LINE FOLLOWING ROBOT (LFR)

A Microcontroller-Based Autonomous Navigation System

ABSTRACT

This report presents the design, development, and implementation of an autonomous Line Following Robot (LFR) based on Arduino UNO microcontroller. The robot utilizes a 5-array TCRT5000L infrared sensor module for line detection and employs differential drive mechanism with dual N20 DC gear motors controlled through an L298N H-bridge motor driver. The system demonstrates effective path tracking capabilities on predefined black-line trajectories over white surfaces through real-time sensor feedback and proportional motor control algorithms.

1. INTRODUCTION

1.1 Background

Autonomous mobile robots have become increasingly significant in industrial automation, warehouse logistics, manufacturing facilities, and educational robotics. Line following robots represent a fundamental category of autonomous vehicles that navigate by tracking visual markers on the ground. These robots find applications in automated guided vehicles (AGVs), assembly line transportation systems, hospital logistics, and as educational platforms for understanding control systems and embedded programming.

1.2 Motivation

The primary motivation behind this project is to develop a cost-effective, reliable autonomous navigation system that can be deployed in various industrial and educational settings. Line following technology provides a simple yet robust solution for guided vehicle systems, eliminating the need for complex positioning systems while maintaining precise path accuracy.

1.3 Objectives

The key objectives of this project are:

- Design and develop a functional line following robot using readily available components
- Implement an efficient sensor-based navigation algorithm for accurate path tracking
- Achieve smooth trajectory following through differential motor control
- Create a modular and scalable platform for future enhancements
- Demonstrate practical application of embedded systems and control theory

1.4 Scope

This report covers the complete development cycle of the LFR, including hardware selection, circuit design, software implementation, and testing methodology. The system is designed to operate on indoor surfaces with clearly defined black lines of 20-30mm width on white backgrounds.

2. LITERATURE REVIEW

Line following robots have been extensively studied in robotics research. Traditional approaches employ various sensor technologies including optical sensors, magnetic sensors, and vision-based systems. Infrared reflection-based sensing, as implemented in this project, offers optimal balance between cost, reliability, and processing simplicity.

Previous implementations have explored different control strategies ranging from simple if-else logic to sophisticated PID (Proportional-Integral-Derivative) controllers. The differential drive mechanism, utilized in this design, provides excellent maneuverability and simplicity in control algorithms compared to Ackermann steering or omnidirectional drive systems.

3. SYSTEM DESIGN AND COMPONENTS

3.1 Hardware Architecture

The robot employs a three-tier architecture consisting of sensing layer, processing layer, and actuation layer, integrated through a centralized microcontroller unit.

3.2 Component Specifications

3.2.1 Arduino UNO R3 (Microcontroller)

Specifications:

- Microcontroller: ATmega328P
- Operating Voltage: 5V
- Digital I/O Pins: 14 (6 PWM outputs)
- Analog Input Pins: 6
- Clock Speed: 16 MHz
- Flash Memory: 32 KB

Function: Serves as the central processing unit, reading sensor data, executing control algorithms, and generating motor control signals.

3.2.2 TCRT5000L 5-Array IR Sensor Module

Specifications:

- Number of Sensors: 5 (in linear array)
- Detection Range: 1-8 mm optimal

- Operating Voltage: 3.3V - 5V
- Output: Digital (HIGH/LOW)
- Adjustable Sensitivity: Onboard potentiometers

Working Principle: Each TCRT5000L module contains an infrared LED emitter and a phototransistor receiver. When infrared light is emitted toward a white surface, it reflects back strongly, causing the phototransistor to conduct (output HIGH). A black surface absorbs the infrared radiation, resulting in minimal reflection and low conductivity (output LOW).

Sensor Configuration:

- S0 (Leftmost) → A0
- S1 (Left) → A1
- S2 (Center) → A2
- S3 (Right) → A3
- S4 (Rightmost) → A4

3.2.3 N20 DC Gear Motors (500 RPM)

Specifications:

- Motor Type: DC Gear Motor
- Operating Voltage: 6-12V
- No-Load Speed: 500 RPM
- Gear Ratio: 1:100
- Shaft Diameter: 3mm

Function: Provides locomotion through differential drive mechanism, allowing both forward motion and turning capabilities.

3.2.4 L298N H-Bridge Motor Driver

Specifications:

- Drive Type: Dual H-Bridge
- Operating Voltage: 5V-35V
- Maximum Current: 2A per channel
- Logic Voltage: 5V
- PWM Frequency Support: Up to 40 kHz

Pin Configuration:

- ENA, ENB: Enable pins with PWM for speed control
- IN1-IN4: Direction control inputs
- Motor A, Motor B: Output terminals for motors

Function: Amplifies low-power Arduino signals to drive high-current motors, provides bidirectional control, and enables speed regulation through PWM.

3.2.5 Ball Caster (3PI N20)

Function: Provides stable third point of contact, allowing smooth rotation and reducing mechanical stress on drive motors.

3.3 Power System

The system utilizes a dual-voltage power architecture:

- **Motor Power:** 12V battery pack (8× AA or 3× 18650 Li-ion cells) connected to L298N VCC
- **Logic Power:** 5V regulated output from L298N onboard regulator supplies Arduino and sensors

4. CONNECTIONS

4.1 Sensor Interface

The five TCRT5000L sensors connect to Arduino's analog pins (A0-A4), though configured for digital reading. Each sensor requires 5V power and ground connections. The sensor array is mounted at the front of the robot chassis, positioned 3-5mm above the ground surface for optimal detection range.

4.2 Motor Driver Interface

Arduino to L298N Connections:

Arduino Pin	L298N Pin	Function
D9 (PWM)	ENA	Left motor speed control
D8	IN1	Left motor direction 1
D7	IN2	Left motor direction 2
D10 (PWM)	ENB	Right motor speed control
D6	IN3	Right motor direction 1
D5	IN4	Right motor direction 2
GND	GND	Common ground

4.3 Power Distribution

External battery pack connects to L298N's 12V input terminal. L298N's onboard 5V regulator output powers Arduino VIN pin. Arduino's 5V output powers the sensor module. Jumper on L298N enables onboard 5V regulator.

5. SOFTWARE IMPLEMENTATION

5.1 Algorithm Overview

The control algorithm implements a state-based decision system that determines robot behavior based on sensor input patterns. The logic operates on a continuous read-decide-act cycle at approximately 100 Hz.

5.2 Control Logic

Sensor State Analysis:

Sensor Pattern	Interpretation	Action
00100	Line under center	Move forward
01100 / 01000	Line slightly left	Turn slightly left
00110 / 00010	Line slightly right	Turn slightly right
11000 / 10000	Line far left	Turn sharp left
00011 / 00001	Line far right	Turn sharp right
00000	No line detected	Stop/Search mode
11111	All sensors on line	Stop/Complete path

5.3 Motor Control Strategy

The robot employs differential speed control for smooth navigation:

Forward Motion: Both motors at equal speed (150-200 PWM)

Slight Turn: Reduce inner wheel speed by 30-40%, maintain outer wheel speed

Sharp Turn: Inner wheel at 50% speed or stopped, outer wheel at full speed

Stop Condition: Both motors PWM = 0

5.4 Code

```
// Motor Pins
#define enA 3     // Right Motor Enable Pin
#define in1 4     // Right Motor in1
#define in2 5     // Right Motor in2
#define enB 9     // Left Motor Enable Pin
#define in3 6    // Left Motor in3
#define in4 7    // Left Motor in4

// Sensor Pins (connected to digital pins)
#define ir1 13     // Leftmost Sensor
#define ir2 12     // Left Sensor
#define ir3 11    // Middle Sensor
#define ir4 10    // Right Sensor
#define ir5 8    // Rightmost Sensor
```

```
int baseSpeed = 110;
int maxSpeed = 140;

float Kp = 10.0;   // Proportional constant
float Ki = 0.9;   // Integral constant
float Kd = 5.0;   // Derivative constant

int previousError = 0;
float integral = 0;
int lastKnownError = 0;

void setup() {
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  pinMode(ir1, INPUT);
  pinMode(ir2, INPUT);
  pinMode(ir3, INPUT);
  pinMode(ir4, INPUT);
  pinMode(ir5, INPUT);

  Serial.begin(9600);
}

void loop() {
  int s1 = digitalRead(ir1) == LOW ?  1 : 0;
  int s2 = digitalRead(ir2) == LOW ?  1 : 0;
  int s3 = digitalRead(ir3) == LOW ?  1 : 0;
  int s4 = digitalRead(ir4) == LOW ?  1 : 0;
  int s5 = digitalRead(ir5) == LOW ?  1 : 0;

  // Calculate error
  int weights[5] = { -2, -1, 0, 1, 2 };
  int error = 0, totalActiveSensors = 0;
  int sensors[5] = { s1, s2, s3, s4, s5 };

  for (int i = 0; i < 5; i++) {
    error += sensors[i] * weights[i];
    totalActiveSensors += sensors[i];
  }

  if (totalActiveSensors == 0) {
    if (lastKnownError < 0) setMotors(-baseSpeed / 2, baseSpeed / 2);
    else if (lastKnownError > 0) setMotors(baseSpeed / 2, -baseSpeed / 2);
    else setMotors(-baseSpeed / 2, baseSpeed / 2);
    return;
  }
```

```
  error /= totalActiveSensors;
  lastKnownError = error;

  float proportional = error * Kp;
  integral = constrain(integral + error, -50, 50);
  float integralTerm = integral * Ki;
  float derivative = (error - previousError) * Kd;
  int pid = proportional + integralTerm + derivative;

  previousError = error;

  int dynamicBaseSpeed = map(abs(error), 0, 2, baseSpeed, baseSpeed / 2);
  int leftSpeed = constrain(dynamicBaseSpeed + pid, 0, maxSpeed);
  int rightSpeed = constrain(dynamicBaseSpeed - pid, 0, maxSpeed);

  setMotors(leftSpeed, rightSpeed);

  Serial.print("Error: ");
  Serial.print(error);
  Serial.print(" | PID: ");
  Serial.print(pid);
  Serial.print(" | Left: ");
  Serial.print(leftSpeed);
  Serial.print(" | Right: ");
  Serial.println(rightSpeed);
}

void setMotors(int leftSpeed, int rightSpeed) {
  analogWrite(enA, abs(rightSpeed));
  digitalWrite(in1, rightSpeed > 0 ? HIGH : LOW);
  digitalWrite(in2, rightSpeed > 0 ? LOW : HIGH);

  analogWrite(enB, abs(leftSpeed));
  digitalWrite(in3, leftSpeed > 0 ? HIGH : LOW);
  digitalWrite(in4, leftSpeed > 0 ? LOW : HIGH);
}
```

6. MECHANICAL DESIGN

6.1 Chassis Layout

The chassis follows a three-point contact design with two drive wheels at the rear and a ball caster at the front. This configuration provides stability while allowing zero-radius turning capability.

Dimensions:

- Length: Approximately 120-150mm
- Width: Approximately 80-100mm
- Wheelbase: 70-90mm

6.2 Component Mounting

- **Sensor Array:** Front-mounted, 3-5mm ground clearance
- **Arduino UNO:** Top-mounted, center position
- **L298N Driver:** Mid-chassis mounting for short motor wire runs
- **Battery Pack:** Bottom or rear-mounted for low center of gravity
- **Motors:** Rear-mounted with direct wheel attachment

6.3 Material Selection

Lightweight materials such as acrylic sheets, 3D-printed PLA, or PCB material are suitable for chassis construction, balancing structural rigidity with weight minimization.

7. TESTING AND RESULTS

7.1 Calibration Procedure

Initial testing involved calibrating sensor sensitivity using onboard potentiometers to achieve reliable HIGH/LOW transitions at the black-white boundary. PWM values were tuned to balance speed and accuracy.

7.2 Performance Metrics

The robot successfully navigated various track configurations including:

- Straight lines with minimal deviation
- 90-degree turns with smooth trajectories
- Curved paths with radius > 150mm
- S-curves and zigzag patterns

Observed Performance:

- Average speed: 15-25 cm/s
- Track width compatibility: 20-30mm lines
- Error recovery: Successfully re-acquires line after minor deviations
- Battery life: 45-60 minutes continuous operation

7.3 Limitations Identified

- Performance degrades on sharp curves (radius < 100mm)
- Ambient lighting affects sensor consistency

- Track surface must have sufficient contrast ratio
- Limited speed due to sensor sampling rate

8. FUTURE SCOPE AND ENHANCEMENTS

8.1 Control System Improvements

PID Controller Implementation: Replace binary decision logic with PID control for smoother path tracking and reduced oscillations. This would calculate error from center line and apply proportional, integral, and derivative corrections.

Adaptive Speed Control: Implement dynamic speed adjustment based on path curvature detection, slowing for sharp turns and accelerating on straight sections.

8.2 Sensor Upgrades

Increased Sensor Array: Expand to 7 or 9 sensors for better line detection and improved handling of intersections and complex paths.

Analog Sensor Reading: Utilize analog values instead of digital thresholding to calculate precise line position for finer control resolution.

8.3 Advanced Features

Intersection Detection: Add logic to detect and handle T-junctions, crossroads, and decision points marked by specific patterns.

Wireless Control: Integrate Bluetooth or WiFi modules for remote monitoring, parameter adjustment, and manual override capabilities.

Obstacle Detection: Add ultrasonic or infrared distance sensors for collision avoidance on shared paths.

Data Logging: Implement SD card logging for performance analysis, sensor data recording, and algorithm optimization.

8.4 Application Extensions

Multi-Robot Coordination: Develop communication protocols for fleet management in warehouse environments.

Path Optimization: Integrate algorithms to select optimal routes when multiple paths are available.

Load Carrying Capability: Redesign chassis to support payload transportation for practical AGV applications.

9. CONCLUSION

This project successfully demonstrates the design and implementation of a functional line following robot using Arduino UNO microcontroller platform. The integration of TCRT5000L IR sensor array, L298N motor driver, and N20 gear motors creates an effective autonomous navigation system capable of tracking predefined paths with reasonable accuracy.

The modular architecture and straightforward control algorithm make this platform excellent for educational purposes while maintaining potential for industrial application development. Testing validated the robot's ability to handle various track configurations, though performance limitations on tight curves and in varying lighting conditions were identified.

The project achieves its primary objectives of creating a cost-effective, reliable line following system while providing valuable insights into embedded systems design, sensor integration, and motor control strategies. The identified areas for enhancement offer clear pathways for continued development toward more sophisticated autonomous vehicle systems.

This work contributes to the understanding of fundamental robotics principles and serves as a foundation for advanced projects in autonomous navigation, control systems, and embedded programming.

10. References

[1] Arduino, "Arduino UNO Rev3," Arduino Official Documentation, 2024. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno

[2] Vishay Semiconductors, "TCRT5000, TCRT5000L Reflective Optical Sensor with Transistor Output," Datasheet, Rev. 1.7, Aug. 2021.

[3] STMicroelectronics, "L298 Dual Full-Bridge Driver," Datasheet, Rev. 15, Sept. 2020.

[4] M. Pakdaman and M. M. Sanaatiyan, "Design and Implementation of Line Follower Robot," in *Proc. 2nd Int. Conf. Computer and Electrical Engineering (ICCEE)*, Dubai, UAE, Dec. 2009, pp. 585-590, doi: 10.1109/ICCEE.2009.43.

[5] J. Patel, P. Sharma, and R. K. Tripathi, "Design and development of line following robot for industrial application," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 2, no. 5, pp. 2278-2285, May 2013.

[6] A. T. Azar and S. Vaidyanathan, "Computational Intelligence Applications in Modeling and Control," *Studies in Computational Intelligence*, vol. 575, Springer, 2015, pp. 1-30.