Design and Implementation of a Line Following and Obstacle Avoidance Robot

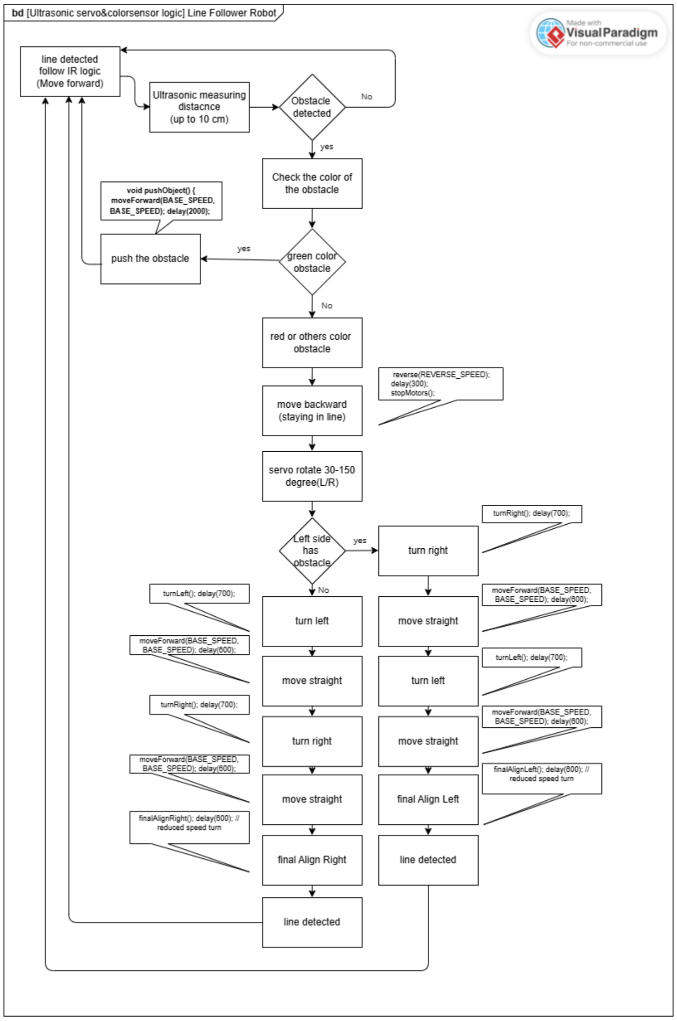
(Final Technical Report)

Team D2: Faysal Ahammed Tonmoy, Jubair Salehin Razin, Md Ratul Ahmed Alvi, Soaib Ferdous
Prototyping and Systems Engineering
BSc Electronic Engineering
Hochschule Hamm-Lippstadt, Lippstadt, Germany

***Abstract*—This paper presents the design, development, and implementation of an Arduino-based robot capable of following a black line and avoiding obstacles in its path. The system integrates IR sensors for line detection and an ultrasonic sensor mounted on a servo motor for environmental scanning. The robot's decision-making algorithm enables smooth path-following while dynamically reacting to obstacles. Both hardware and software components were prototyped virtually before real-world testing. Results show reliable performance in various test scenarios.**

***Index Terms*—Line follower robot, Obstacle avoidance, Arduino, Servo motor, Ultrasonic sensor**

I. System Modeling using SysML


bd [Ultrasonic servo&colorsensor logic] Line Follower Robot
line detected follow IR logic (Move forward)
Ultrasonic measuring distacnce (up to 10 cm)
Obstacle detected
No
yes
Check the color of the obstacle
push the obstacle
yes
green color obstacle
No
red or others color obstacle
move backward (staying in line)
servo rotate 30-150 degree(L/R)
Left side has obstacle
yes
turn right
No
turn left
move straight
turn right
move straight
final Align Right
line detected
move straight
turn left
move straight
final Align Left
line detected

```
void pushObject() { moveForward(BASE_SPEED, BASE_SPEED); delay(2000);
reverse(REVERSE_SPEED); delay(300); stopMotors();
turnRight(); delay(700);
turnLeft(); delay(700);
moveForward(BASE_SPEED, BASE_SPEED); delay(600);
turnRight(); delay(700);
moveForward(BASE_SPEED, BASE_SPEED); delay(600);
finalAlignRight(); delay(600); // reduced speed turn
moveForward(BASE_SPEED, BASE_SPEED); delay(600);
turnLeft(); delay(700);
moveForward(BASE_SPEED, BASE_SPEED); delay(600);
finalAlignLeft(); delay(600); // reduced speed turn
```

Fig. 1. Overall System Block Diagram of the Robot

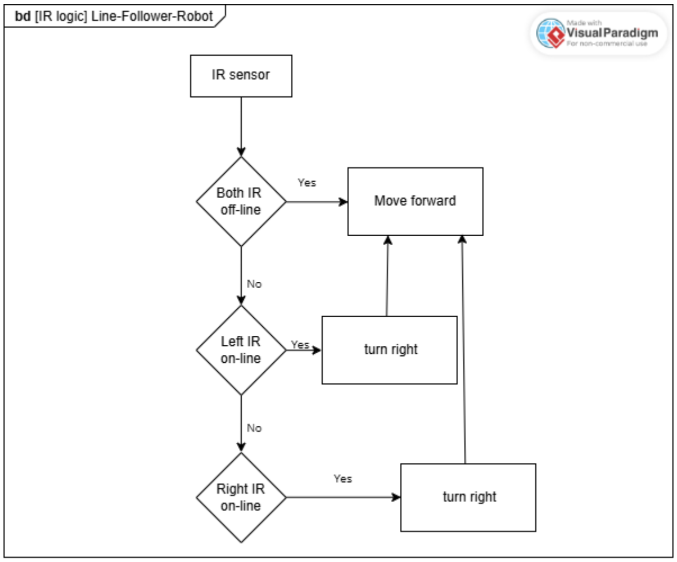


Fig. 2. IR Line Following Logic Block Diagram

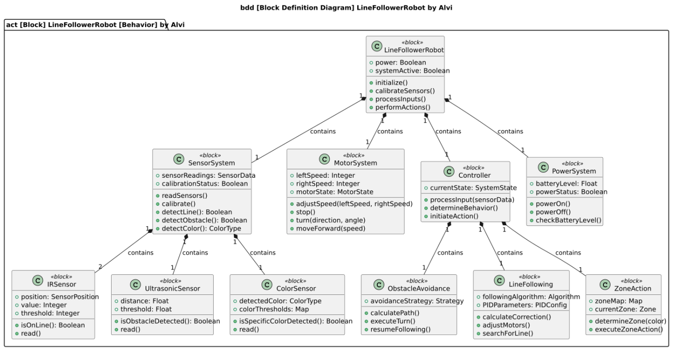


Fig. 3. SysML Block Definition Diagram (BDD)

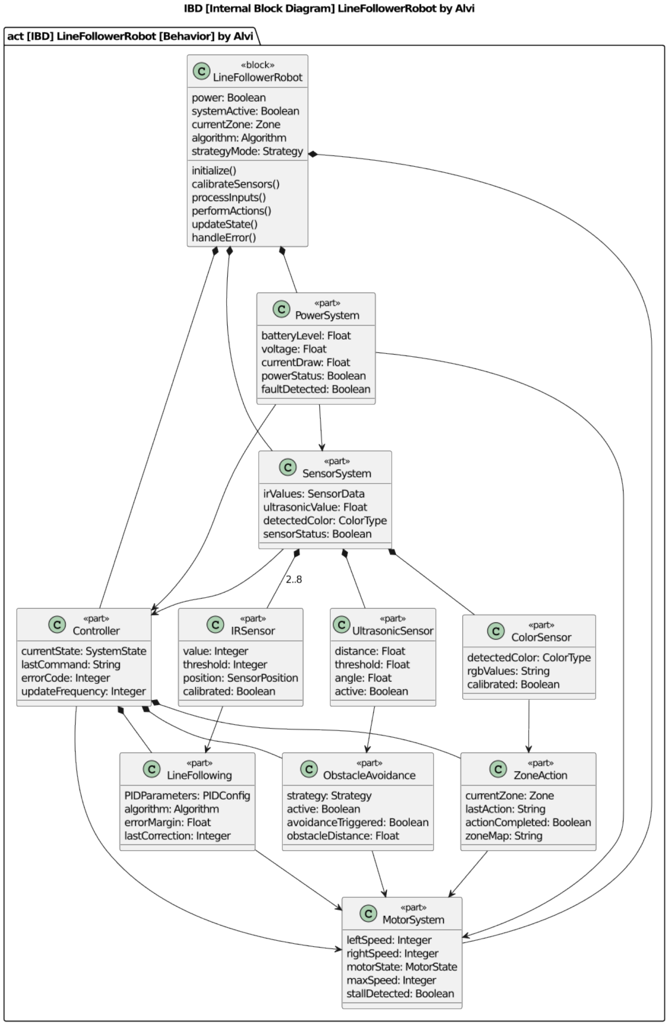


Fig. 4. SysML Internal Block Diagram (IBD)

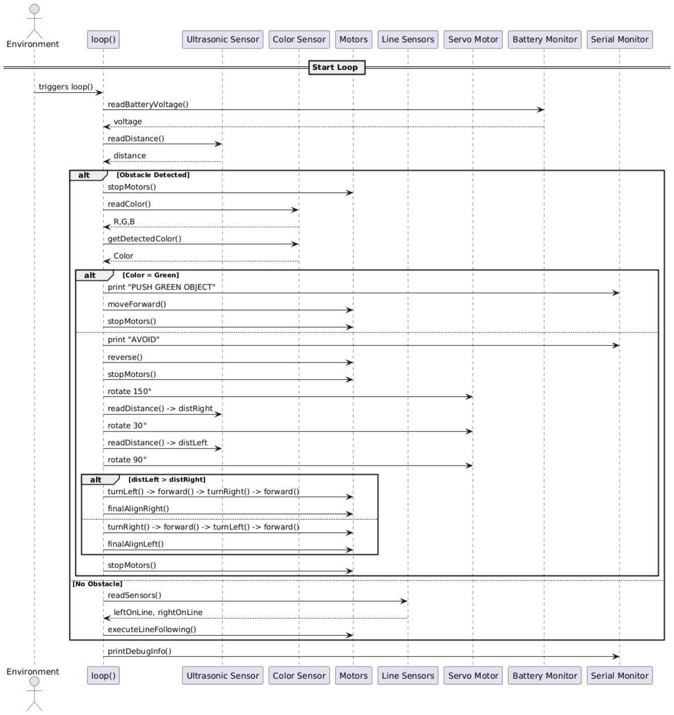


Fig. 5. SysML Sequence Diagram for Obstacle Avoidance

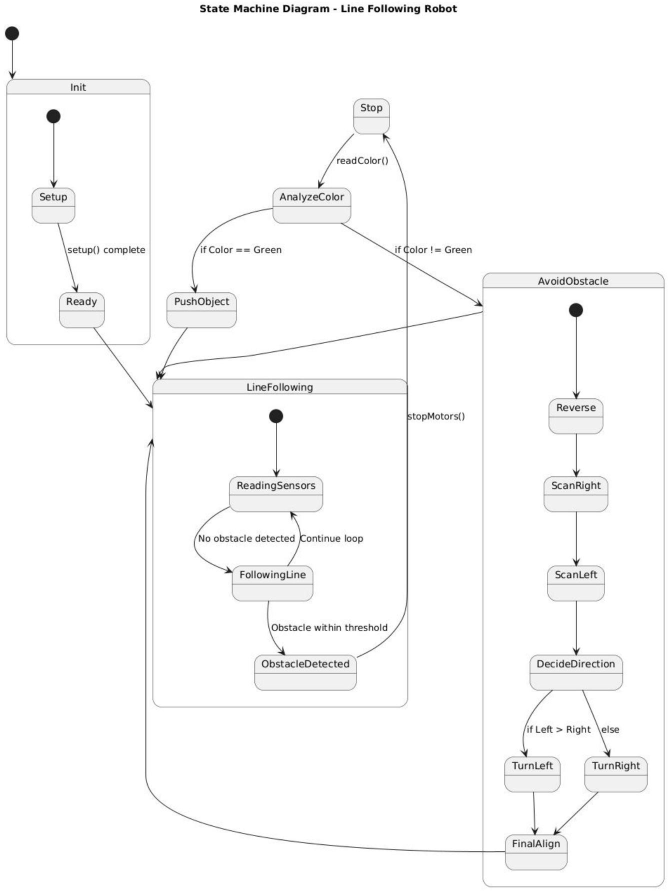


Fig. 6. SysML State Machine Diagram

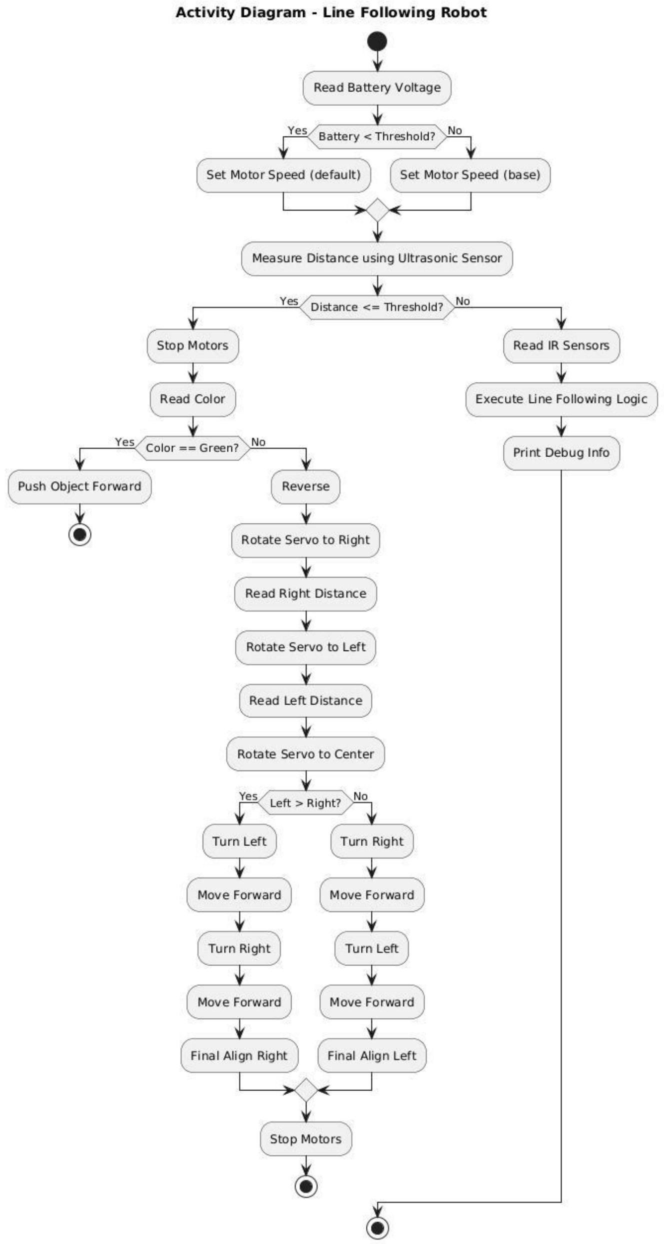


Fig. 7. SysML Activity Diagram for Robot Behavior

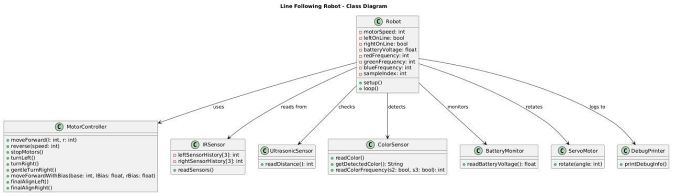


Fig. 8. SysML Class Diagram for Software Components

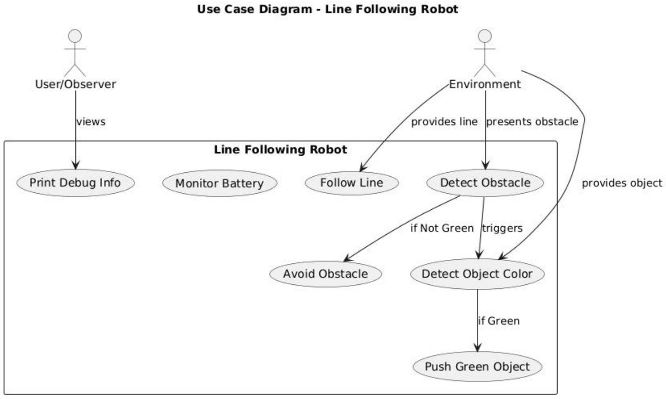


Fig. 9. SysML Use Case Diagram for Robot Functions

II. Prototyping and Design

A. Electronic Components

- Arduino Uno R3
- L298N Motor Driver
- 2 × IR Sensors
- 1 × Ultrasonic Sensor (HC-SR04)
- 1 × Servo Motor (SG90)
- 2 × DC Motors with Wheels
- Chassis, Wires, Breadboard

B. Virtual and Physical Prototypes

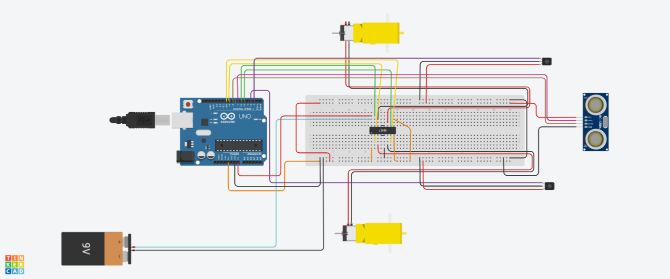


Fig. 10. Tinkercad Simulation: Full Circuit Layout

C. Assembled Car Design

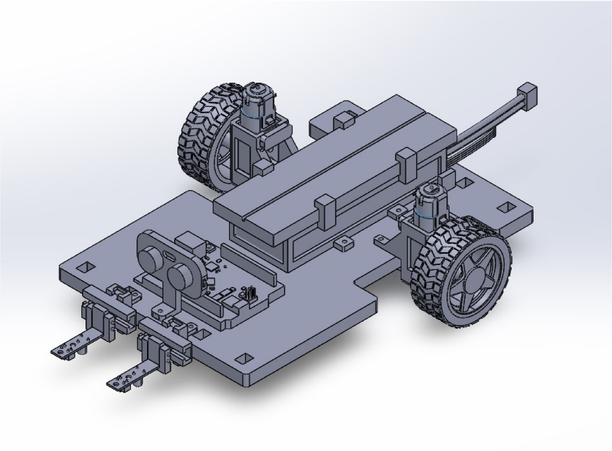


Fig. 11. SolidWorks Render of the Robot Design

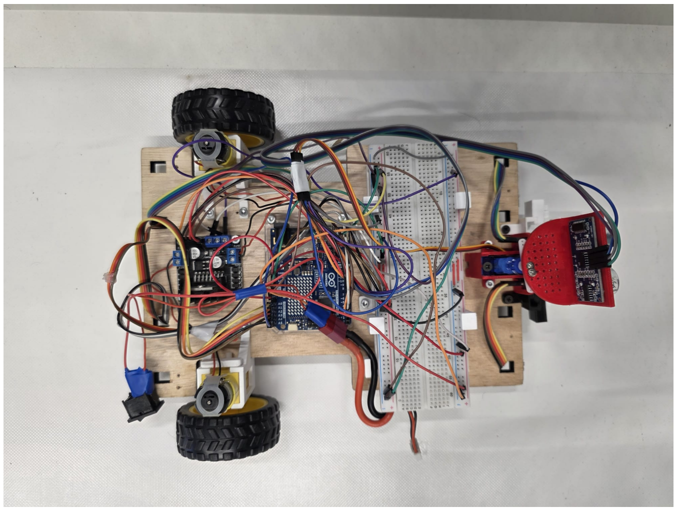


Fig. 12. Top view of the assembled robot

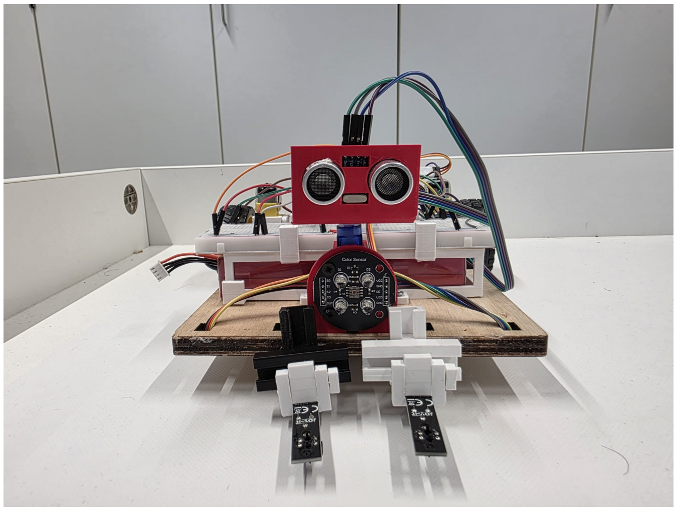


Fig. 13. Front view showing ultrasonic and color sensors

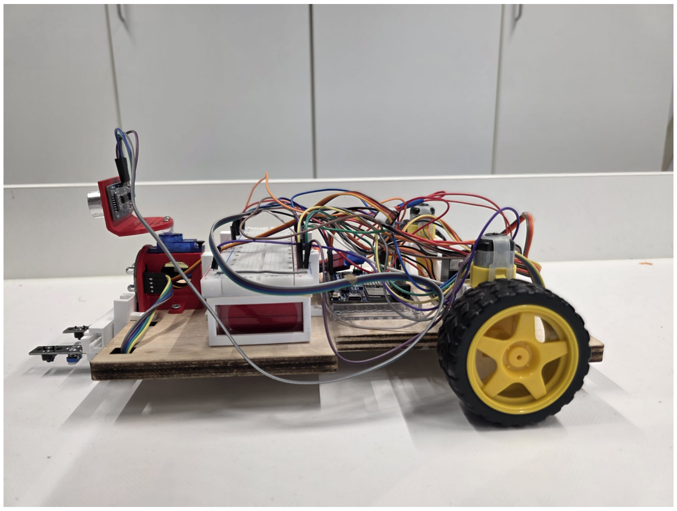


Fig. 14. Side view showing servo and motor arrangement

III. Circuit Design

- IR sensors → pins 2 and 3
- Ultrasonic → TRIG (12), ECHO (13)
- Servo → pin 9
- Motor driver → IN1-IN4: pins 5–8; ENA/ENB: pins 10–11

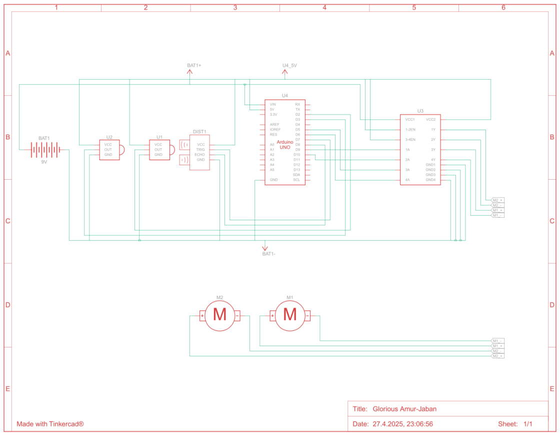
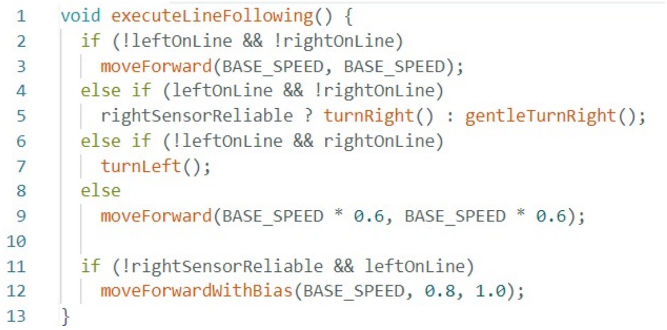


Fig. 15. Complete circuit layout of the robot

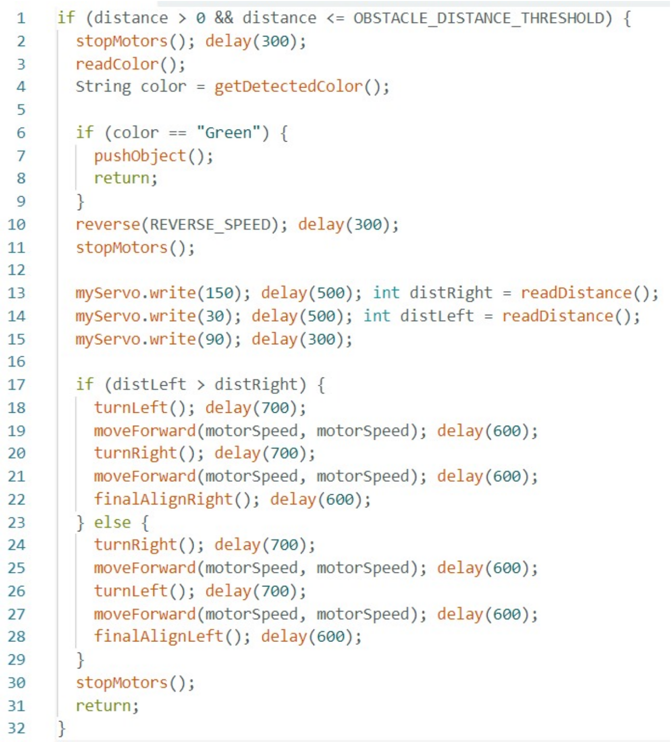
IV. Software Implementation

A. Core Functional Logic



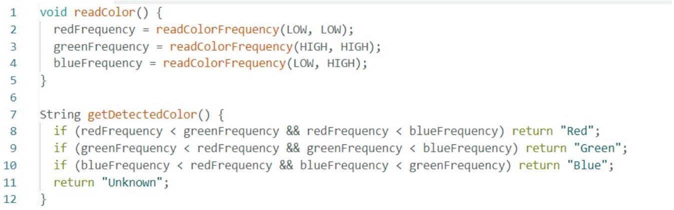
```
void executeLineFollowing() {
  if (!leftOnLine && !rightOnLine)
    moveForward(BASE_SPEED, BASE_SPEED);
  else if (leftOnLine && !rightOnLine)
    rightSensorReliable ? turnRight() : gentleTurnRight();
  else if (!leftOnLine && rightOnLine)
    turnLeft();
  else
    moveForward(BASE_SPEED * 0.6, BASE_SPEED * 0.6);

  if (!rightSensorReliable && leftOnLine)
    moveForwardWithBias(BASE_SPEED, 0.8, 1.0);
}
```

Fig. 16. 1. Line Following Logic



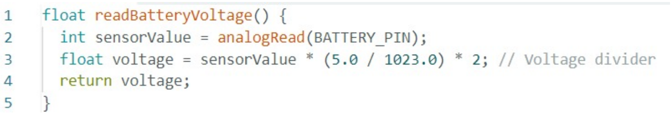
```
if (distance > 0 && distance <= OBSTACLE_DISTANCE_THRESHOLD) {
  stopMotors(); delay(300);
  readColor();
  String color = getDetectedColor();

  if (color == "Green") {
    pushObject();
    return;
  }
  reverse(REVERSE_SPEED); delay(300);
  stopMotors();

  myServo.write(150); delay(500); int distRight = readDistance();
  myServo.write(30); delay(500); int distLeft = readDistance();
  myServo.write(90); delay(300);

  if (distLeft > distRight) {
    turnLeft(); delay(700);
    moveForward(motorSpeed, motorSpeed); delay(600);
    turnRight(); delay(700);
    moveForward(motorSpeed, motorSpeed); delay(600);
    finalAlignRight(); delay(600);
  } else {
    turnRight(); delay(700);
    moveForward(motorSpeed, motorSpeed); delay(600);
    turnLeft(); delay(700);
    moveForward(motorSpeed, motorSpeed); delay(600);
    finalAlignLeft(); delay(600);
  }
  stopMotors();
  return;
}
```

Fig. 17. 2. Obstacle Avoidance with Servo Scanning



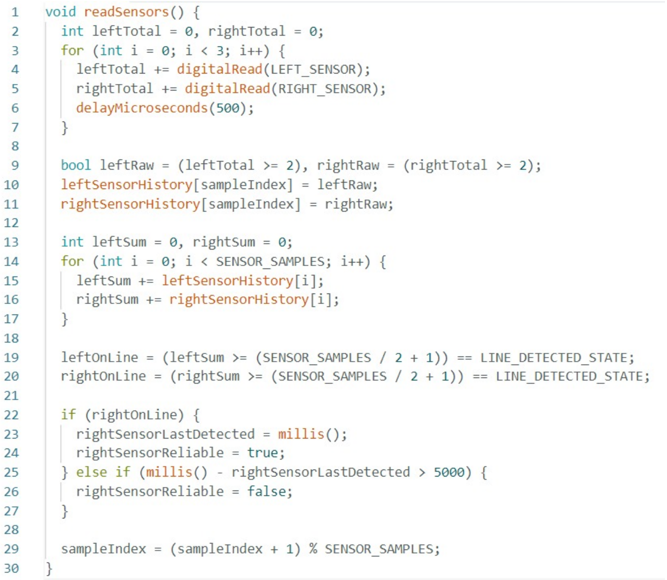
```
void readColor() {
  redFrequency = readColorFrequency(LOW, LOW);
  greenFrequency = readColorFrequency(HIGH, HIGH);
  blueFrequency = readColorFrequency(LOW, HIGH);
}

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";
}
```

Fig. 18. 3. Color Detection Logic



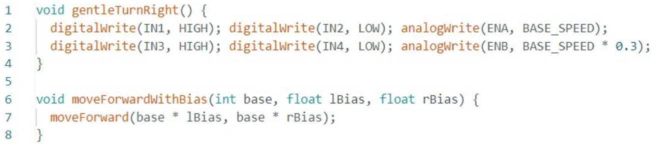
```
float readBatteryVoltage() {
  int sensorValue = analogRead(BATTERY_PIN);
  float voltage = sensorValue * (5.0 / 1023.0) * 2; // Voltage divider
  return voltage;
}
```

Fig. 19. 4. Battery Voltage Monitoring



```
void readSensors() {
  int leftTotal = 0, rightTotal = 0;
  for (int i = 0; i < 3; i++) {
    leftTotal += digitalRead(LEFT_SENSOR);
    rightTotal += digitalRead(RIGHT_SENSOR);
    delayMicroseconds(500);
  }

  bool leftRaw = (leftTotal >= 2), rightRaw = (rightTotal >= 2);
  leftSensorHistory[sampleIndex] = leftRaw;
  rightSensorHistory[sampleIndex] = rightRaw;

  int leftSum = 0, rightSum = 0;
  for (int i = 0; i < SENSOR_SAMPLES; i++) {
    leftSum += leftSensorHistory[i];
    rightSum += rightSensorHistory[i];
  }

  leftOnLine = (leftSum >= (SENSOR_SAMPLES / 2 + 1)) == LINE_DETECTED_STATE;
  rightOnLine = (rightSum >= (SENSOR_SAMPLES / 2 + 1)) == LINE_DETECTED_STATE;

  if (rightOnLine) {
    rightSensorLastDetected = millis();
    rightSensorReliable = true;
  } else if (millis() - rightSensorLastDetected > 5000) {
    rightSensorReliable = false;
  }

  sampleIndex = (sampleIndex + 1) % SENSOR_SAMPLES;
}
```

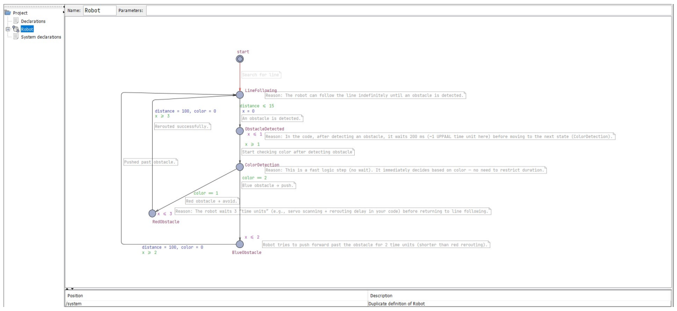
Fig. 20. 5. Sensor Filtering for Stability



```
void gentleTurnRight() {
  digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW); analogWrite(ENA, BASE_SPEED);
  digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW); analogWrite(ENB, BASE_SPEED * 0.3);
}

void moveForwardWithBias(int base, float lBias, float rBias) {
  moveForward(base * lBias, base * rBias);
}
```

Fig. 21. 6. Differential Speed for Smooth Turns

B. A. UPPAAL Simulation



Reason: The robot can follow the line indefinitely until an obstacle is detected.

An obstacle is detected.

Reason: In the code, after detecting an obstacle, it waits 200 ms (~1 UPPAAL time unit here) before moving to the next state (ColorDetection).

Start checking color after detecting obstacle

Reason: This is a fast logic step (no wait). It immediately decides based on color — no need to restrict duration.

Blue obstacle → push.

Red obstacle → avoid.

Reason: The robot waits 3 "time units" (e.g., servo scanning + rerouting delay in your code) before returning to line following.

Robot tries to push forward past the obstacle for 2 time units (shorter than red rerouting).

Rerouted successfully.

Pushed past obstacle.

Position	Description
/system	Duplicate definition of Robot

Fig. 22. UPPAAL Timed Automata Model for Robot Control

C. B. LTspice Simulation

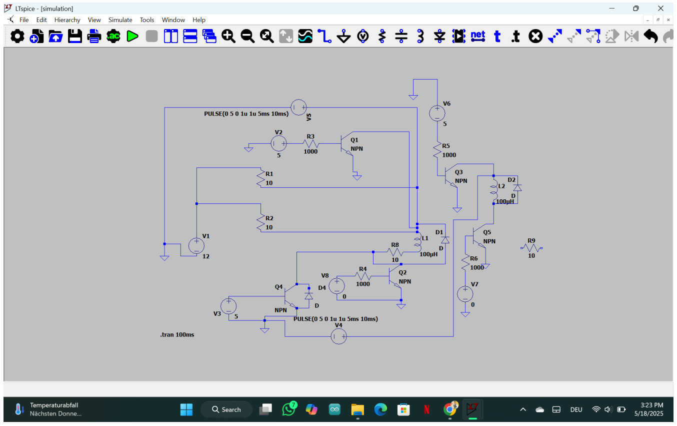


Fig. 23. LTspice Motor Driver Circuit Simulation

V. Git Usage and Collaboration

- **Jubair Salehin Razin**: Simulation, testing, software logic, hardware integration
- **Md Ratul Ahmed Alvi**: Arduino coding, debugging, wiring and logic testing
- **Faysal Ahammed Tonmoy**: Documentation, CAD, simulations, integration
- **Soaib Mahin Ferdous**: Diagram creation, documentation, support tasks

Team Member	Commits
Faysal Ahammed Tonmoy	90
Jubair Salehin Razin	148
Md Ratul Ahmed Alvi	162
Soaib Mahin Ferdous	36
Total	436

TABLE I
GitHub Commits by Team Members

VI. Key Achievements

- Built a reliable, intelligent robot system
- Designed and printed all mechanical parts in SolidWorks
- Simulated logic and hardware in Tinkercad, UPPAAL, LTspice
- Documented system architecture using SysML
- Worked collaboratively using GitHub

VII. Conclusion and Future Work

The robot successfully integrates sensor data and motor control to follow a black line while avoiding obstacles. Future enhancements may include path memory, camera-based vision, Bluetooth control, and autonomous charging.

Acknowledgment

We sincerely thank our professor for their guidance, feedback, and encouragement throughout this project. We also appreciate the support from lab assistants and the facilities provided by Hochschule Hamm-Lippstadt, which were essential to our development and testing. Finally, we acknowledge the teamwork and dedication of all members that made this project a success.

References

[1] Arduino Documentation. https://www.arduino.cc/en/Guide
[2] Tinkercad Simulation. https://www.tinkercad.com
[3] Uppaal Model Checker. http://www.uppaal.org/
[4] LTspice Simulator. https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html

Declaration of Originality

We hereby declare that this report is our own work and that we have acknowledged all sources used.

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