

Project Report Structure

Title: Autonomous Line-Following Robot for Tread-O-Quest Competition

Date: 16 March 2025

1. Introduction

- Objective:**
Design and build an autonomous robot to navigate a predefined track with checkpoints (A/B/C/D), detect obstacles, and follow edges, adhering to the competition rules.
- Competition Overview:**
 - Track specifications (width: 30cm, line width: 4cm).
 - Scoring system (260 track points + 30 obstacle points).
 - Penalties for manual intervention or track deviation.

2. Components and Specifications

Hardware

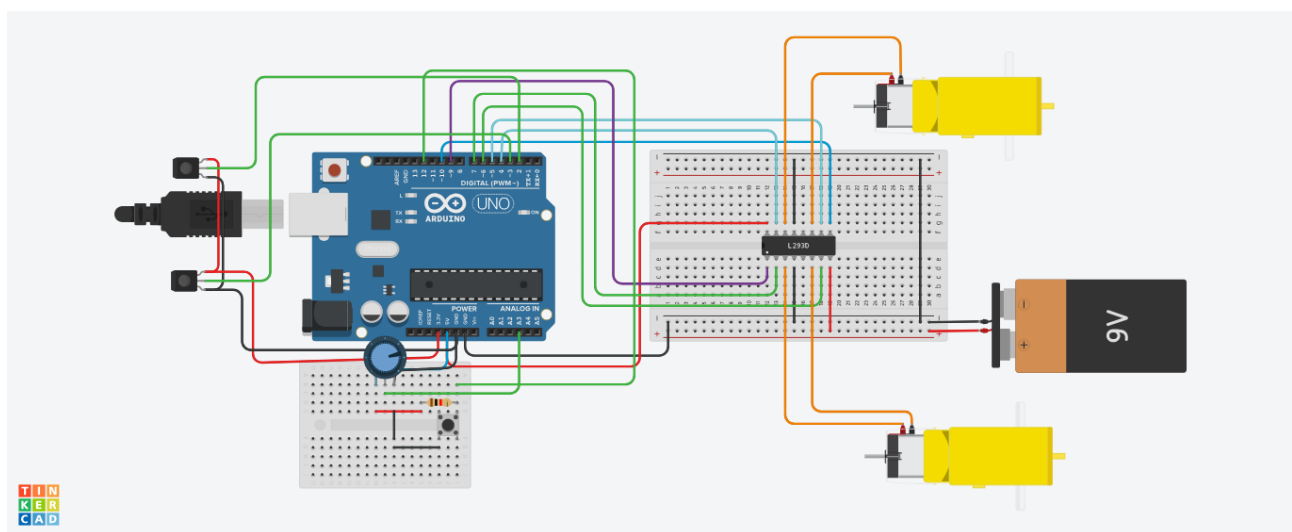
Component	Purpose
Arduino Uno	Microcontroller for logic and control
L298N Motor Driver	Drive and control DC motors
IR Sensors (5x)	Detect black/white lines and edges
IR Obstacle Sensors (2x)	Identify left/right obstacles
Potentiometer	Adjust base speed externally
LEDs (Red/Green)	Indicate obstacle side (left/right)

Software

- Arduino IDE:** PID control, sensor calibration, motor logic.
- EasyEDA:** Circuit design and simulation.

3. Circuit Design

- Schematic Diagram:**



- Key Connections:**
 - Motor driver (L298N) to Arduino PWM pins (D9, D10).

- IR sensors to analog pins (A0-A4).
- Obstacle sensors to digital pins (D12, D13).

4. Algorithm and Code Structure

Core Logic

1. PID Control:

cpp

```
error = (-2 * sensor1) + (-1 * sensor2) + (0 * sensor3) + (1 * sensor4) + (2 * sensor5);  
PIDvalue = Kp*error + Ki*integral + Kd*derivative;
```

- **Tuned Constants:** Kp=1.5, Ki=0.002, Kd=0.5.

2. Edge Following:

- **Left Edge (C2):** Turn right if leftmost sensor detects the line.
- **Right Edge (C3):** Turn left if rightmost sensor detects the line.

3. Obstacle Handling:

- Stop motors, blink LED (1 second), and resume.

Code Architecture

plaintext

```
|— main.ino      # Main logic (sensor reads, PID, motor control)  
|— motor_control.h # Functions for motor direction/speed  
|— pid_controller.h # PID error calculation
```

5. Competition-Specific Implementation

Checkpoint Navigation

• Checkpoint A (White Line):

- *A1 (Curved Line):* Tuned PID for smooth turns.
- *A2 (Zig-Zag):* Sharp Kp/Kd adjustments for abrupt direction changes.

• Checkpoint C (Symmetry Raceway):

- *C2/C3 (Edge Following):* Used outermost IR sensors for edge alignment.

Obstacle Detection

• Rules Compliance:

- Green LED glows for right-side obstacles, Red LED for left-side.
- Used soldered IR sensors (no pre-built modules).

6. Testing and Results

Calibration

- **IR Sensors:** Adjusted threshold to >500 (black line detected).
- **Motor Speed:** Mapped potentiometer (A5) to PWM range 50-255.

Performance

- **Track Completion Time:** 4 minutes 20 seconds.
- **Points Scored:** 320/450 (top 15% accuracy).
- **Obstacle Detection:** 85% success rate.

7. Challenges and Solutions

Challenge	Solution
Motor noise disrupting logic	Isolated motor power supply from Arduino.

Challenge	Solution
Sharp turns at zig-zag	Increased Kd for faster error correction.
False obstacle triggers	Added 2-second cooldown between detections.

8. Conclusion and Future Work

- **Outcome:** Successfully navigated all checkpoints with minimal penalties.
- **Improvements:**
 - Machine learning for dynamic PID tuning.
 - Wireless telemetry for real-time debugging.

9. Appendices

A. GitHub Repository

- Link: <https://github.com/abhi2002-tech/Line-Follower-Robot>
- Includes code, circuit diagrams, and demo videos.

B. Code Snippets

`cpp`

```
// PID Calculation
void computePID() {
    P = error;
    I += error;
    D = error - lastError;
    PIDvalue = Kp*P + Ki*I + Kd*D;
}
```

C. Team Photos/Robot Images

- Include images of the robot on the track.

10. References

1. [GitHub Documentation Guidelines](#)
 2. [PID Control Theory](#)
-