Hamm-Lippstadt University of Applied Sciences

Prototyping & Systems Engineering Presentation

Group 7

Table of contents

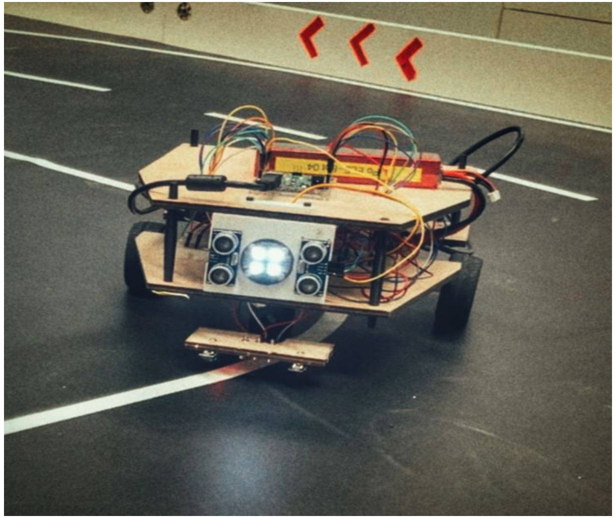
Introduction

Team Members:

- Md Sayem
- Sayed Galib Hossen Rizvi
- Ronjon Sarker
- Raphael C. G. Catchpole
- Dapsara Kapuge

Project Overview

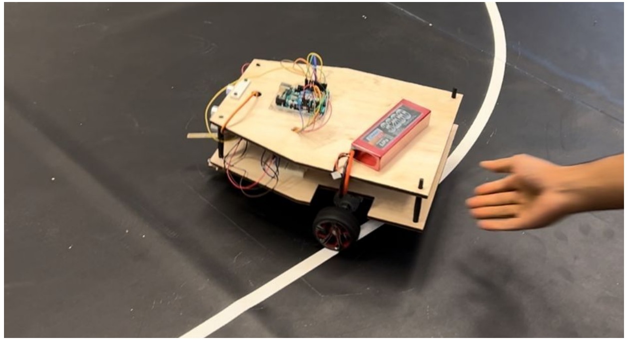
- Developing an autonomous car using Arduino.
- Integrates ultrasonic sensors for obstacle detection
- Infrared sensors for line following
- Uses color sensor for for color detection
- DC motors drive the car.
- Servo motor for obstacle removal



Goals : Demonstrate advanced navigation, obstacle avoidance and removal using servo and color sensor, and adaptive behavior

Mission Statement

"Develop an Arduino-based autonomous car with advanced sensors for intelligent navigation, obstacle avoidance and removal, and environmental interaction, showcasing innovation in DIY robotics."



Project Requirements

Functional Requirements

1. **Path Tracing**: The vehicle should be able to trace a predefined path.
2. **Sharp Turns**: The vehicle must be capable of executing sharp turns (e.g., 90° turns).
3. **Circuit Navigation**: The vehicle should be able to navigate various circuit patterns (e.g., circular or oval).
4. **Speed Adjustment**: The vehicle must be able to adjust its speed efficiently.
5. **Obstacle Detection**: The vehicle should be able to identify and avoid obstacles.
6. **Color Recognition**: The vehicle must be able to recognize different colors.
7. **Ignoring Maneuver**: The vehicle should be capable of ignoring an osbstacle and get back to line again
8. **Obstacle Bypass**: The vehicle must be able to bypass obstructions.
9. **Parking**: The vehicle should be able to successfully park in designated spots.

Non-Functional Requirements

1. **Reliability**: The vehicle should operate reliably under various conditions.
2. **Efficiency**: The vehicle must optimize power usage to ensure long operation times.
3. **Durability**: The vehicle components should be durable enough to withstand regular use.
4. **Scalability**: The vehicle design should allow for easy upgrades and enhancements.
5. **Ease of Maintenance**: The vehicle should be easy to maintain and repair.

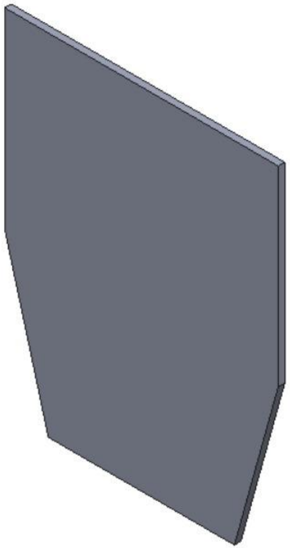
Success Criteria

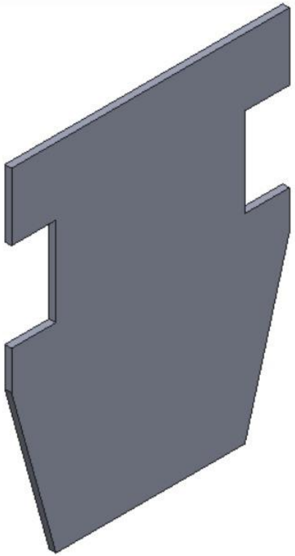
1. **Accuracy**: The vehicle successfully traces the predefined path with minimal deviation.
2. **Performance in Turns**: The vehicle executes sharp turns smoothly and accurately.
3. **Circuit Completion**: The vehicle navigates various circuit patterns without issues.
4. **Speed Optimization**: The vehicle adjusts speed efficiently based on conditions.
5. **Obstacle Handling**: The vehicle detects and avoids obstacles effectively.
6. **Color Detection**: The vehicle accurately recognizes and responds to different colors.
7. **Turnaround Execution**: The vehicle performs a 180-degree turn without errors.
8. **Obstacle Bypass Success**: The vehicle bypasses obstructions without collision.
9. **Parking Accuracy**: The vehicle parks accurately in designated spots.

Design

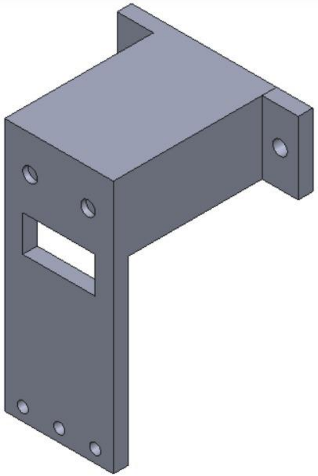
CAD Models

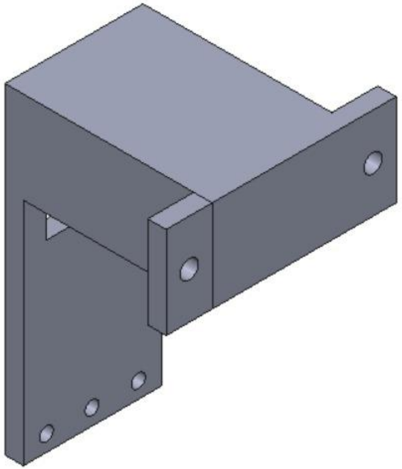
Laser Cut Parts

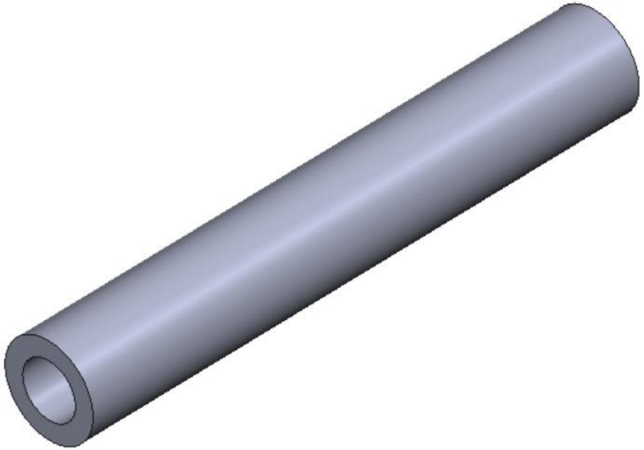




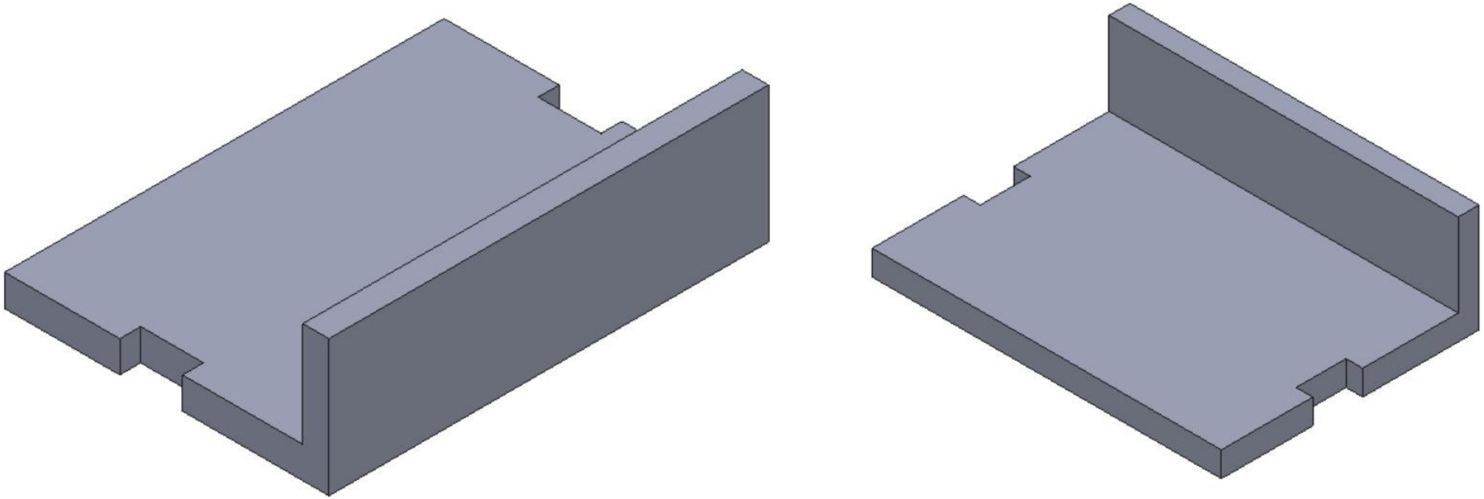
3D Printed Parts



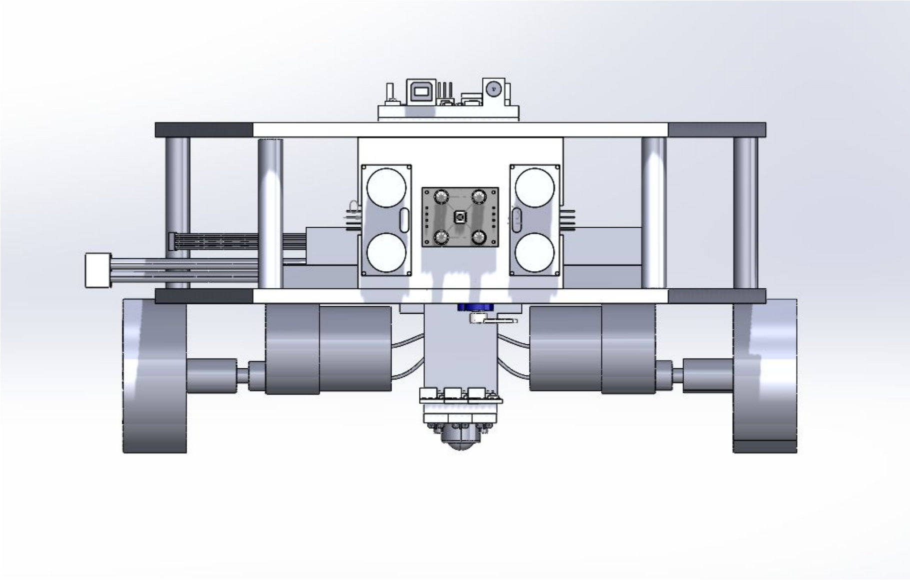


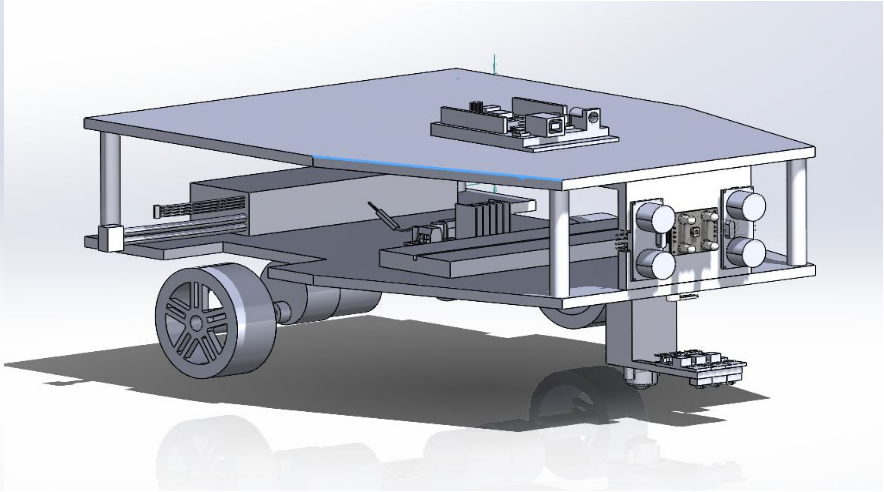


3D Printed Parts

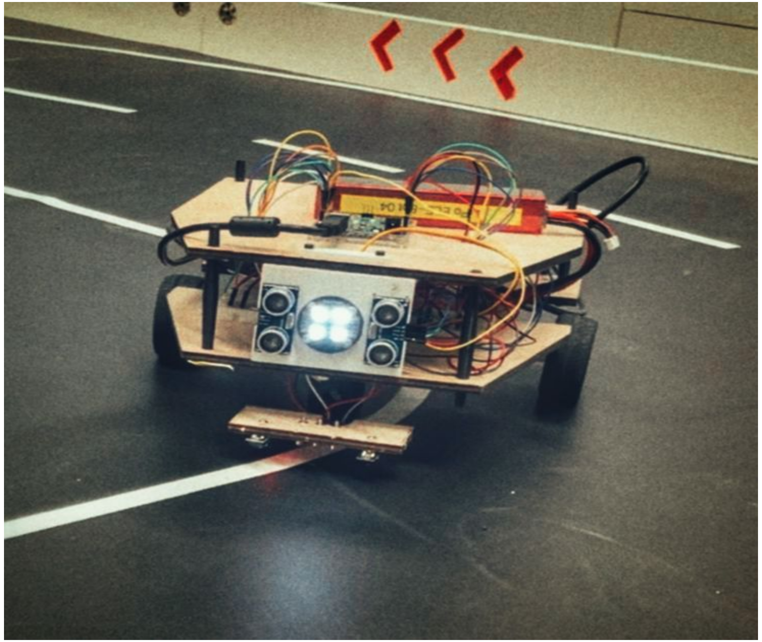


Finished Assembly





Final Product



Hardware

Components

Microcontroller:

Arduino Uno

Sensors:

ST 1140 Line Tracking IR Sensors (2 units)

HC SR-04 Ultrasonic Sensor

TCS3200 Color Sensor

Actuators:

DC Motors (2 units)

Servo Motor

Motor Driver:

L298 Motor Driver

Power Supply:

Battery Pack

Miscellaneous:

Various connecting wires

Various screws

Breadboard

Hardware Integration

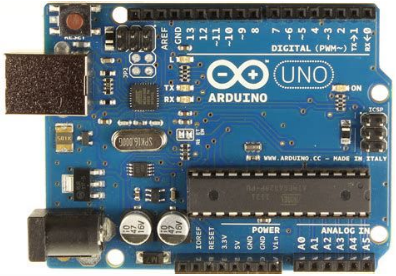
Hardware Integration

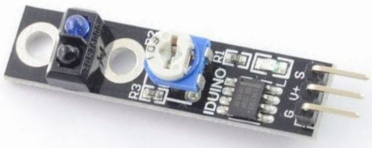
Central Controller: Arduino Uno

- Coordinates sensors and actuators

Sensors:

- **IR Sensors (ST 1140):**
    - Detect the line
    - Connected to digital input pins
- **Ultrasonic Sensor (HC-SR04):**
    - Measures distance to obstacles
    - Uses trigger and echo system
    - Connected to digital pins
- **Color Sensor (TCS3200):**
    - Detects color of objects
    - Connected to digital and analog pins





Hardware Integration

Actuators:

- **DC Motors:**
  - Controlled by L298 motor driver
  - Connected to PWM pins
- **Servo Motor:**
  - Executes color-based actions
  - Connected to digital pin

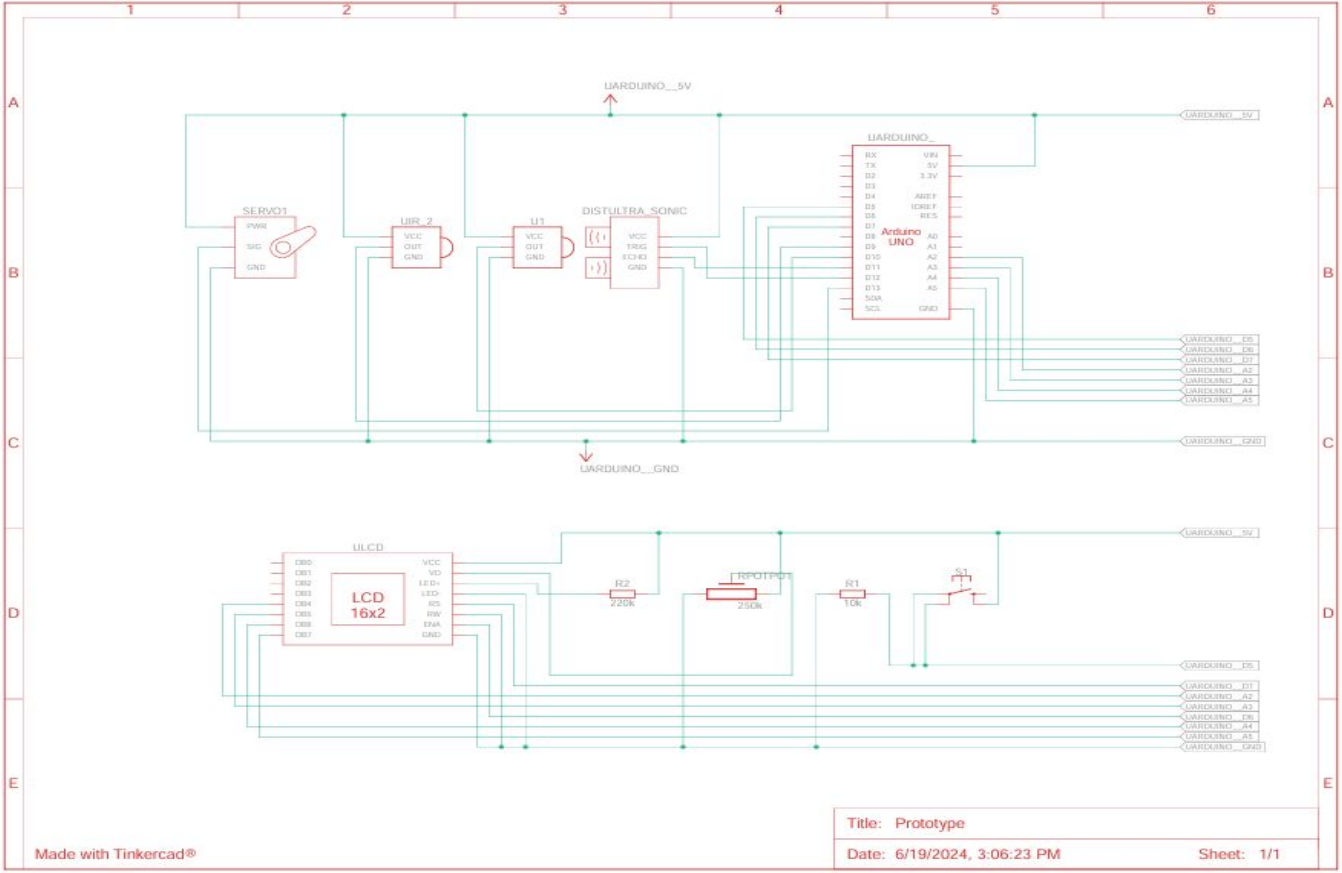
Power Supply:

- Battery pack powers the system

Integration Process:

- Test each component individually
- Program Arduino to read sensors and control actuators
- IR sensors adjust motor speeds for line-following
- Ultrasonic sensor readings trigger color sensor actions

Schematics and Wiring Diagrams



UARDUINO__5V
SERVO1
UIR_2
U1
DISTULTRA_SONIC
Arduino UNO
UARDUINO__GND
ULCD
LCD 16x2
R2
220k
250k
R1
10k
S1
Made with Tinkercad®
Title: Prototype
Date: 6/19/2024, 3:06:23 PM
Sheet: 1/1

Software

Software Architecture

Our software architecture integrates hardware components with modular code for efficient operation.

Main Control Loop

Description: Continuously reads sensor data, processes it, and controls actuators.

Functions:

Reads IR, ultrasonic, and color sensors.

Processes data for real-time decisions.

Controls motor driver and servo motor.

Sensor Modules

IR Sensors: Detect lines, using digitalRead for binary signals.

Ultrasonic Sensor: Measures distance, using pulseIn for echo time.

Color Sensor: Detects colors, measuring pulse widths for red, green, and blue.

Software and Architecture

Actuator Modules

Motor Driver (L298): Controls motor speed and direction using analogWrite and digitalWrite.

Servo Motor: Executes actions based on color detection, controlled by the Servo library.

Decision-Making Logic

Line Following: Moves forward, turns left, or turns right based on IR sensor inputs.

Obstacle Avoidance: Stops and checks color sensor when an obstacle is detected, performs actions based on color.

Communication Interfaces

Serial Communication: Used for debugging and monitoring, implements Serial.begin and Serial.print.

Key Algorithms and Code

Line Following Algorithm

Purpose: Keep the robot on the line using IR sensors.

Logic:

Both sensors detect line: Move forward.

Left sensor off line: Turn right.

Right sensor off line: Turn left.

```
void loop() {
  distance_F = Ultrasonic_read();
  Serial.print("D F="); Serial.println(distance_F);

  int rightSensor = digitalRead(R_S);
  int leftSensor = digitalRead(L_S);
  Serial.print("Right Sensor: "); Serial.println(rightSensor);
  Serial.print("Left Sensor: "); Serial.println(leftSensor);

  if (rightSensor == 1 && leftSensor == 1) {
    if (distance_F < Set) {
      Stop();
       delay(1000);
      int color = getColor();
      if (color == 1) {
        IgnoreObstacle();
      } else if (color == 2) {
        servo.write(0);
        delay(1000);
        servo.write(180);
        delay(1000);

      } else if (color == 3) {
        ParkCar();
      }
    } else {
      forward();
    }
  } else if (rightSensor == 1 && leftSensor == 0) {

       turnLeft();


  } else if (rightSensor == 0 && leftSensor == 1) {

     turnRight();



  }
```

Key Algorithm and code

Obstacle Detection and Avoidance

Purpose: Detect obstacles with ultrasonic sensor, act based on color.

Logic:

Measure distance. If obstacle detected, stop and check color sensor.

Red: Avoid obstacle. Green: Park. Blue: Move servo.

```
void loop() {
  distance_F = Ultrasonic_read();
  Serial.print("D F=");
Serial.println(distance_F);

  int rightSensor =
digitalRead(R_S);
  int leftSensor =
digitalRead(L_S);
  Serial.print("Right Sensor: ");
Serial.println(rightSensor);
  Serial.print("Left Sensor: ");
Serial.println(leftSensor);
```

Key Algorithm and code

```
if (rightSensor == 0 && leftSensor == 0) {
  if (distance_F < Set) {
    Stop();
    int color = getColor();
    if (color == 1) {
      IgnoreObstacle();
    } else if (color == 2) {
      servo.write(0);
      delay(1000);
      servo.write(90);
      delay(1000);
    } else if (color == 3) {
      ParkCar();
    }
  } else {
    forward();
  }
} else if (rightSensor == 1 && leftSensor == 0) {
  turnRight();
} else if (rightSensor == 0 && leftSensor == 1) {
  turnLeft();
}
}
```

Key Algorithm and code

Ultrasonic Sensor Reading

Purpose: Measure distance to obstacle.

Logic: Send pulse, measure echo time, calculate distance.

```
long Ultrasonic_read() {
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  long time = pulseIn(echo, HIGH);
  return time / 29 / 2;
```

Key Algorithm and code

Color Detection

Purpose: Detect object color using TCS3200 sensor.

Logic: Measure pulse widths for red, green, blue. Determine color by smallest value.

```
int getColor() {
  int redValue, greenValue, blueValue;

  digitalWrite(S2_PIN, LOW);
  digitalWrite(S3_PIN, LOW);
  redValue = pulseIn(OUT_PIN, LOW);

  digitalWrite(S2_PIN, HIGH);
  digitalWrite(S3_PIN, HIGH);
  greenValue = pulseIn(OUT_PIN, LOW);

  digitalWrite(S2_PIN, LOW);
  digitalWrite(S3_PIN, HIGH);
  blueValue = pulseIn(OUT_PIN, LOW);

  Serial.print("Red: ");
  Serial.print(redValue);
  Serial.print("\tGreen: ");
  Serial.print(greenValue);
  Serial.print("\tBlue: ");
  Serial.println(blueValue);
```

Key Algorithm and code

```
  if (redValue < greenValue && redValue < blueValue)
{
    Serial.println("Color: Red");
    return 1;
  } else if (blueValue < redValue && blueValue <
greenValue) {
    Serial.println("Color: Blue");
    return 2;
  } else if (greenValue < redValue && greenValue <
blueValue) {
    Serial.println("Color: Green");
    return 3;
  } else {
    return 0;
  }
}
```

Key Algorithm and code

Motor Control Functions

Purpose: Control robot movement.

Logic: Use motor driver to set direction and speed.

```
void forward() {
  Serial.println("Moving forward");
  analogWrite(enA, 117);
  analogWrite(enB, 142);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
void aforward() {
  analogWrite(enA, 200);
  analogWrite(enB, 200);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
void backward() {
  analogWrite(enA, 200);
  analogWrite(enB, 200);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
```

Key Algorithm and code

```
void abackward() {
  analogWrite(enA, 200);
  analogWrite(enB, 200);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void turnRight() {
  Serial.println("Turning right");
  analogWrite(enA, 145);
  analogWrite(enB, 150);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void aturnRight() {
  analogWrite(enA, 200);
  analogWrite(enB, 200);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
```

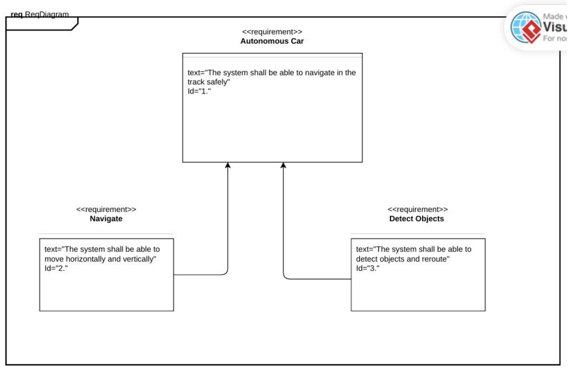
Key Algorithm and code

```
void ParkCar() {
  Serial.println("Park");
  abackward();
  delay(400);
  aturnRight();
  delay(3000);

}
```

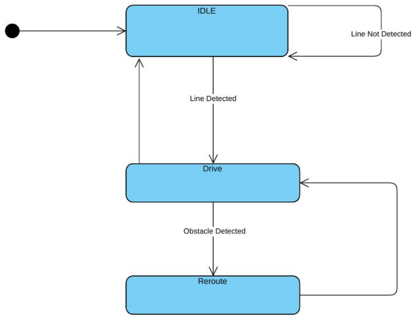
```
void turnLeft() {
  Serial.println("Turning left");
  analogWrite(enA, 145);
  analogWrite(enB, 150);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void aturnLeft() {
  analogWrite(enA, 200);
  analogWrite(enB, 200);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void Stop() {
  Serial.println("Stopping");
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
```

State Charts and UML Diagrams

Requirement Diagram

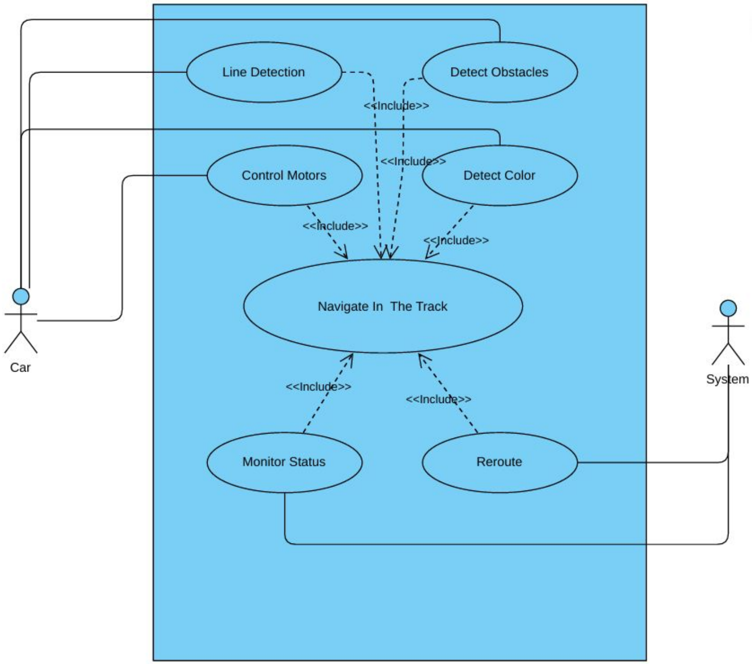


State Machine Diagram



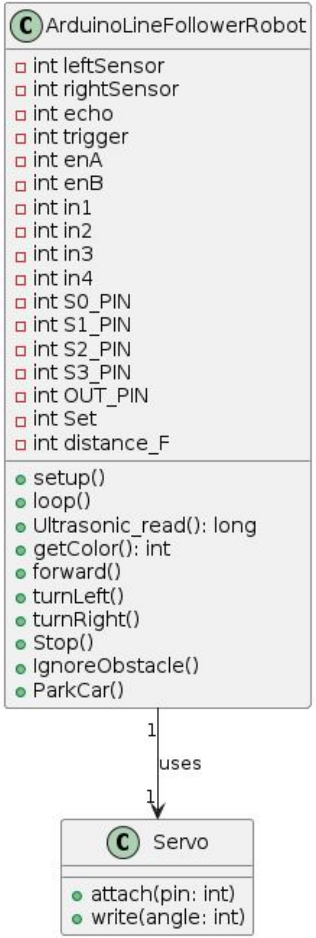
State Charts and UML Diagrams

User Case Diagram

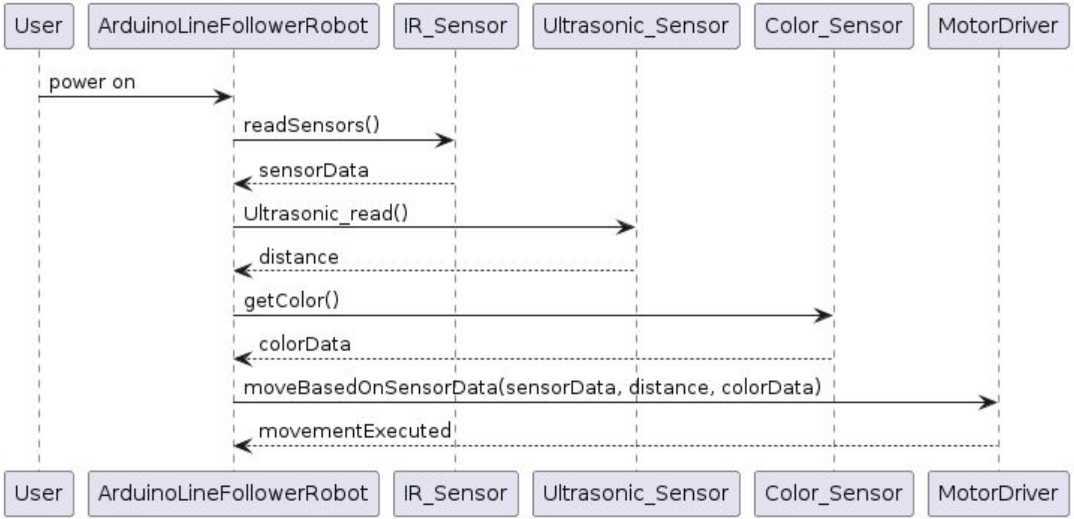


State Charts and UML Diagrams

Class Diagram

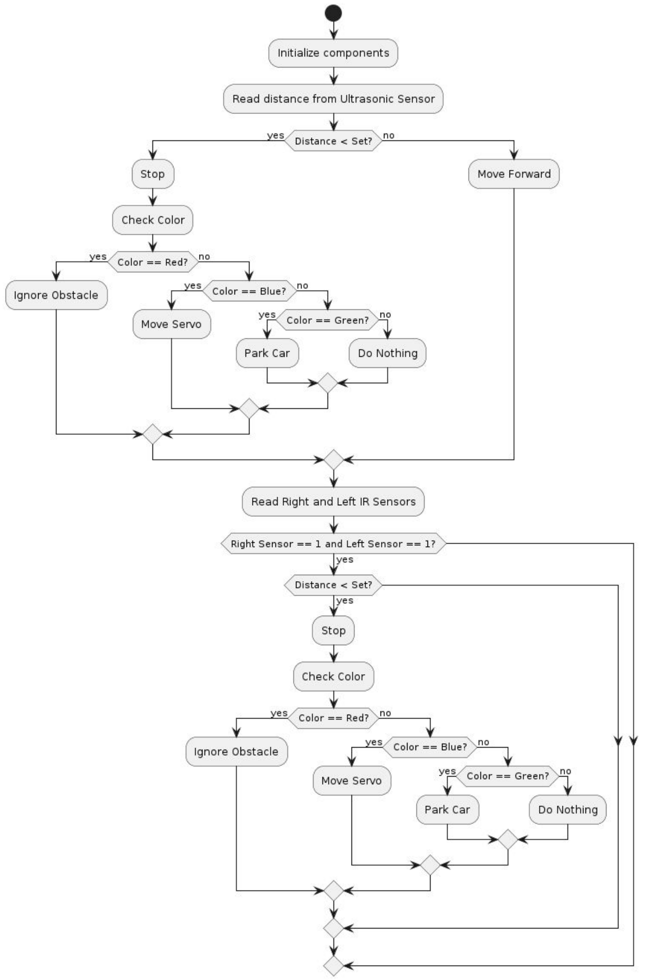


Sequence diagram



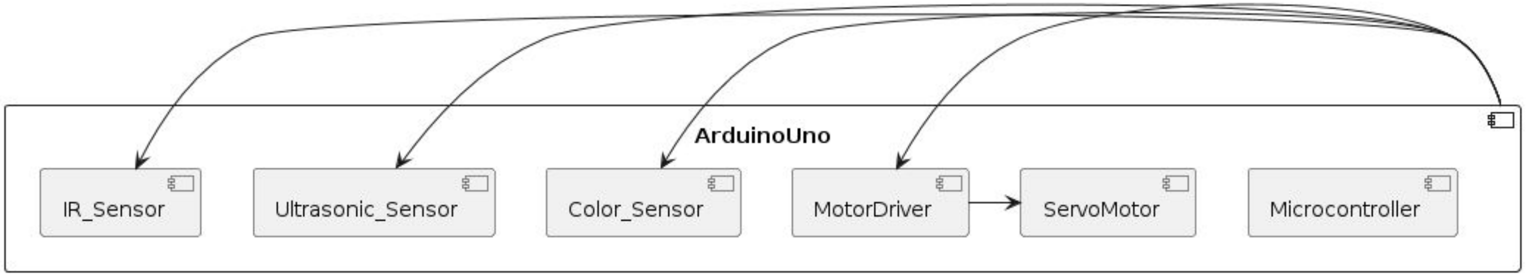
State Charts and UML Diagrams

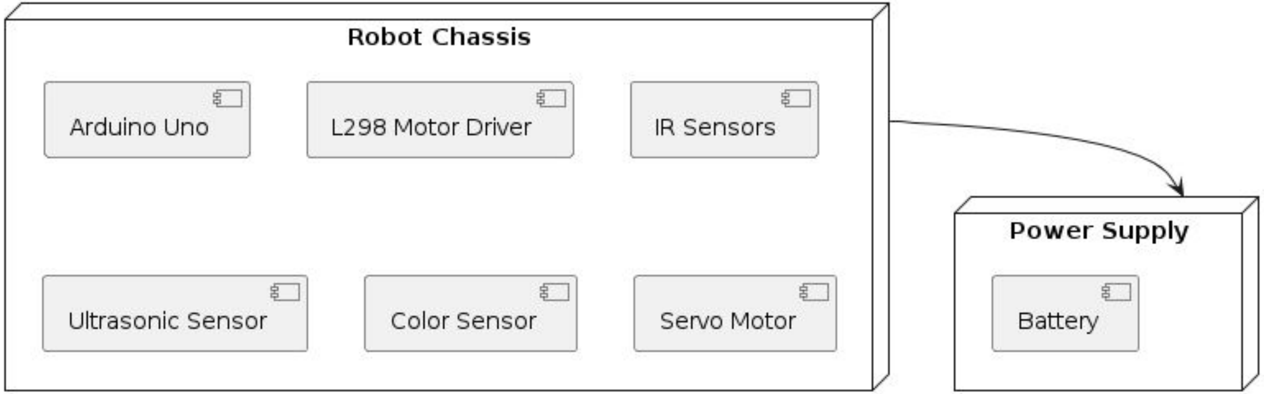
Activity diagram



State Charts and UML Diagrams

Component Diagram





Deployment Diagram

System Integration

Integration of Hardware and Software

Our project integrates hardware and software to create a line-following robot that detects colors and obstacles. This interaction is essential for the robot's functionality.

Hardware Components:

Arduino Uno: Executes code and interfaces with hardware.

Sensors:

IR Sensors: Detect lines.

Ultrasonic Sensor: Measures distances.

Color Sensor: Detects red, green, and blue light.

Actuators:

DC Motors: Controlled by L298 driver for movement.

Servo Motor: Performs actions based on colors.

Software Implementation

Arduino IDE: Writes, compiles, and uploads code.

Sensor Data Processing:

IR Sensors: digitalRead to follow lines.

Ultrasonic Sensor: Ultrasonic_read for obstacle detection.

Color Sensor: getColor for color detection.

Motor Control: analogWrite for speed, digitalWrite for direction.

Decision Making: loop function reads sensor data to control movements.

Integration Process

Setup: Test each component.

Code Development: Create modular functions.

System Integration: Combine sensor data, motor control, and logic.

Testing: Calibrate sensors and adjust motor speeds.

Challenges

Sensor Calibration: Improved accuracy.

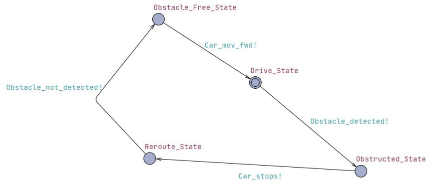
Motor Control: Fine-tuned PWM signals.

Real-time Decisions: Optimized code

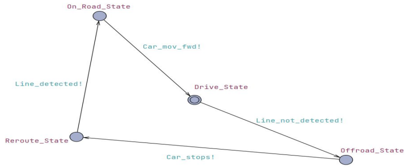
Simulation

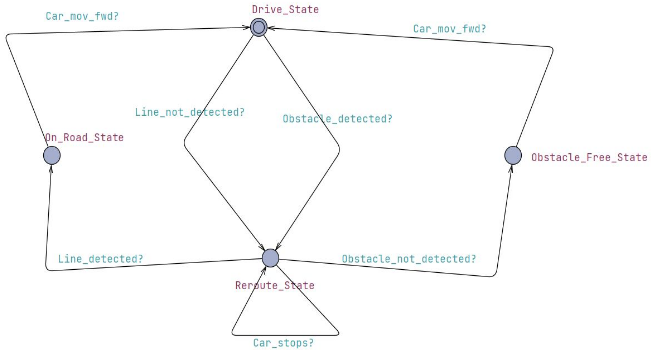
UPPAAL Simulation

Obstacle

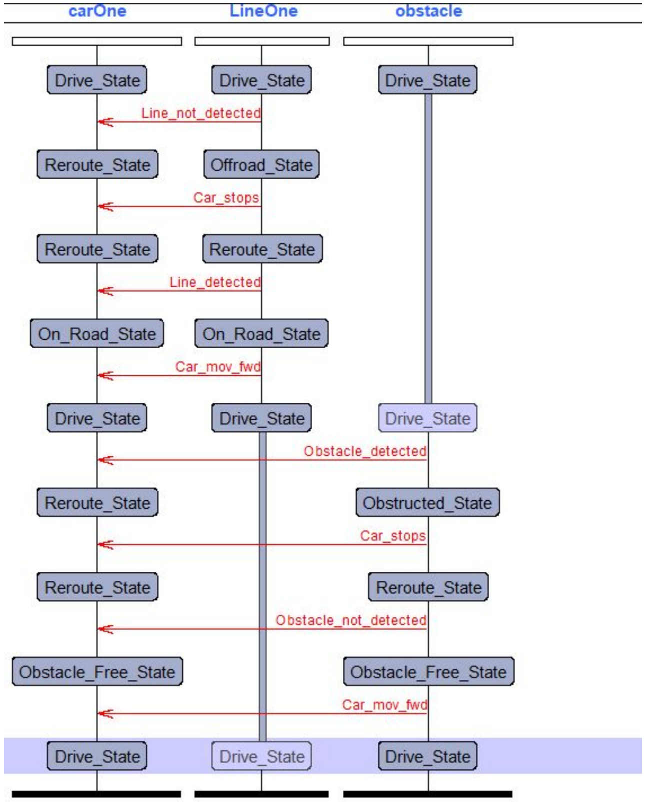


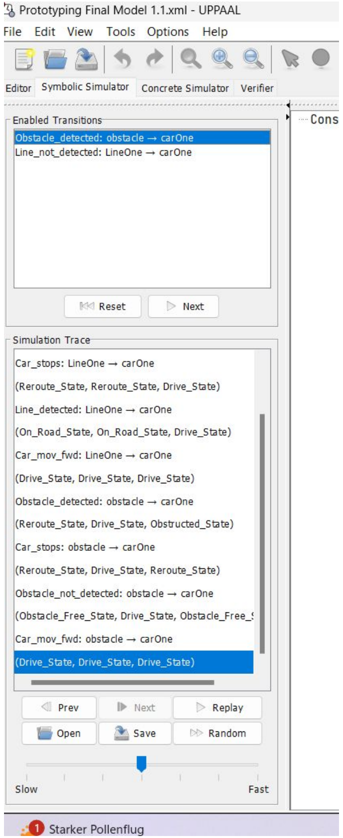
Line





Car


carOne
LineOne
obstacle
Drive_State
Drive_State
Drive_State
Line_not_detected
Reroute_State
Offroad_State
Car_stops
Reroute_State
Reroute_State
Line_detected
On_Road_State
On_Road_State
Car_mov_fwd
Drive_State
Drive_State
Drive_State
Obstacle_detected
Reroute_State
Obstructed_State
Car_stops
Reroute_State
Reroute_State
Obstacle_not_detected
Obstacle_Free_State
Obstacle_Free_State
Car_mov_fwd
Drive_State
Drive_State
Drive_State


Prototyping Final Model 1.1.xml - UPPAAL
File Edit View Tools Options Help
Editor
Symbolic Simulator
Concrete Simulator
Verifier
Cons
Enabled Transitions
Obstacle_detected: obstacle → carOne
Line_not_detected: LineOne → carOne
Reset
Next
Simulation Trace
Car_stops: LineOne → carOne
(Reroute_State, Reroute_State, Drive_State)
Line_detected: LineOne → carOne
(On_Road_State, On_Road_State, Drive_State)
Car_mov_fwd: LineOne → carOne
(Drive_State, Drive_State, Drive_State)
Obstacle_detected: obstacle → carOne
(Reroute_State, Drive_State, Obstructed_State)
Car_stops: obstacle → carOne
(Reroute_State, Drive_State, Reroute_State)
Obstacle_not_detected: obstacle → carOne
(Obstacle_Free_State, Drive_State, Obstacle_Free_
Car_mov_fwd: obstacle → carOne
(Drive_State, Drive_State, Drive_State)
Prev
Next
Replay
Open
Save
Random
Slow
Fast
Starker Pollenflug

Testing Troubleshooting and Debugging

- **Component Testing:**
    - **IR Sensors:** Verified under various lighting conditions.
    - **Ultrasonic Sensors:** Tested at different distances and angles.
    - **Color Sensors:** Tested with different colors and lighting.

- **Integration Testing:** Ensured seamless operation of sensors and motor controls.

- **Functional Testing:**
    - Tested line-following accuracy on various tracks.
    - Verified obstacle detection and color-based responses.

- **Stress Testing:** Assessed battery life and performance on different surfaces.
- **Code Review & Debugging:** Used real-time monitoring to resolve issues.
- **User Testing:** Incorporated feedback to improve usability and performance.

Challenges and Limitations

Limitations of the Current Prototype

R Sensors: Affected by ambient light and reflective surfaces.

Ultrasonic Sensors: Accuracy influenced by surface texture and obstacle angle.

Motor Control: Basic, with limited speed and turning adjustments.

Color Sensors: Inconsistent under varying lighting conditions.

Battery Life: High power consumption limits performance.

Scalability: Limited scope for adding new sensors or functionalities.

Surface Effectiveness: Less effective on uneven or reflective surfaces.

Processing Power: Arduino Uno’s limited processing power.

Obstacle Avoidance Algorithm: Basic, lacks sophisticated pathfinding.

Future Work

Potential Improvements

- **Add More Sensors**: Incorporate additional ultrasonic sensors or use LiDAR for 360-degree obstacle detection.
- **Improve Accuracy**: Use higher-resolution sensors for more precise measurements and detection capabilities.
- **Multi-Sensor Fusion**: Combine data from multiple sensors to make more informed decisions.
- **Traffic Sign Detection**: Implement traffic sign recognition to obey road signs and signals.
- **Dynamic Adjustments**: Implement algorithms to adjust speed and direction based on real-time sensor data.

Future Development Plans

- Implement advanced algorithms like PID control for precise line following.
- Integrate GPS modules for outdoor navigation to follow predefined routes.
- Expand color sensor usage for more accurate color detection..
- Add a camera module for advanced obstacle recognition and lane detection.
- Add wireless modules (Bluetooth, Wi-Fi) for remote control and monitoring.

The End

Thank you for your Attention!

Q&A Session