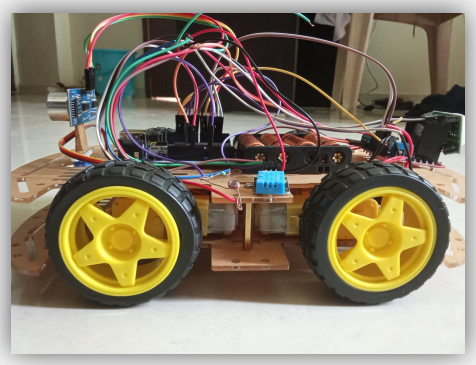
VISHWAKARMA UNIVERSITY
Maximising Human Potential

Name: Panashe kunaka
Roll number :100
Division B

Micro-Controller and Micro-Processor Lab

Project Report –2024_25

★ **Title**



MOBILE WAREHOUSE SENSOR

★ **Aim**

• To design and construct a mobile robot that can autonomously navigate a warehouse environment while avoiding obstacles.
• To implement sensor-based detection (ultrasonic, DHT11, LDR) for collision avoidance, environmental monitoring, and alert generation.
• To integrate a servo for scanning and decision-making based on sensor data.
• To demonstrate efficient power management using separate supplies and voltage regulation.

★ Apparatus (Hardware Requirements)

- **Arduino UNO**
- **L298N Motor Driver Module**
  (with ENA/ENB and 4 direction control pins)
- **4 DC Motors**
  (Left and right motors, connected in parallel as per design)
- **SG90 Servo Motor**
  (For ultrasonic sensor scanning)
- **HC-SR04 Ultrasonic Sensor**
- **DHT11 Temperature & Humidity Sensor**
- **LDR (Light Dependent Resistor)**
  (with 10kΩ resistor for voltage divider)
- **Buzzer**
- **Power Supplies:**
  - **9.6V battery for motor driver (via L298N 12V input)**
- **Connecting wires,switch, and Battery Holders(handmade)**

★ Theory

- **• Obstacle Detection: The HC-SR04 ultrasonic sensor sends out a sound pulse and calculates the time it takes to receive the echo, determining the distance to obstacles. The servo rotates the sensor to scan different directions.**

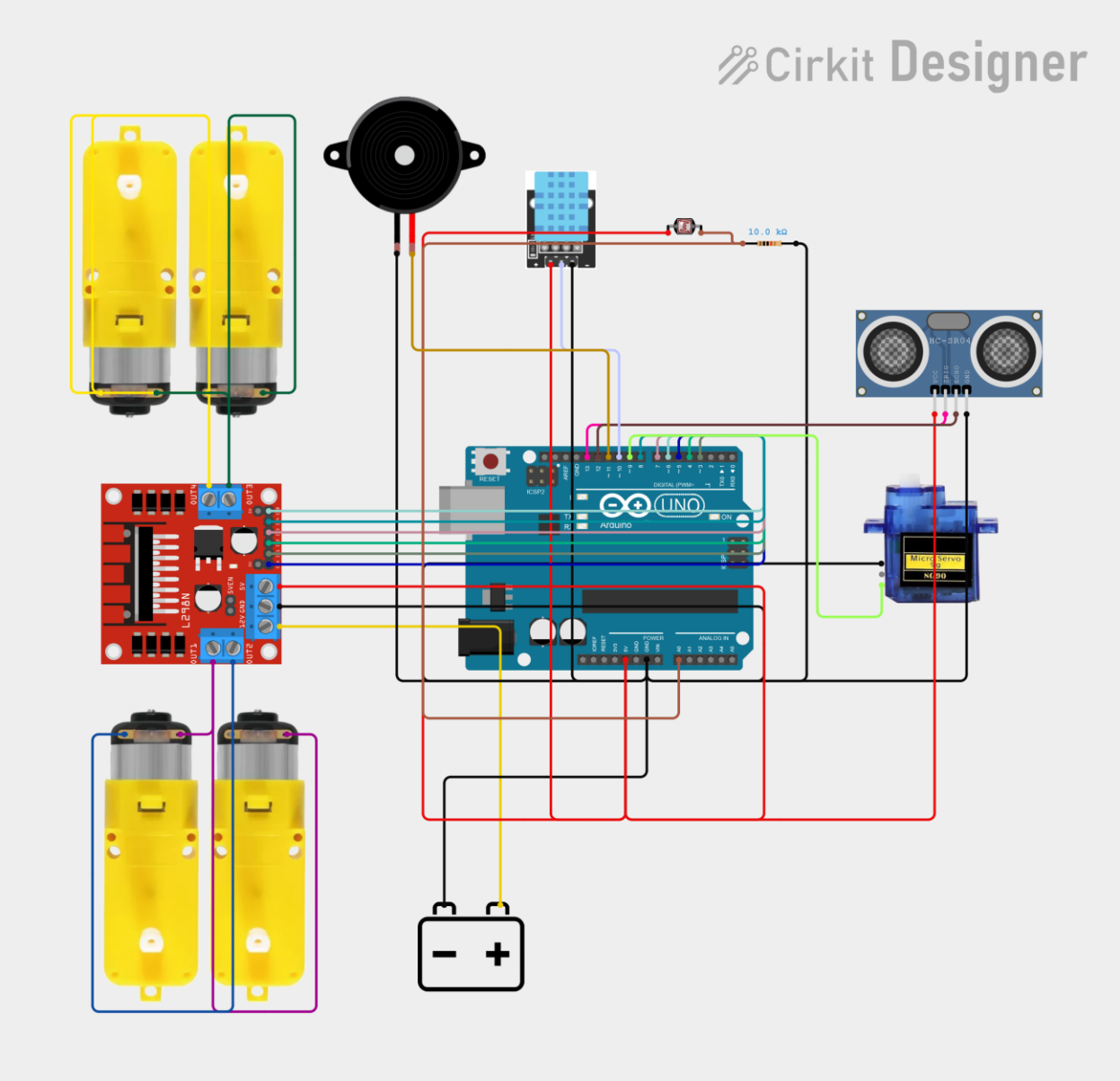
  • Environmental Sensing: The DHT11 sensor measures temperature, and if values exceed preset thresholds, it triggers an alert.

  • Light Detection: An LDR, combined with a resistor in a voltage divider configuration, outputs an analog value reflecting ambient light levels. If the light level falls below a threshold, an alert is triggered.

  • Motor Control: The L298N motor driver controls the direction and speed (via PWM on the enable pins) of the motors, allowing the robot to move forward, turn, or reverse.

  • Power Management: The system uses separate power sources for the motor driver (9V battery) and the servo (6V battery), with common grounds to ensure stable operation.
-

★ Circuit Diagrams



★ Source Code (with Comments)

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

// Motor Driver Connections
#define MOTOR_LEFT_IN1 3
#define MOTOR_LEFT_IN2 4
#define MOTOR_RIGHT_IN1 7
#define MOTOR_RIGHT_IN2 8
```

```
#define ENA 5
#define ENB 6

// Sensor Pins
#define TRIG_PIN 13
#define ECHO_PIN 12
#define DHT_PIN 10
#define LDR_PIN A0
#define BUZZER_PIN 11
#define SERVO_PIN 9

// Constants
#define OBSTACLE_DISTANCE 40.0   // cm
#define SCAN_DELAY 250           // ms
#define BASE_SPEED 200           // PWM (0-255)
#define TURN_TIME 650            // ms for 90-degree turn
#define TEMP_THRESHOLD 45.0      // °C
#define LIGHT_THRESHOLD 200      // LDR value for darkness
#define DHT_TYPE DHT11

Servo steeringServo;
DHT dht(DHT_PIN, DHT_TYPE);

unsigned long lastSensorCheck = 0;
bool lastLightState = true;
bool lastTempState = true;

void setup() {
  Serial.begin(9600);
  pinMode(MOTOR_LEFT_IN1, OUTPUT);
  pinMode(MOTOR_LEFT_IN2, OUTPUT);
  pinMode(MOTOR_RIGHT_IN1, OUTPUT);
  pinMode(MOTOR_RIGHT_IN2, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  analogWrite(ENA, BASE_SPEED);
  analogWrite(ENB, BASE_SPEED);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LDR_PIN, INPUT);
  dht.begin();
  steeringServo.attach(SERVO_PIN);
  steeringServo.write(90);
```

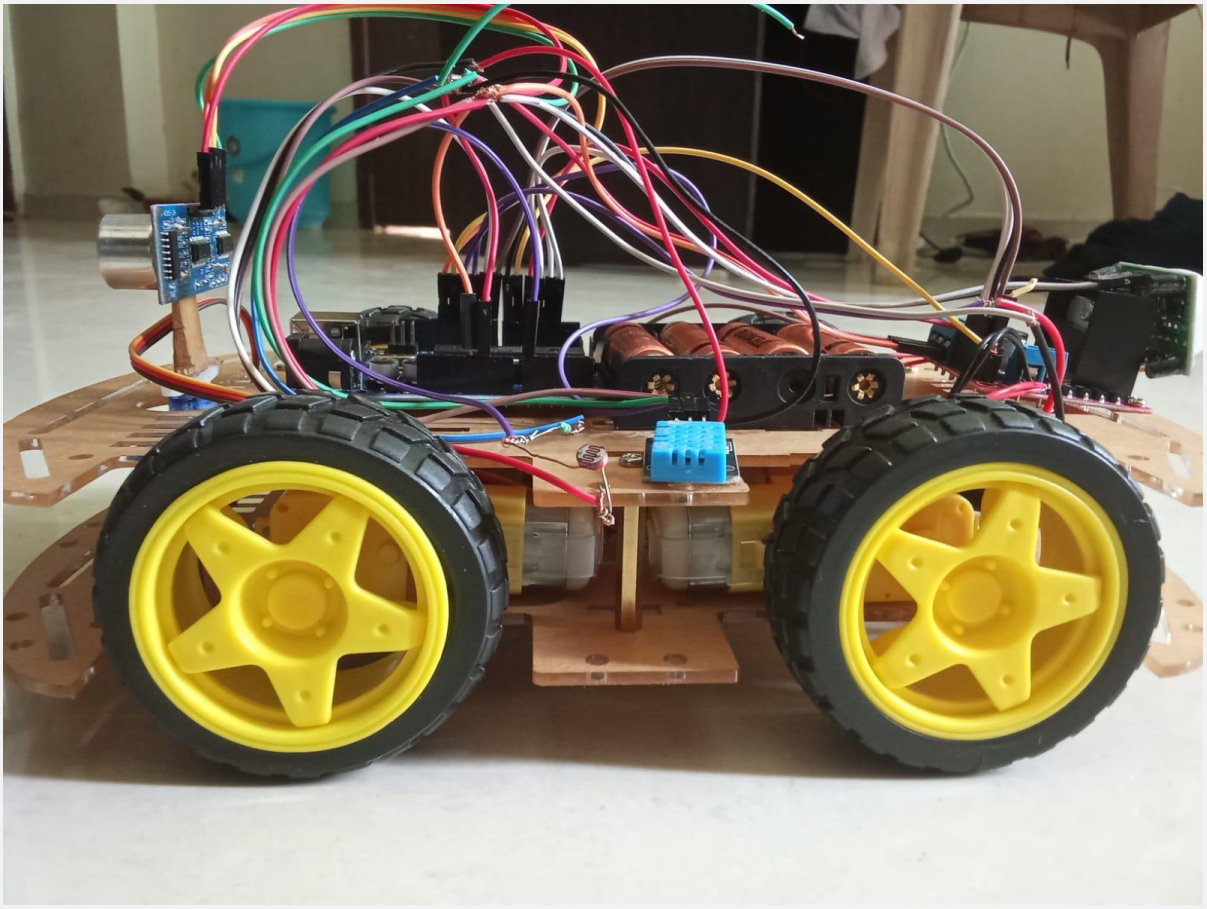
```
  delay(500);
}

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - lastSensorCheck >= SCAN_DELAY) {
    lastSensorCheck = currentMillis;
    float temperature = 30 + dht.readTemperature();
    int lightLevel = analogRead(LDR_PIN);
    if (!isnan(temperature)) {
      if (temperature > TEMP_THRESHOLD && lastTempState) {
        beepThrice();
        lastTempState = false;
      } else if (temperature <= TEMP_THRESHOLD) {
        lastTempState = true;
      }
    }
    if (lightLevel < LIGHT_THRESHOLD && lastLightState) {
      beepTwice();
      lastLightState = false;
    } else if (lightLevel >= LIGHT_THRESHOLD) {
      lastLightState = true;
    }
    float distance = measureDistance();
    Serial.print("Front: "); Serial.print(distance); Serial.print(" cm | Temp: ");
    Serial.print(temperature); Serial.print("°C | Light: "); Serial.println(lightLevel);
    if (distance > OBSTACLE_DISTANCE) {
      moveForward();
    } else {
      avoidObstacle();
    }
  }
}

void avoidObstacle() {
  emergencyStop();
  beepOnce();
  delay(500);
  float leftDist = scanDirection(0);
  float rightDist = scanDirection(180);
  steeringServo.write(90);
  delay(200);
  Serial.print("Obstacle! L: "); Serial.print(leftDist);
  Serial.print("cm R: "); Serial.println(rightDist);
  if (leftDist > rightDist && leftDist > OBSTACLE_DISTANCE) {
```

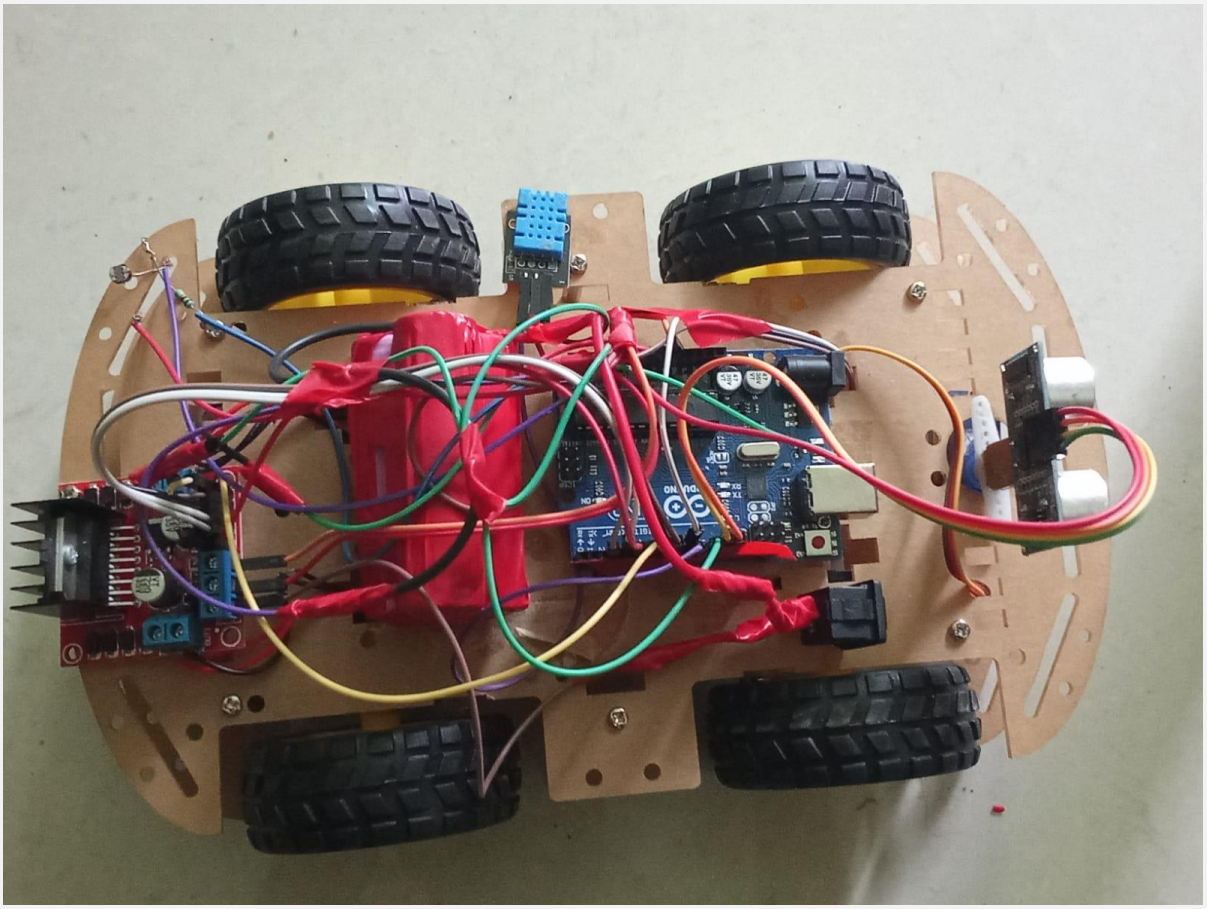
```
    turnLeft90();
  } else if (rightDist > OBSTACLE_DISTANCE) {
    turnRight90();
  } else {
    moveBackward();
    delay(800);
    emergencyStop();
    delay(300);
  }
}

void beepOnce() {
  tone(BUZZER_PIN, 1000, 200);
  delay(300);
}

void beepTwice() {
  for(int i=0; i<2; i++) {
    tone(BUZZER_PIN, 1000, 100);
    delay(150);
  }
  delay(300);
}

void beepThrice() {
  for(int i=0; i<3; i++) {
    tone(BUZZER_PIN, 1000, 100);
    delay(150);
  }
  delay(300);
}

float measureDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(4);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH, 30000);
  return duration * 0.0343 / 2;
}

float scanDirection(int angle) {
  steeringServo.write(angle);
```

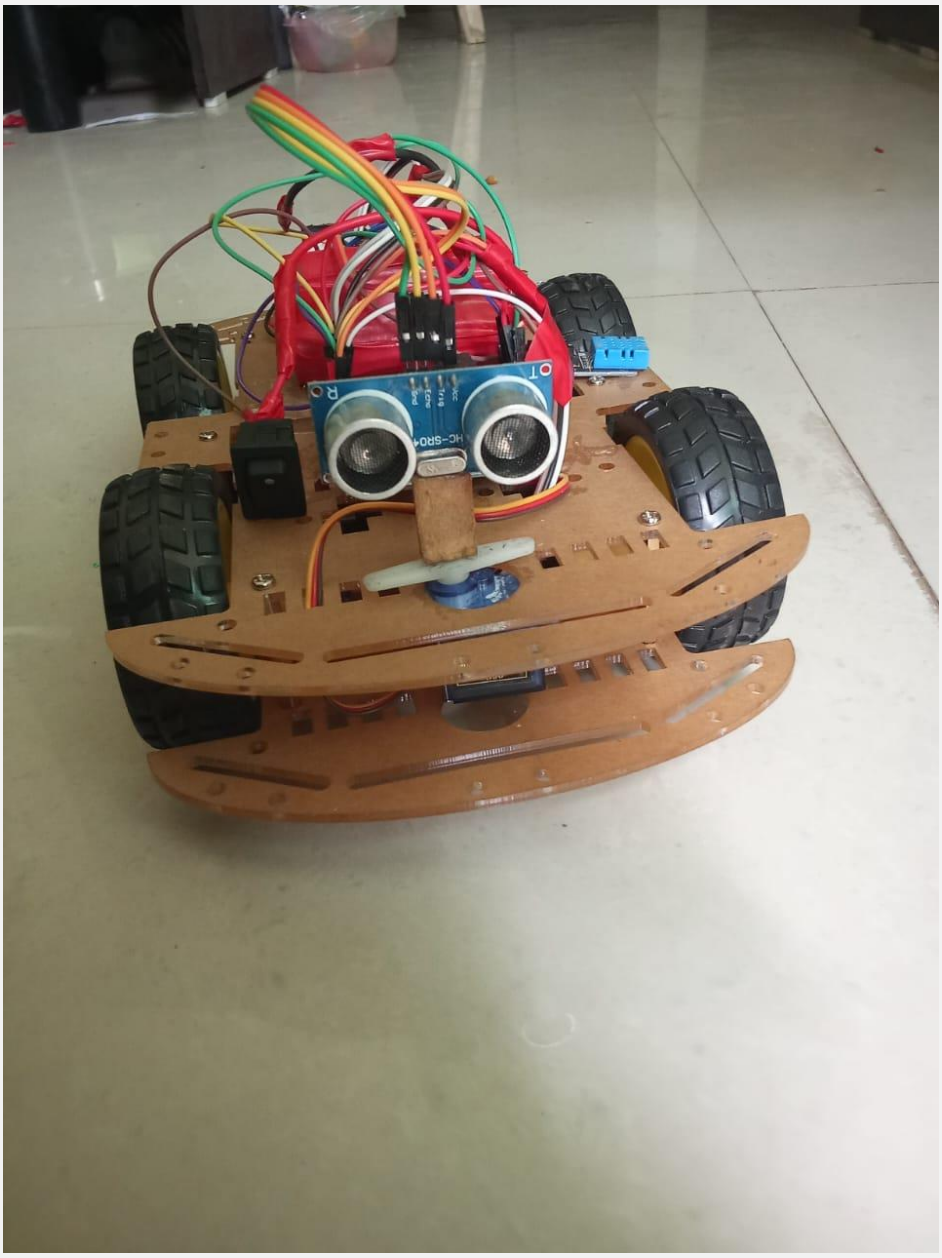
```
  delay(300);
  return measureDistance();
}

void moveForward() {
  digitalWrite(MOTOR_LEFT_IN1, HIGH);
  digitalWrite(MOTOR_LEFT_IN2, LOW);
  digitalWrite(MOTOR_RIGHT_IN1, HIGH);
  digitalWrite(MOTOR_RIGHT_IN2, LOW);
}

void moveBackward() {
  digitalWrite(MOTOR_LEFT_IN1, LOW);
  digitalWrite(MOTOR_LEFT_IN2, HIGH);
  digitalWrite(MOTOR_RIGHT_IN1, LOW);
  digitalWrite(MOTOR_RIGHT_IN2, HIGH);
}

void turnLeft90() {
  digitalWrite(MOTOR_LEFT_IN1, LOW);
  digitalWrite(MOTOR_LEFT_IN2, HIGH);
  digitalWrite(MOTOR_RIGHT_IN1, HIGH);
  digitalWrite(MOTOR_RIGHT_IN2, LOW);
  delay(TURN_TIME);
  emergencyStop();
}

void turnRight90() {
  digitalWrite(MOTOR_LEFT_IN1, HIGH);
  digitalWrite(MOTOR_LEFT_IN2, LOW);
  digitalWrite(MOTOR_RIGHT_IN1, LOW);
  digitalWrite(MOTOR_RIGHT_IN2, HIGH);
  delay(TURN_TIME);
  emergencyStop();
}

void emergencyStop() {
  digitalWrite(MOTOR_LEFT_IN1, LOW);
  digitalWrite(MOTOR_LEFT_IN2, LOW);
  digitalWrite(MOTOR_RIGHT_IN1, LOW);
  digitalWrite(MOTOR_RIGHT_IN2, LOW);
}
```

★ Observations / (Photos, Posters, Video Links)

- **Photos of the Constructed Model:**







★ Applications

• Warehouse Monitoring: The robot autonomously navigates and detects anomalies in a warehouse setting.
• Security and Surveillance: Serves as a prototype for automated security systems in industrial environments.
• Educational Tool: Demonstrates integration of sensors, actuators, and control algorithms.

★ Conclusion

- **Summary:**
  The project successfully integrates multiple sensors and actuators with an Arduino UNO to create an obstacle-avoiding robot for warehouse monitoring.
- **Learning Outcome:**
  Emphasized modular design, sensor integration, and efficient power management.
- **Future Work:**
  Future improvements could include wireless control, additional sensors (like gas detectors), and more advanced navigation algorithms.