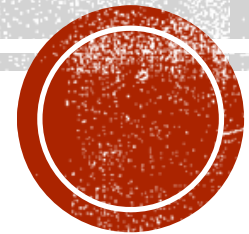# PROTOTYPING- SS25

Line Follower Robot
-Team D2

By
Md Jubair Salehin Razin
Faysal Ahammed Tonmoy
Md Ratul Ahmed Alvi
Soaib Ferdous

PROJECT OVERVIEW

- Building Line Follower Robot powered by Arduino
- Equipped with ultrasonic sensors for detecting obstacles
- Follows black lines using infrared (IR) sensors
- Detects object colors using a color recognition sensor
- Driven by dual DC motors for smooth movement
- Includes a servo motor to help remove or bypass obstacles

Goal: To showcase intelligent navigation, smart obstacle detection and avoidance, color-based decision making, and adaptive robotic behavior using sensors and actuators.
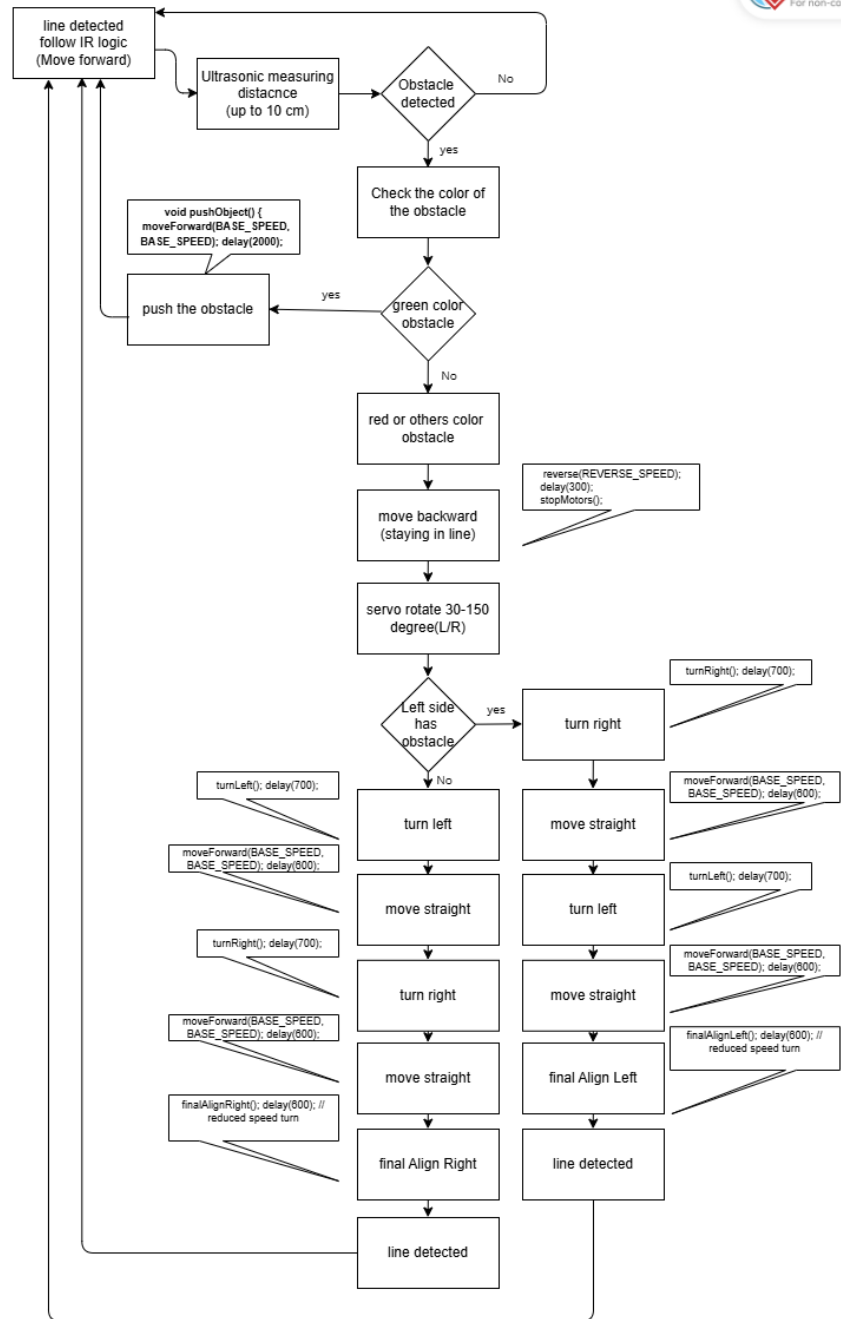
Block Diagram and IR sensor's logic:
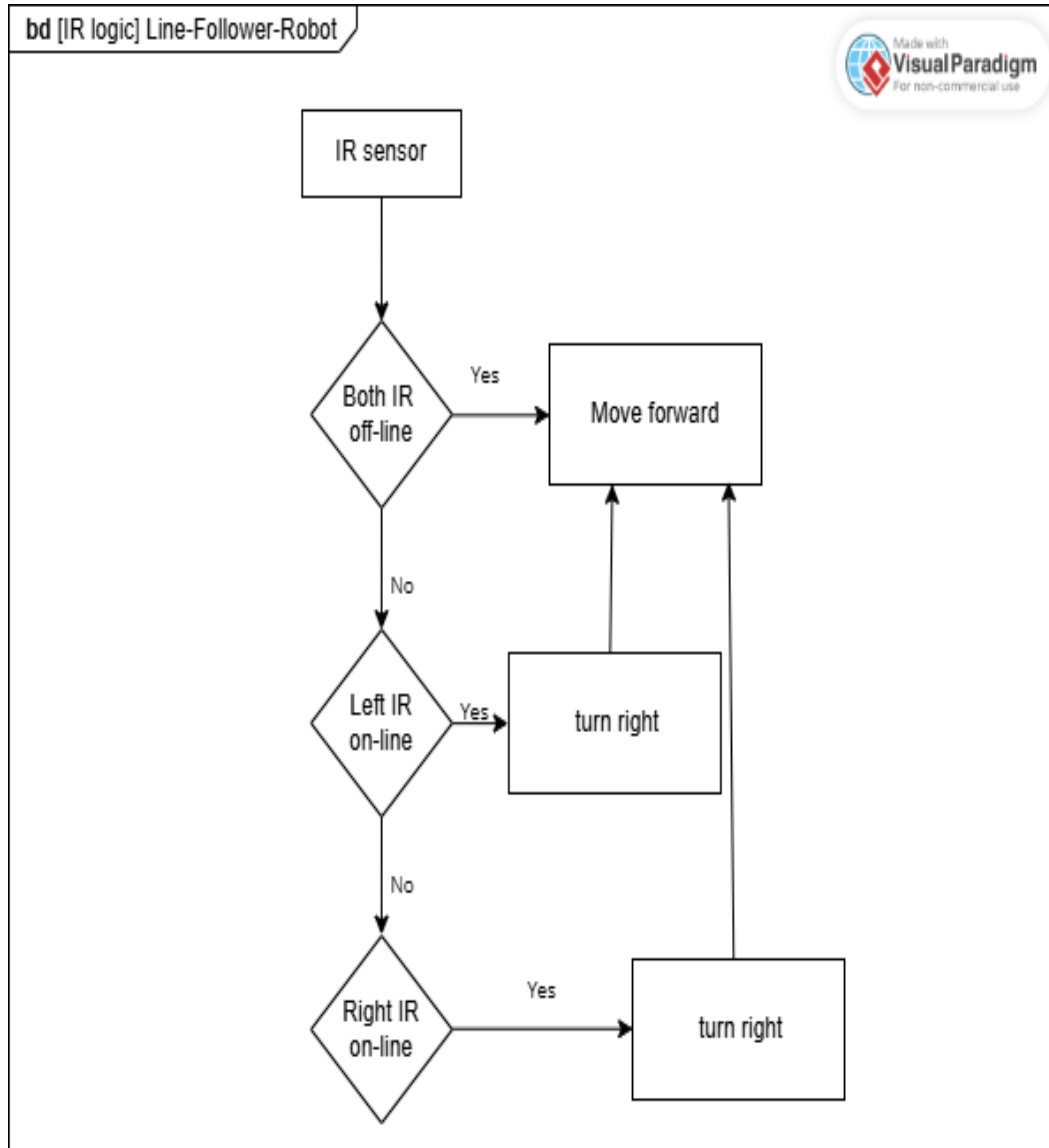

```
167   void executeLineFollowing() {
168     if (!leftOnLine && !rightOnLine) moveForward(BASE_SPEED, BASE_SPEED);
169     else if (leftOnLine && !rightOnLine) rightSensorReliable ? turnRight() : gentleTurnRight();
170     else if (!leftOnLine && rightOnLine) turnLeft();
171     else moveForward(BASE_SPEED * 0.6, BASE_SPEED * 0.6);
172
173     if (!rightSensorReliable && leftOnLine)
174       moveForwardWithBias(BASE_SPEED, 0.8, 1.0);
175   }
```

Flowchart:

- line detected follow IR logic (Move forward)
- Ultrasonic measuring distacnce (up to 10 cm)
- Obstacle detected — No → (loop back); yes ↓
- Check the color of the obstacle
- green color obstacle — yes → push the obstacle
  - void pushObject() { moveForward(BASE_SPEED, BASE_SPEED); delay(2000);
- No ↓
- red or others color obstacle
- move backward (staying in line)
  - reverse(REVERSE_SPEED); delay(300); stopMotors();
- servo rotate 30-150 degree(L/R)
- Left side has obstacle
  - yes → turn right
    - turnRight(); delay(700);
    - move straight
      - moveForward(BASE_SPEED, BASE_SPEED); delay(600);
    - turn left
      - turnLeft(); delay(700);
    - move straight
      - moveForward(BASE_SPEED, BASE_SPEED); delay(600);
    - final Align Left
      - finalAlignLeft(); delay(600); // reduced speed turn
    - line detected
  - No → turn left
    - turnLeft(); delay(700);
    - move straight
      - moveForward(BASE_SPEED, BASE_SPEED); delay(600);
    - turn right
      - turnRight(); delay(700);
    - move straight
      - moveForward(BASE_SPEED, BASE_SPEED); delay(600);
    - final Align Right
      - finalAlignRight(); delay(600); // reduced speed turn
    - line detected

Code:

```cpp
63    void loop() {
64      unsigned long currentTime = millis();
65      int distance = readDistance();
66
67      if (distance > 0 && distance <= OBSTACLE_DISTANCE_THRESHOLD) {
68        stopMotors(); delay(300);
```

```cpp
String getDetectedColor() {
  if (redFrequency < greenFrequency && redFrequency < blueFrequency) return "Red";
  if (greenFrequency < redFrequency && greenFrequency < blueFrequency) return "Green";
  if (blueFrequency < redFrequency && blueFrequency < greenFrequency) return "Blue";
  return "Unknown";
}
```

```cpp
readColor();
String color = getDetectedColor();
if (color == "Green") {
  pushObject(); // push if green
} else {
  reverse(REVERSE_SPEED); delay(300);
  stopMotors();
  ...
}
```

```cpp
230    void pushObject() {
231      moveForward(BASE_SPEED, BASE_SPEED); delay(2000); // updated from 1000 to 2000
232      stopMotors(); delay(500);
233    }
```

```cpp
90        if (distLeft > distRight) {
91          turnLeft(); delay(700);
92          moveForward(BASE_SPEED, BASE_SPEED); delay(600);
93          turnRight(); delay(700);
94          moveForward(BASE_SPEED, BASE_SPEED); delay(600);
95          finalAlignRight(); delay(600); // reduced speed turn
96        } else {
97          turnRight(); delay(700);
98          moveForward(BASE_SPEED, BASE_SPEED); delay(600);
99          turnLeft(); delay(700);
100         moveForward(BASE_SPEED, BASE_SPEED); delay(600);
101         finalAlignLeft(); delay(600); // reduced speed turn
102       }
103       stopMotors();
104       return;
105     }
106   }
```

```cpp
235    void finalAlignRight() {
236      digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW); analogWrite(ENA, 60);
237      digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH); analogWrite(ENB, 60);
238    }
239
240    void finalAlignLeft() {
241      digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH); analogWrite(ENA, 60);
242      digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW); analogWrite(ENB, 60);
243    }
```

MODEL-WISE CODE EXPLANATION

➢ **IR Sensor Code for following the line:**

https://github.com/alvi231078/D2---Line-Follower-Robot/blob/main/Main%20Code/Code%20for%20presentation/IR%20Sensor%20Code%20for%20following%20the%20line%3A

➢ **Color Sensor Code:**

https://github.com/alvi231078/D2---Line-Follower-Robot/blob/main/Main%20Code/Code%20for%20presentation/Color%20Sensor%20Code%3A%20RGB

➢ **Ultrasonic Sensor Code:**

https://github.com/alvi231078/D2---Line-Follower-Robot/blob/main/Main%20Code/Code%20for%20presentation/Ultrasonic%20Sensor%20Code%3A

➢ **Servo Code:**

https://github.com/alvi231078/D2---Line-Follower-Robot/blob/main/Main%20Code/Code%20for%20presentation/4.%20Code%20for%20servo%20motor

MODEL-WISE CODE EXPLANATION

➢ **Line Following Robot: Line Tracking + Smart Obstacle Bypass**

https://github.com/alvi231078/D2---Line-Follower-Robot/blob/main/Main%20Code/Code%20for%20presentation/6.%20Line%20Following%20Robot%3A%20Line%20Tracking%20%2B%20Smart%20Obstacle%20Bypass
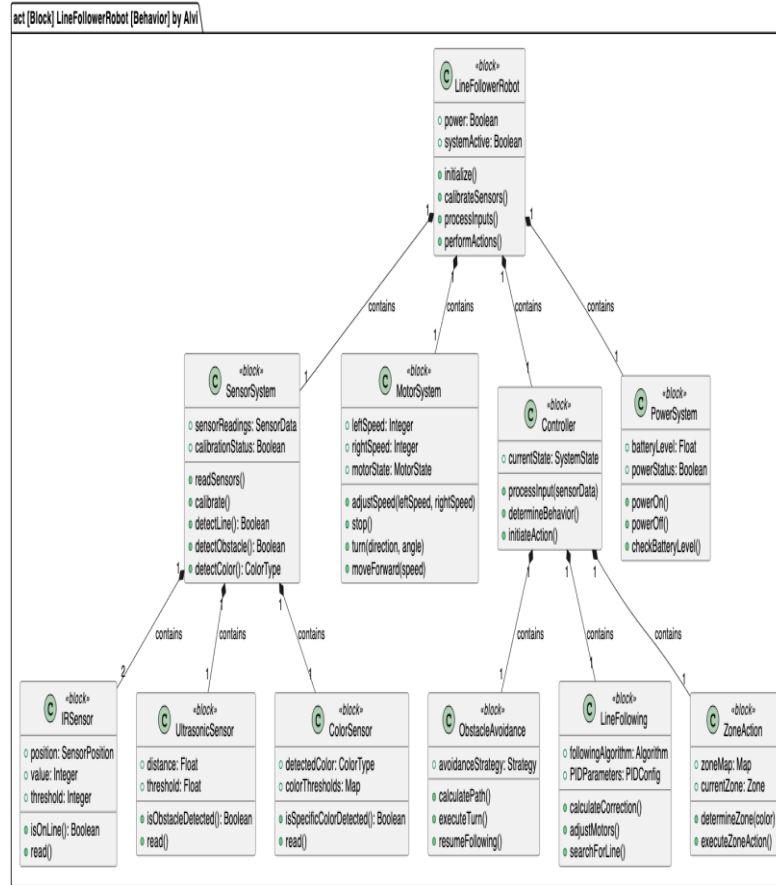
➢ **Line Following Robot with Line, Obstacle, and Color Sensing**

https://github.com/alvi231078/D2---Line-Follower-Robot/blob/main/Main%20Code/Code%20for%20presentation/7.%20Line%20Following%20Robot%20with%20Line%2C%20Obstacle%2C%20and%20Color%20Sensing
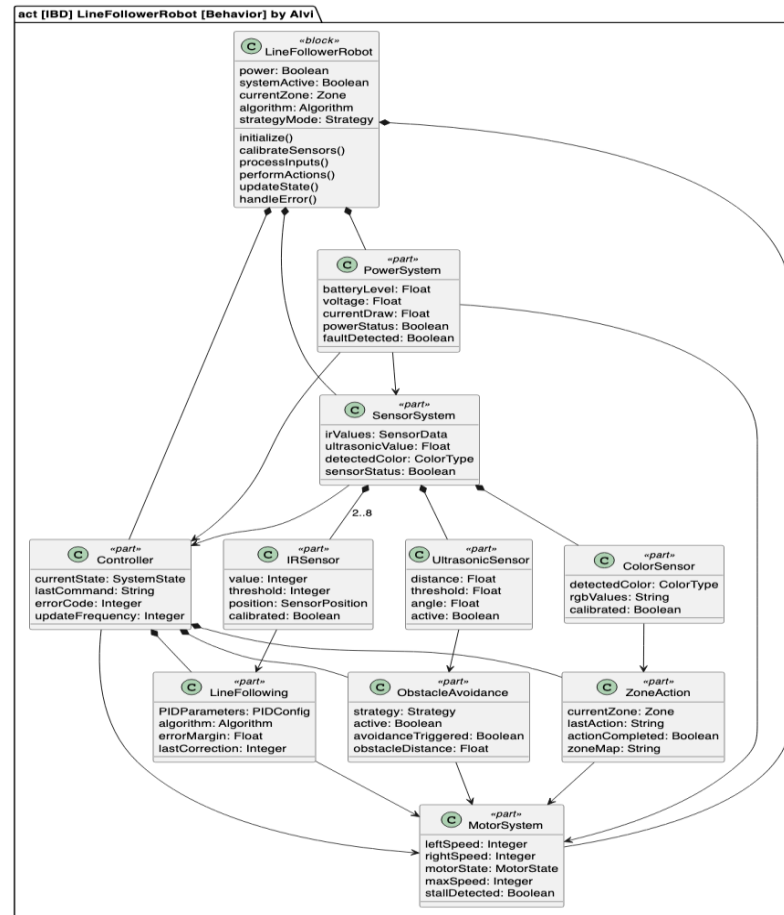
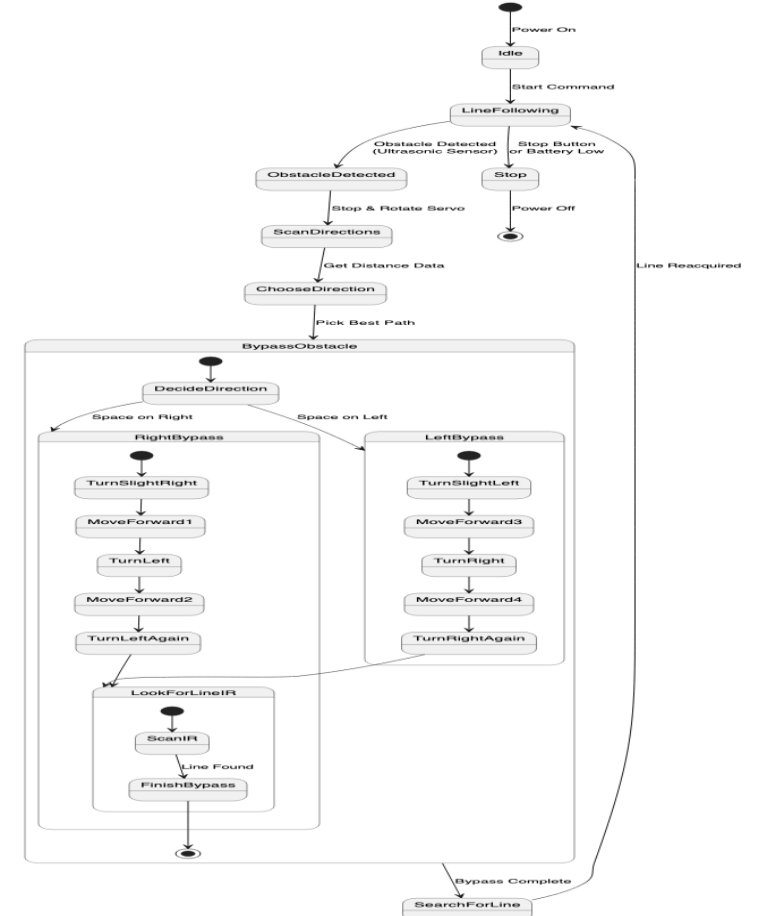# SYSTEM MODELING USING SYSML

Block Definition Diagram

Internal Block Diagram

State Machine Diagram

To understand the behaviour:

Line Follower Robot - Activity Diagram by Alvi

act [Block] LineFollowerRobot [Behavior] by Alvi

Power On → Initialize System → Enter Idle State → Transition to LineFollowing State → Read IR Sensors → Line Detected?

yes → Adjust Motor Speeds to Follow Line → Read Ultrasonic Sensor → Obstacle Detected?

yes → Enter ObstacleDetected State → Stop Motors → ScanDirections with Servo Motor → ChooseDirection Based on Distance → Go Right?

yes → Enter RightBypass State → Turn Right → Bypass Obstacle
Enter LeftBypass State → Turn Left → Bypass Obstacle

LookForLineIR → FinishBypass → Return to LineFollowing

Continue Following Line

Enter LookForLineIR State → Check Last Known Direction → Was Last Left?

yes → Turn Left Slowly While Scanning
Turn Right Slowly While Scanning

Line Found?

yes → Resume LineFollowing
Begin Zig-Zag Scanning → Turn Left Slightly → Check For Line → Line Found?

yes → Resume LineFollowing
Turn Right Slightly → Check For Line → Scan Attempts < Max

Line Still Not Found?

yes → Enter Fallback Behavior (e.g., Stop or Alert)

Battery OK? and Stop Button Not Pressed → Enter Stop State → System Shutdown

PROTOTYPING & DESIGN

➢ Physical Design (SolidWorks)

PROTOTYPING & DESIGN

➢ Virtual Prototypes (TinkerCad)
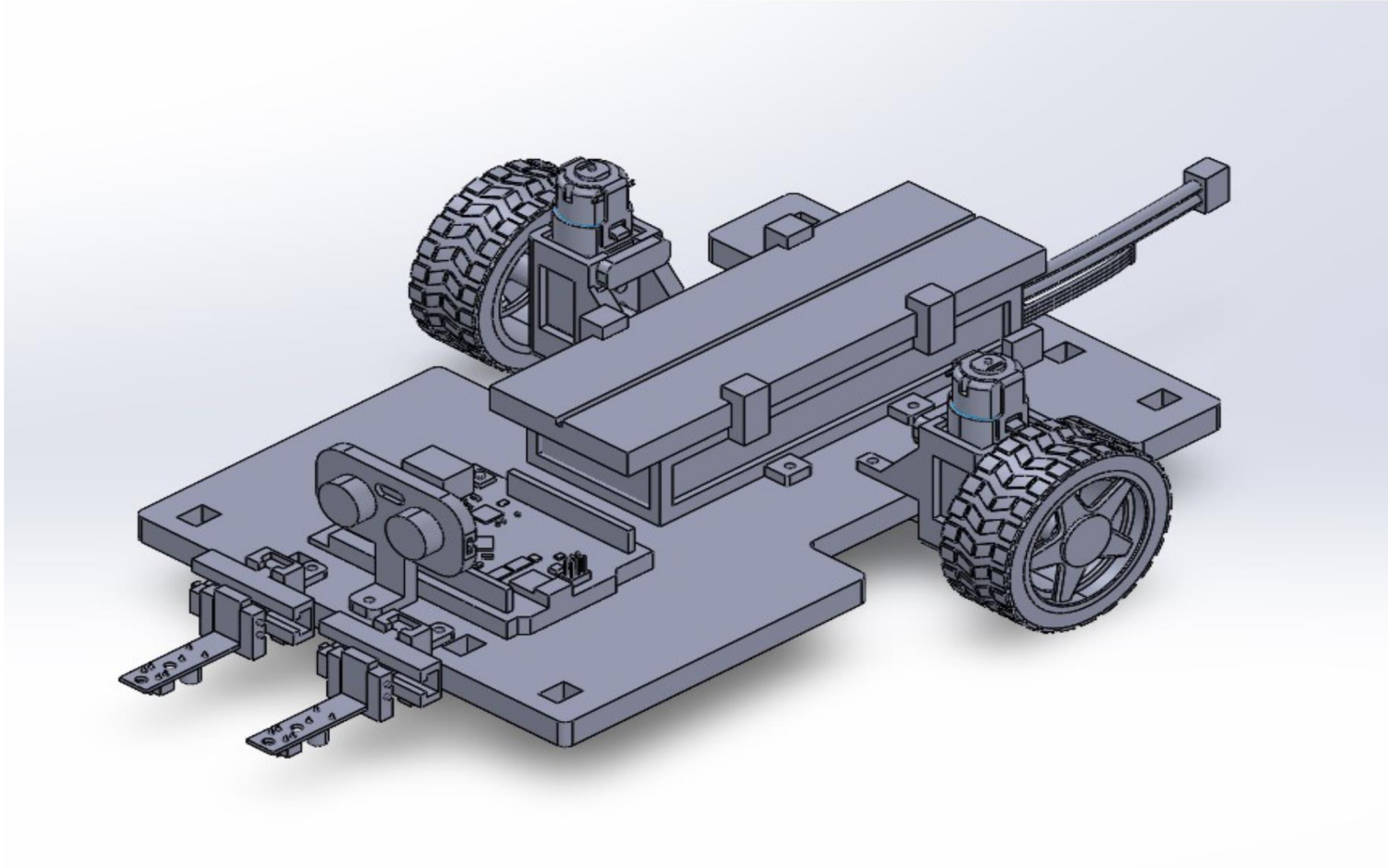
CIRCUIT DESIGN

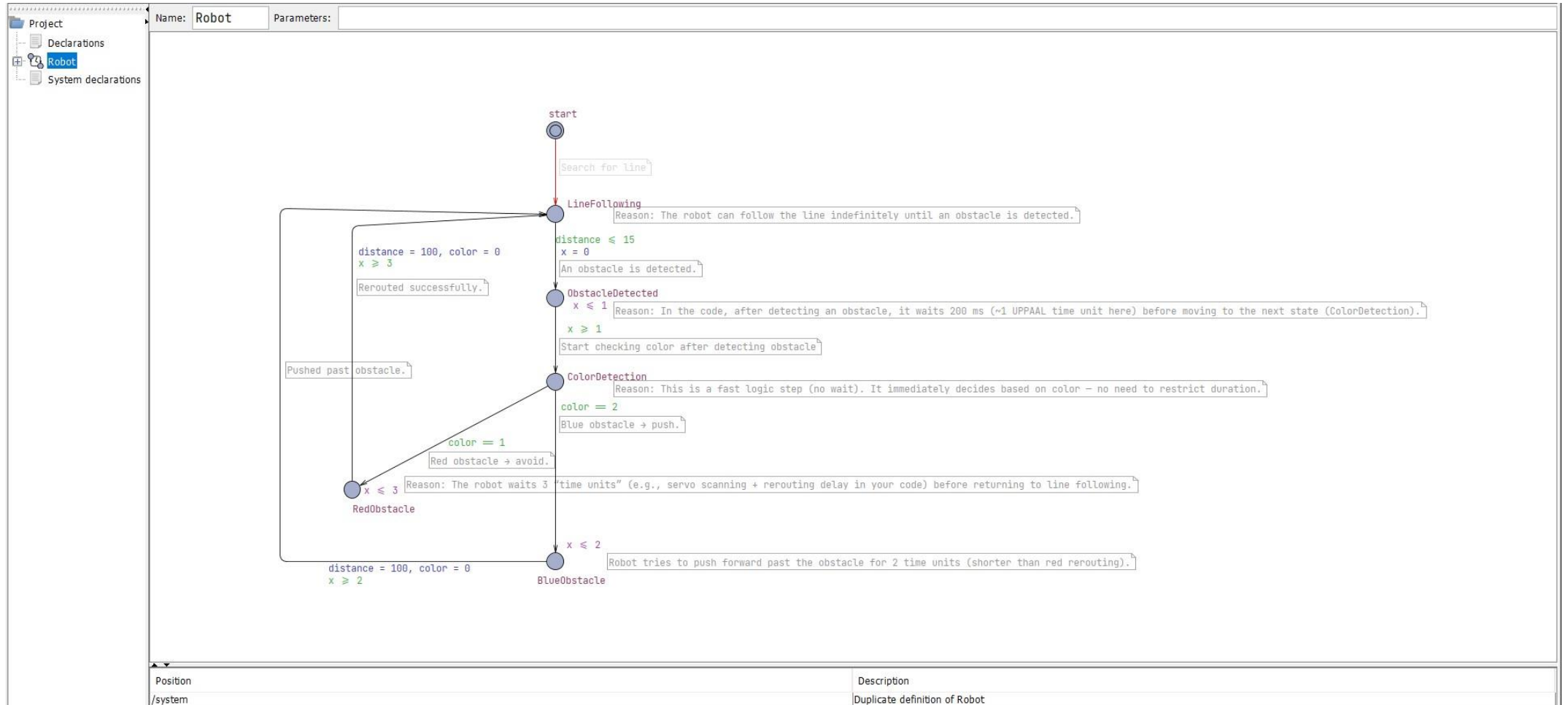➢ Sensor & Motor Wiring and Complete Circuit Layout
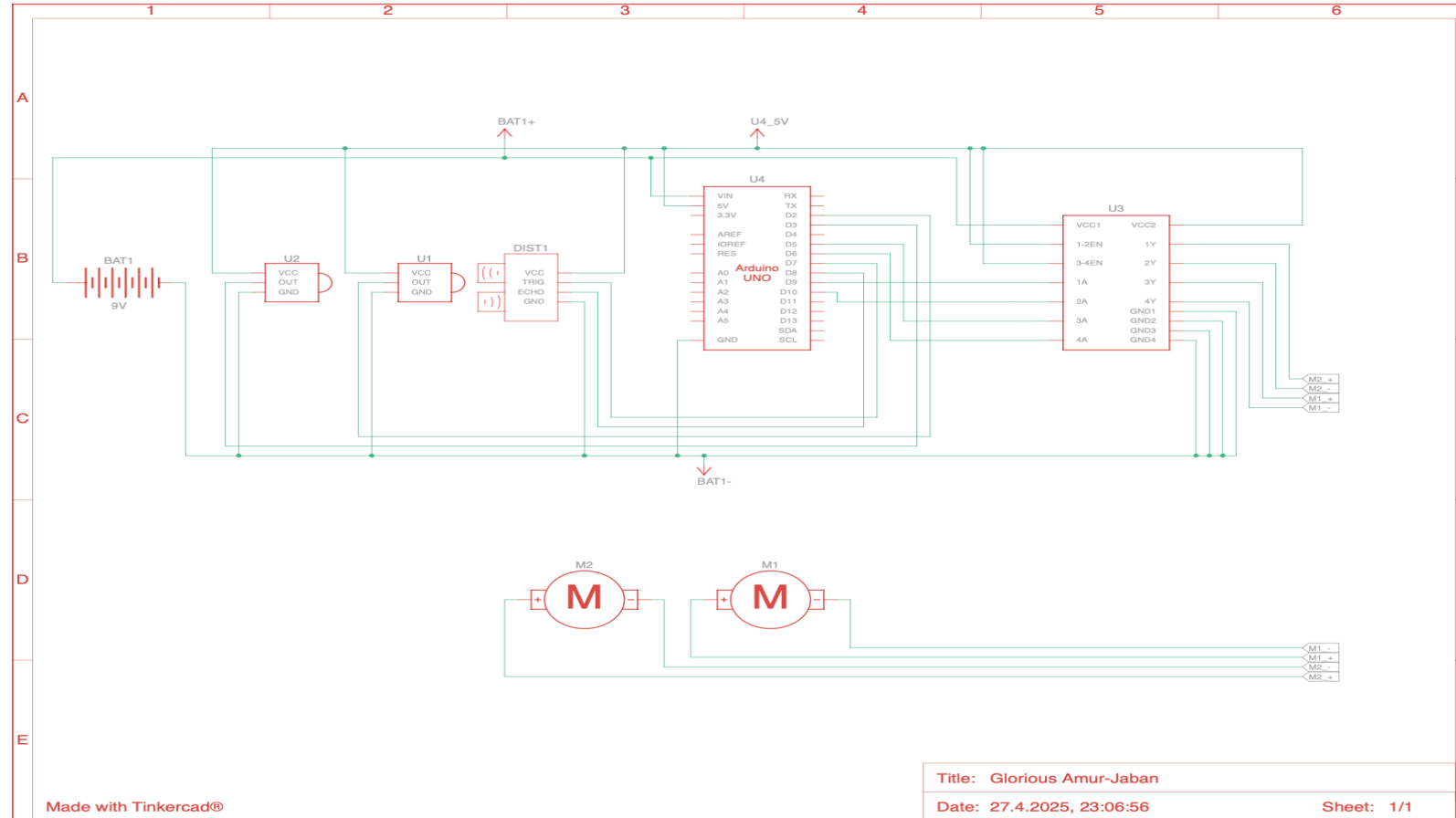
Circuit Layout

Sensors:
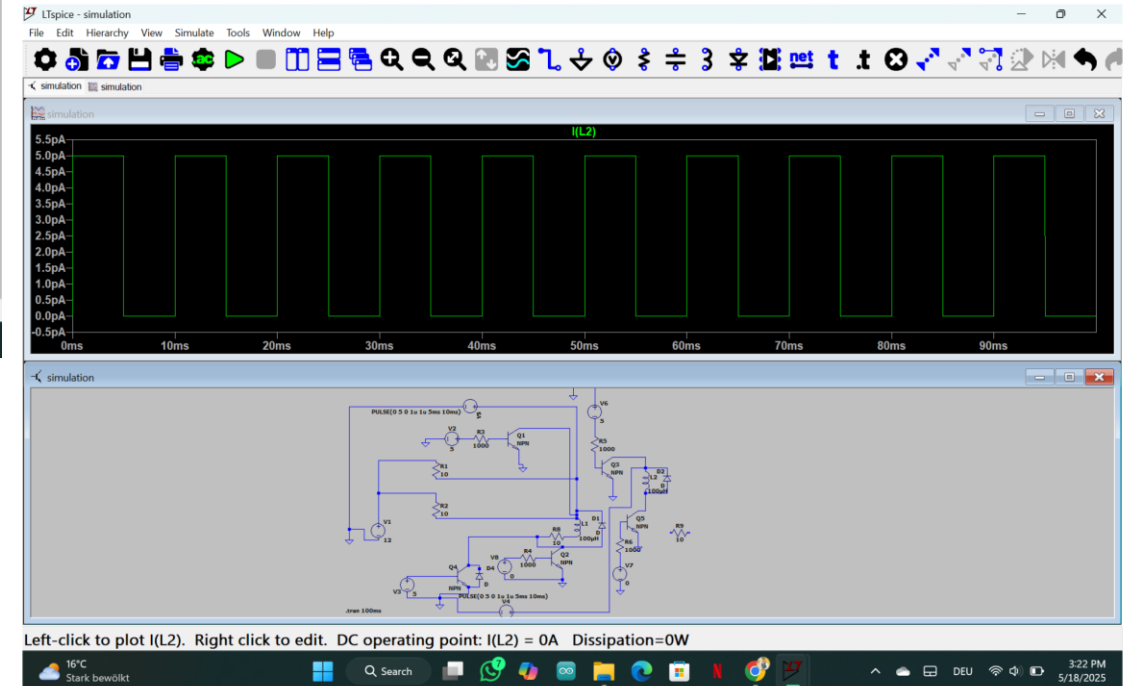
1. Line Tracking IR Sensor
2. Ultrasonic Sensor
3. Color Sensor

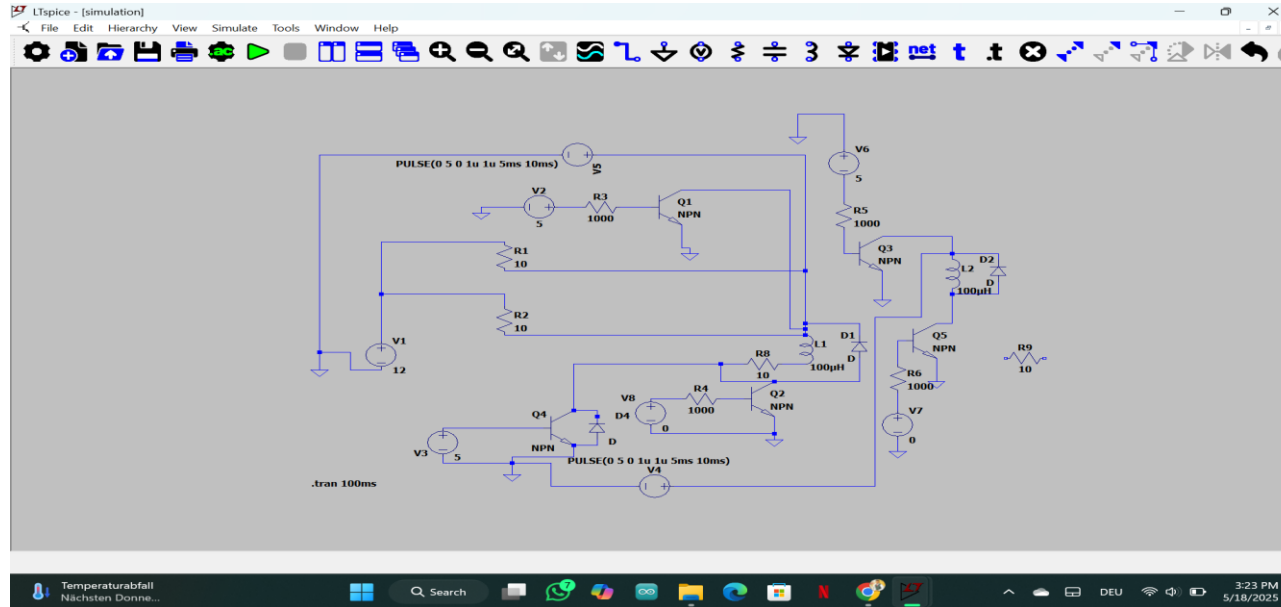# SOFTWARE IMPLEMENTATION

➢ UPPAAL Timed Automata

SOFTWARE IMPLEMENTATION

➢ LTspice Circuit Simulation

PROOF OF CONCEPT: VIDEO

- https://youtu.be/kWiVvY1VcH8?si=XqxR6bjObIauFDW2

KEY ACHIEVEMENTS AND CONCLUSION

Collaborative Outcomes:

Integrated Multi-Sensor System

Autonomous Navigation

Robust Embedded Design

Applied SysML & UML Thinking

Design Principles Used

This project showcases an autonomous embedded system combining line tracking, obstacle avoidance, and color-based decisions forming a foundation for future intelligent robotics.