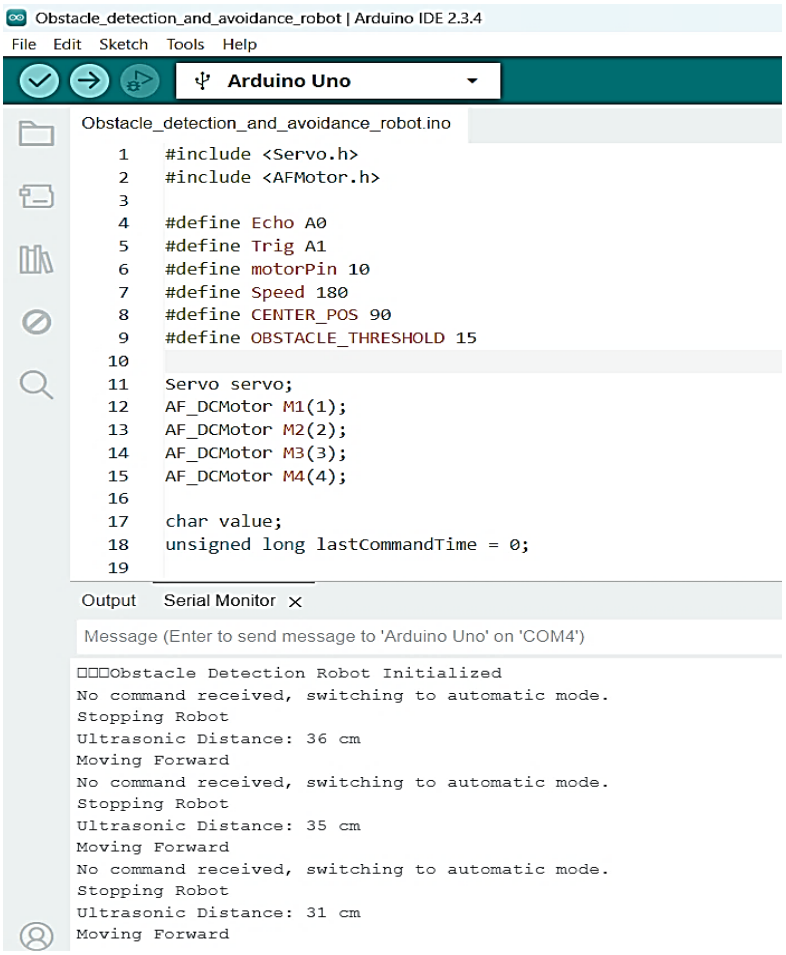


- *Open **Arduino IDE** on your laptop.*
- *Select the correct **Board** (Arduino Uno) and **Port**.*



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





Output Serial Monitor ×

Message (Enter to send message to 'Arduino Uno' on 'COM5')

```
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
```

CHAPTER 7

METHODOLOGY AND ALGORITHMS

7. METHODOLOGY AND ALGORITHMS

7.1 Methodology

- The obstacle detection and avoidance robot is designed to autonomously navigate its environment while avoiding obstacles using ultrasonic sensors and motor controls.
- The system integrates servo motors, an ultrasonic sensor, and an Arduino-based control mechanism to detect objects and alter the robot's path accordingly. The main features include:

1. **Ultrasonic Sensing**: Measures distances to obstacles and determines safe navigation paths.
2. **Motor Control**: Four DC motors enable movement in multiple directions.
3. **Decision-Making Algorithm**: Uses predefined conditions to decide whether to move forward, turn, or stop.
4. **Bluetooth and Voice Control**: Allows manual control using serial communication.
5. **Backup Recovery Mechanism**: Ensures movement even in complex environments by taking alternative paths when obstacles are detected.
6. **Adaptive Turning**: Adjusts turn angles dynamically based on detected obstacles.
7. **Speed Adjustment**: Modifies the robot's speed depending on the proximity of detected obstacles.

- The robot continuously scans its environment using an ultrasonic sensor and makes movement decisions based on the detected distances.
- If an obstacle is within a predefined threshold, the robot dynamically determines the best direction to turn and avoids it. When multiple obstacles are present, it executes a backup recovery strategy.

7.2 Algorithms

1. Ultrasonic Distance Measurement

```
Function safeReadUltrasonic():
    sum = 0
    Repeat 5 times:
        Trigger ultrasonic pulse
        Measure echo response time
        Sum up the results
    Return average distance
```

- This function reads the ultrasonic sensor multiple times (5 times) and calculates the average distance to avoid errors.
- The sensor is triggered to send an ultrasonic pulse, and the return time of the echo is measured to calculate the obstacle's distance.
- The final average reading is returned for accuracy.

2. Obstacle Detection and Avoidance

```
        Move backward dynamically based on distance
        Measure left, right, and front distances
        If left is clear, turn left
        Else If right is clear, turn right
        Else If front is clear, move forward
        Else:
            Measure rear distances
            If rear left is clear, turn left
            Else If rear right is clear, turn right
            Else execute backup recovery
    Else:
        Move forward
```

- The robot continuously checks the distance using the ultrasonic sensor.
- If an obstacle is detected within the threshold distance, it stops and moves backward dynamically, adjusting the backward movement duration based on the distance.

- It then scans left, right, and front directions to determine the best route.
- If no clear path is found, it checks the rear left and rear right directions.
- If all paths are blocked, the **backup recovery mechanism** is triggered.

3. Dynamic Backward Movement

```
Function dynamicBackward(frontDistance):
    Calculate delay based on front distance
    Move backward for calculated time
    Stop the robot
```

- When an obstacle is too close, the robot moves backward for a duration proportional to the obstacle's distance.
- The further the obstacle, the longer it moves backward before stopping.

4. Backup Recovery Mechanism

```
Function backupRecovery():
    Stop the robot
    Move backward briefly
    Randomly choose left or right turn
    Execute the turn
    Stop the robot
```

- If no clear path is available, the robot executes a **random turn** (left or right) after moving slightly backward.
- This mechanism ensures that the robot does not get stuck in an enclosed area.

5. Bluetooth and Voice Control

```
Function BluetoothAndVoice():
    If data available in Serial:
        Read input
        If 'F', move forward
        If 'B', move backward
        If 'L', turn left
        If 'R', turn right
        If 'S', stop
```

- The robot can be controlled manually through **Bluetooth commands** received via a serial interface.
- The user can send commands ('F' for forward, 'B' for backward, 'L' for left, 'R' for right, 'S' for stop), and the robot executes the corresponding movement.

6. Adaptive Turning Algorithm

```
Function lookAtAngle(angle):
    Rotate servo to given angle
    Delay to allow stabilization
    Return measured distance using safeReadUltrasonic()
```

- This function allows the robot to scan its surroundings before deciding on a direction.

7. Randomized Escape Strategy

```
Function backupRecovery():
    Stop the robot
    Move backward briefly
    Generate a random number (0 or 1)
    If 0, turn left
    If 1, turn right
    Execute the turn
    Stop the robot
```

- Adds a random decision-making element when no clear path is available.

8.Rear Simulation Check (Multi-Directional Scanning)

```
Function loop():
    If all forward directions are blocked:
        Measure rear left and rear right distances
        If rear left is clear, turn left
        Else If rear right is clear, turn right
        Else execute backupRecovery()
```

- Ensures that the robot checks its rear surroundings when all front paths are blocked.

CHAPTER 8
TESTING

8. TESTING

Testing is a critical phase in the development of the **Obstacle Detection and Avoidance Robot** to ensure its functionality, reliability, and performance under different conditions. Various types of testing have been conducted to validate individual components, integration between different modules, and overall system performance.

8.1 Unit Testing

Objective: Unit testing focuses on testing individual components or modules of the robot separately to ensure they function correctly.

Testing Approach:

- The ultrasonic sensor module was tested independently by measuring distances at various angles.
- The motor driver was tested by sending commands to move the robot forward, backward, left, and right.
- The Bluetooth module was tested to verify that it correctly receives and executes remote commands.

Expected Outcome:

Ultrasonic sensor test output:

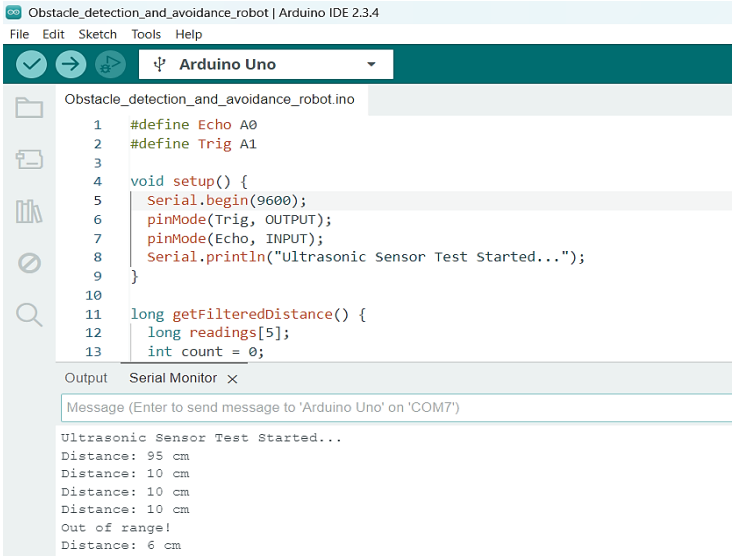


Fig.8.1.1 Unit Testing of Ultrasonic Sensor in Obstacle Detection and Avoidance Robot

Capture the serial monitor showing the distance measurements output from the ultrasonic sensor when the robot is tested in front of various objects.

Motor control test output:

Show the serial monitor or LED indicators verifying motor commands for forward, backward, left, right movements, and stopping.

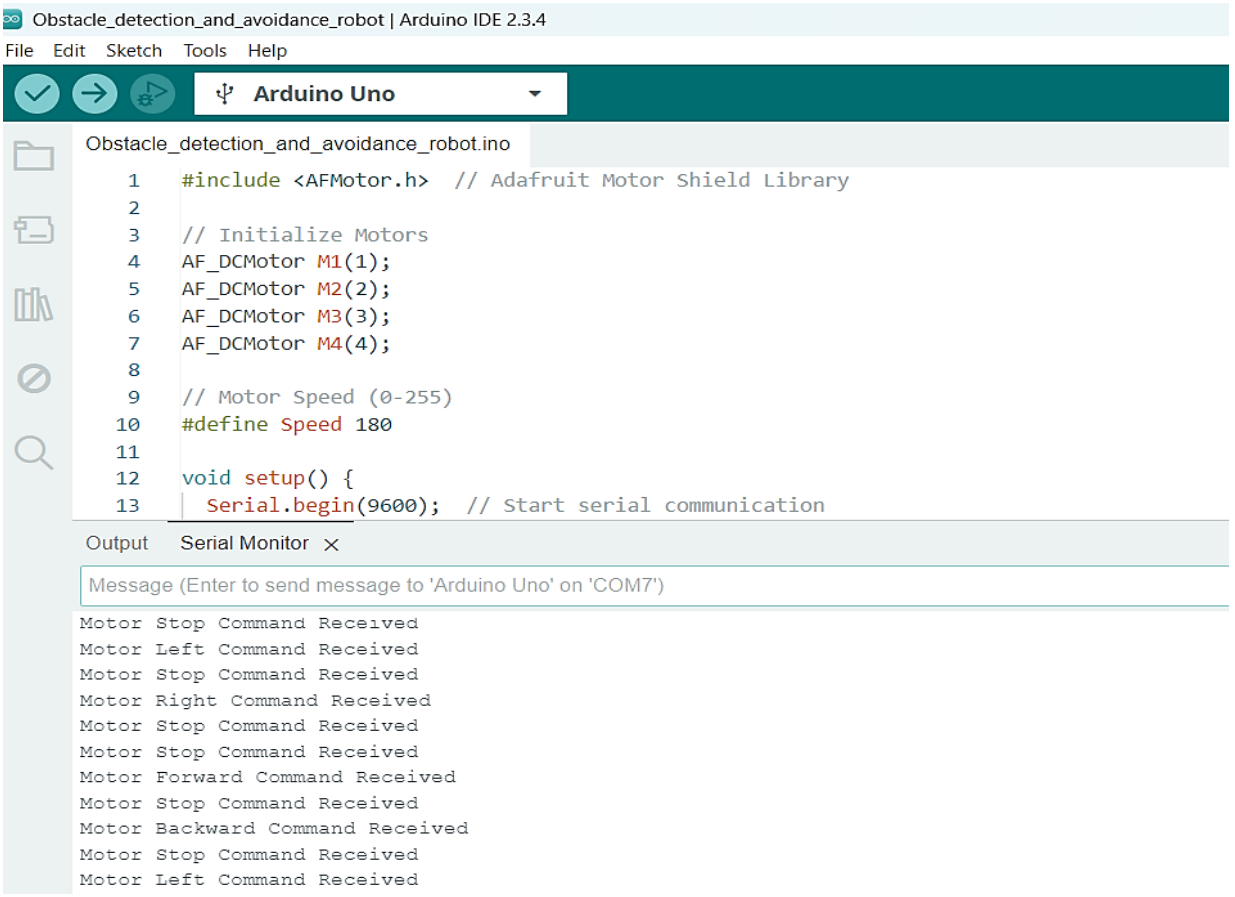


Fig.8.1.2 Unit Testing of Motor control movement in Obstacle Detection and Avoidance Robot

Bluetooth control test output:

Display a screenshot of the serial monitor showing control messages from Bluetooth commands.

```
Bluetooth Command: 'F' (Moving Forward)
Bluetooth Command: 'B' (Moving Backward)
Bluetooth Command: 'L' (Turning Left)
Bluetooth Command: 'R' (Turning Right)
Bluetooth Command: 'S' (Stop)
```

8.2 Integration Testing

Objective: Integration testing ensures that different modules work together as expected.

Testing Approach:

- The ultrasonic sensor was integrated with motor control to verify obstacle detection and avoidance logic.
- The Bluetooth module was tested in combination with motor control to check remote navigation functionality.
- The backup recovery mechanism was tested to ensure it properly executes when no clear path is available.

Expected Outcome:

Sensor and motor integration:
Output showing the robot stopping when an obstacle is detected and moving backward, followed by turning left or right based on sensor data.

```
Obstacle detected at 10 cm
Moving Backward
Left Distance: 50 cm
Right Distance: 30 cm
Turning Left
```

Bluetooth + motor control integration:
Show Bluetooth commands being received and the motor executing the movement accordingly.

```
Bluetooth Command: 'R' (Turning Right)
Motor Turning Right
```

8.3 System Testing

Objective: System testing verifies the complete functionality of the robot in real-world conditions, ensuring it performs as expected when fully assembled.

Testing Approach:

- The robot was tested in various obstacle courses with different object placements.
- The robot’s responses to sudden obstacles were evaluated.

Expected Outcome:

Fig. 8.3.1 System Testing of Obstacle Detection and Avoidance Robot

8.4 Alpha Testing

Objective: Alpha testing is performed in a controlled environment by the development team to identify major issues before external testing.

Testing Approach:

- The robot was tested in a controlled lab environment.
- Multiple test cases were created to check different movement scenarios.

Expected Outcome:

Figure 8.4.1: Team Collaboration in Robot Assembly and Testing

8.5 Beta Testing

Objective: Beta testing is conducted by external users or testers in real-world environments to gather feedback.

Testing Approach:

- The robot was tested in homes, offices, and open spaces.
- Users provided feedback on its navigation accuracy and responsiveness.
- Adjustments were made based on user feedback.

Expected Outcome:

User feedback during beta testing:

If beta testers provided feedback via the serial monitor or through a logging mechanism, include relevant feedback output, such as:

```
Feedback: Robot turned too fast, needs smoother motion.
```

```
Feedback: Obstacle detection worked well, but backup recovery is
slow.
```

8.6 Functional Testing

Objective: Functional testing ensures that the robot meets all design requirements and performs as expected.

Testing Approach:

- The robot was tested for accurate obstacle detection and proper avoidance.
- The emergency stop function was tested for immediate halting when a critical obstacle is detected.
- The voice and Bluetooth control features were tested for responsiveness.

Expected Outcome:

Functional test results:

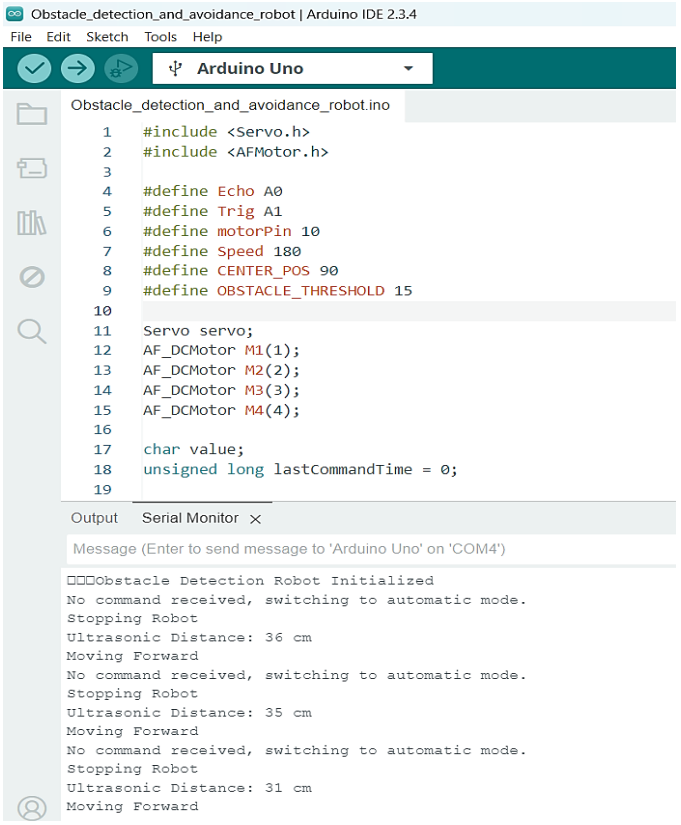


Figure 8.6.1: serial monitor output of Functional testing

8.7 Performance and Stress Testing

Objective: This type of testing evaluates the robot's durability, response time, and efficiency under extreme conditions.

Testing Approach:

- The robot was tested on various surfaces, including smooth floors, carpets, and rough terrain.
- The reaction time from obstacle detection to movement execution was measured.
- The robot was run continuously for several hours to check long-term reliability.

Expected Outcome:

Figure 8.7.1: Robot Prototype Testing with Arduino

The robot is connected to a computer for code execution and debugging. The Arduino board and ultrasonic sensor are visible, indicating the robot's obstacle detection functionality being tested.

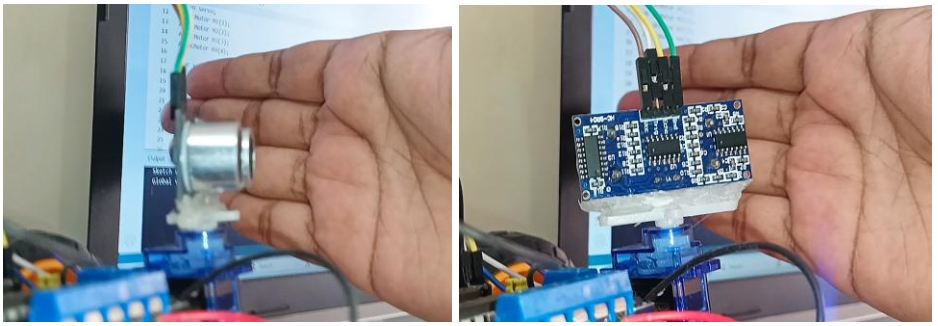


Figure 8.7.2 & 3: Sensor Calibration and Testing

The onboard sensor module is being verified for accurate obstacle detection and navigation.This step is crucial to ensure the robot can identify obstacles and react accordingly in real-time.

CHAPTER 9

CONCLUSION

9. CONCLUSION

The **Obstacle Detection and Avoidance Robot** was successfully designed and implemented to autonomously navigate its environment while avoiding obstacles. The system integrates ultrasonic sensors, servo motors, and an Arduino-based control mechanism to ensure real-time obstacle detection and efficient navigation. The implementation of adaptive turning, emergency stop, and backup recovery strategies enhanced the robot's decision-making capabilities, allowing it to operate in dynamic and unpredictable environments.

The **primary objectives** of this project were to develop an autonomous mobile robot that can:

1. Detect obstacles in real-time using an ultrasonic sensor.
2. Decide the best path to move forward, left, or right based on sensor data.
3. Stop immediately if a critical obstacle is detected.
4. Provide an alternative manual control option via Bluetooth and voice commands.

Through rigorous **testing and validation**, the system demonstrated **accurate obstacle detection, effective path correction, and seamless Bluetooth control**. The integration of dynamic backward movement and randomized escape strategies ensured that the robot could handle challenging scenarios, such as narrow pathways and congested environments.

The project also highlighted the **importance of real-time decision-making and adaptive algorithms** in robotics. The combination of **hardware and software** proved effective in creating a functional and autonomous robot capable of intelligent movement.

Overall, this project serves as a **strong foundation for further advancements in mobile robotics**. With additional enhancements, the system can be expanded to support more complex navigation strategies, multi-sensor integration, and AI-based decision-making, making it suitable for real-world applications such as autonomous delivery systems, smart surveillance, and industrial automation.

CHAPTER 10

FUTURE ENHANCEMENT

10. FUTURE ENHANCEMENT

Although the **Obstacle Detection and Avoidance Robot** is capable of autonomous navigation and real-time obstacle detection, several **enhancements** can be incorporated to improve its efficiency, intelligence, and usability.

Several enhancements can be incorporated to improve the **Obstacle Detection and Avoidance Robot**:

- **AI-Based Object Recognition:** Using machine learning, the robot can classify obstacles and navigate more efficiently.
- **Multi-Sensor Integration:** Adding infrared, LIDAR, and IMU sensors will improve accuracy in obstacle detection.
- **GPS-Based Navigation:** Implementing GPS will enable the robot to follow predefined paths for outdoor use.
- **IoT Connectivity:** Remote control and monitoring through a mobile app or web interface can enhance usability.
- **Energy Optimization:** Using solar panels and efficient power management can extend battery life.
- **Gesture and Voice Control:** Enhancing user interaction through hand gestures and advanced voice commands.

CHAPTER 11

REFERENCES

11. REFERENCES

1. **Arduino Official Documentation** – Arduino Ultrasonic Sensor Guide. Available at: https://www.arduino.cc
2. **HC-SR04 Ultrasonic Sensor** – Datasheet and Working Principle. Available at: https://www.sparkfun.com
3. **Motor Control using Arduino** – DC Motor Interfacing. Available at: https://www.robotshop.com
4. **Path Planning Algorithms** – Introduction to A* and Dijkstra's Algorithm. Available at: https://towardsdatascience.com
5. **Machine Learning in Robotics** – Object Detection and Recognition in Autonomous Navigation. Research Paper: IEEE Xplore.
6. **Internet of Things in Robotics** – Applications of IoT in Autonomous Vehicles. Available at: https://www.sciencedirect.com
7. **Battery Optimization Techniques** – Efficient Power Management in Robotics. Available at: https://www.mdpi.com
8. **Bluetooth Communication in Embedded Systems** – How Bluetooth HC-05 Works with Arduino. Available at: https://randomnerdtutorials.com
9. **Gesture Control in Robotics** – Implementing Gesture-Based Navigation Using Sensors. Available at: https://ieeexplore.ieee.org