Report on Line Following Robot Design and Development

YEAR 1 – SEMESTER 1 – EE1010 – Mini Project

Prepared by:

Shadow Trackers(Team 33)

March , 2025

Contents

Abstract

This report details the development of a line-following robot project using a single design to evaluate trade-offs between cost, complexity, and performance. The design integrates an Arduino Nano microcontroller, high-precision N20 motors, and a 8-channel QTR-8RC IR sensor array. A PID control algorithm was implemented to ensure accurate line tracking and stable navigation. The results show that the chosen design provides a balance between precision and stability on complex paths while offering a cost-effective solution with moderate tracking accuracy. These findings highlight how hardware choices impact line-following performance, offering insights for future improvements in affordable autonomous robotics.

Acknowledgements

We would like to extend our heartfelt gratitude to Mr. Janaka Circuit, Project Coordinator, Department of Electrical and Electronics Engineering, for his invaluable guidance, support, and insightful suggestions throughout our project. His assistance at each stage greatly contributed to the success of this project and report.

Our sincere appreciation also goes to Mr. Tharindu Weerakoon, Project Supervisor, Department of Electrical and Electronics Engineering, for dedicating his time and expertise to support our efforts. His encouragement and assistance have been instrumental in the completion of our work.

We are also indebted to the authors of numerous articles and references that provided foundational knowledge and insights essential to our project.

Finally, we wish to express our thanks to our friends for their constant support and assistance in various aspects of the project, which played a vital role in the completion of this report.

Motivations and Objectives

1. To build a robust and accurate line-following robot that can navigate a defined path using sensor feedback and motor adjustments.
2. To implement and tune a PID (Proportional-Integral-Derivative) control algorithm, allowing the robot to handle various line-following scenarios with precision, including curves and sharp turns.
3. To evaluate the trade-offs between complexity, cost, and functionality, providing insights for future design improvements and applications.

Methodology

In this project, IR sensors are used for line detection, with data processed by an ESP32 microcontroller to control motor speed and direction. A PID control algorithm enables the robot to smoothly adjust to the line path and correct deviations. The components, including sensors, motor drivers, the microcontroller, and the power source, were selected to optimize performance and stability. The design prioritizes precise control and efficiency, ensuring accurate line tracking, stable turns, and optimal power consumption. Testing provides valuable insights into the effectiveness of the algorithm in maintaining speed, stability, and overall performance.

Chapter 1
Line Following Robot

1.1 Component Selection

- **Sensors: QTR-8RC Channel IR Sensor Array**
  The QTR-8RC is an IR sensor array with eight channels for line detection. It provides digital output signals, making it ideal for line-following robots. It detects surface reflectivity, distinguishing between light and dark areas for precise tracking.

Figure 1: QTR-8RC Channel IR Sensor Array

- **Motors: N20 200rpm DC Motors**
  Chosen for its reliability and precision in controlling and providing smooth and accurate motor responses.

Figure 2: N20 200rpm DC Motor

- **Motor Driver: DRV 8833 Motor driver**
  This motor driver is more used to control the N20 motors. It is compatible with Arduino Nano board and can handle the required voltage and current for driving the motors.

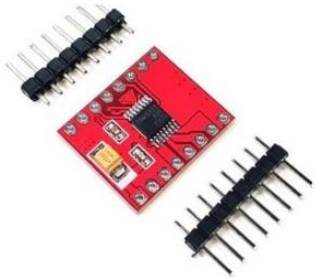


Figure 3: DRV 8833 Motor driver

- **Power Supply: rechargeable battery 3.7 V**
  A 3.7V rechargeable battery is lightweight, long-lasting, fast-charging, and cost-effective, making it ideal for portable electronics.so this battery was selected.

Figure4:3.7V Rechargeable Battery

Figure 5:Side view of the robot

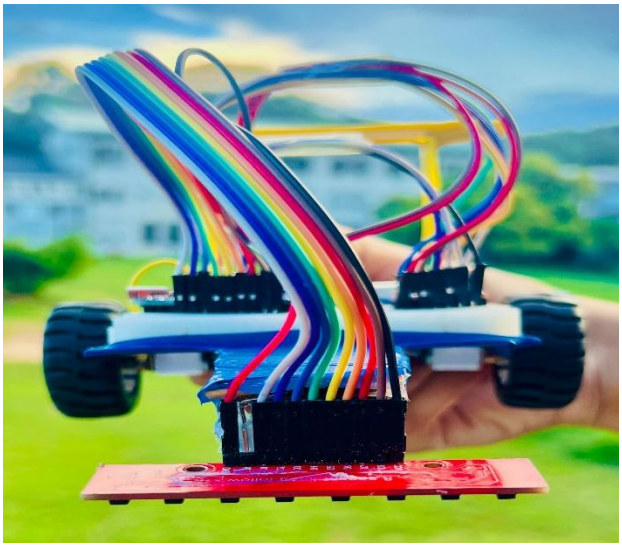


Figure 6:Front view of the robot

1.2 Circuit Design and Schematic

The Arduino Nano board was integrated into the circuit, along with the QTR-8RC channel IR sensor array, motor driver, and battery pack. The schematic ensured that all components were connected according to their pinouts and functionality. Adjustments were made to account for the Arduino Nano's pinout and the requirements of the QTR-8RC sensor array.

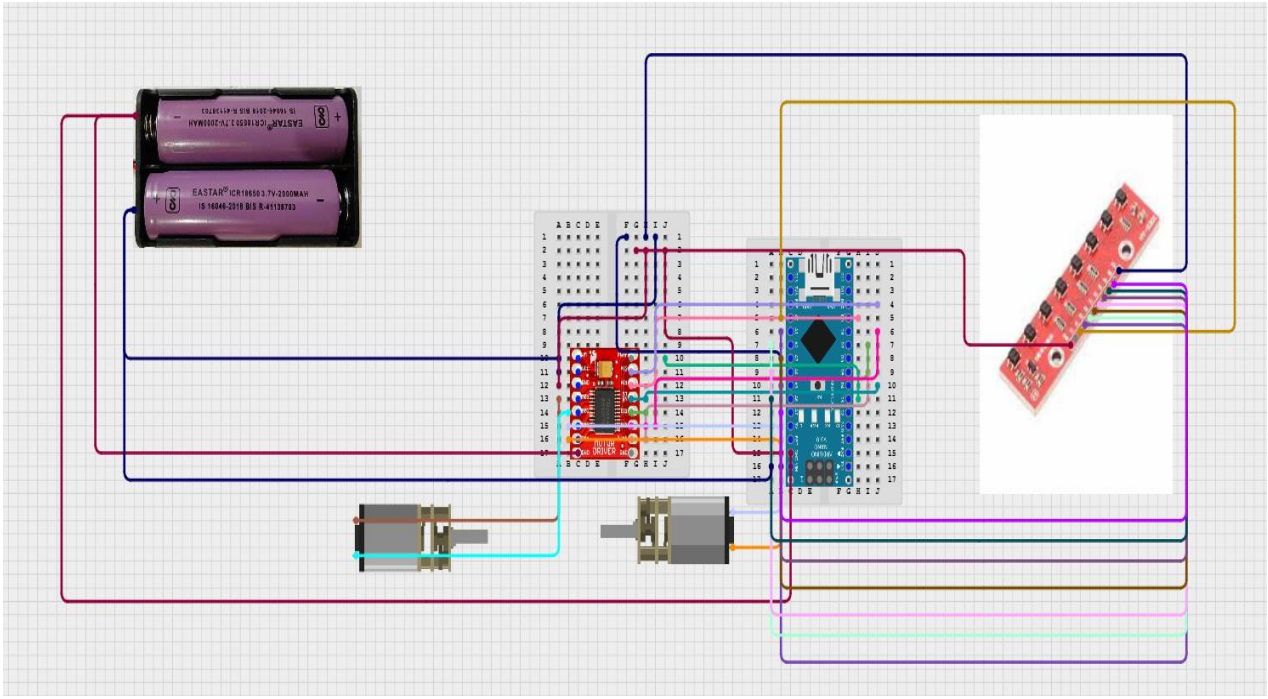
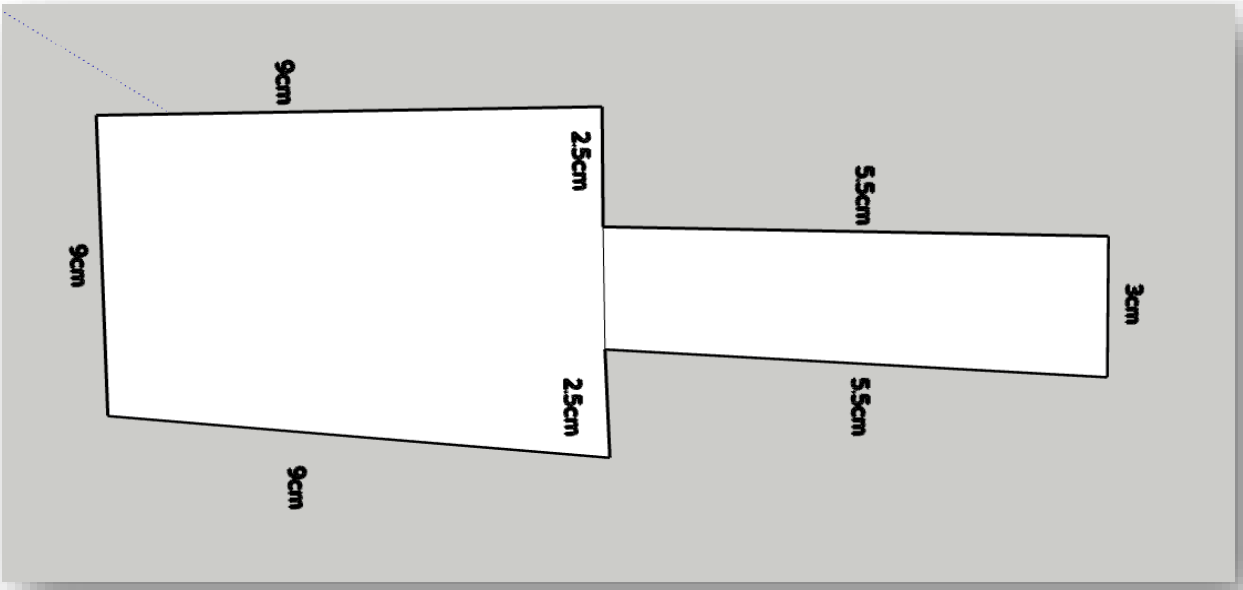


Figure7:Cicuit design of the robot

1.3 Design Process

1. **Chassis Selection and Size Considerations:** A smaller chassis was chosen to fit the small wheels attached to metal gear motors. A larger chassis would prevent proper wheel functioning, as the wheels would not align well with a larger frame.

Figure 8:Chasis design



2. **Component Placement:** Components were placed to be easily re movable, allowing replacements or adjustments as needed.

3. **Battery Pack Attachment:** The battery pack is attached by glue gun for get extra stability.

4. **Sensor Array Positioning:** The IR sensor array is mounted on the front of the chassis

5. **Component Placement:** Components were placed to be easily re movable, allowing replacements or adjustments as needed.

1.4 PID Algorithm Implementation and Tuning

The PID (Proportional-Integral-Derivative) algorithm is essential for precise and stable line-following. Here's how each component of the PID algorithm is implemented:

- Proportional (P) Control: The proportional term creates a correc tion proportional to the error (distance from the line). Adjusting the proportional gain Kp prevents oscillations and ensures responsiveness.
- Integral (I) Control: The integral term addresses accumulated error over time, correcting any drift. Adjusting Ki helps maintain steady tracking over long distances.
- Derivative (D) Control: The derivative term anticipates changes, providing smoother turns. Adjusting Kd prevents overshoot and allows for smooth cornering.

The PID output C(t) is calculated as:

$$C(t) = K_p \times \text{Error} + K_i \times \int (\text{Error}) dt + K_d \times d(\text{Error})/dt$$

1.5 Preview of the Code

the PID algorithm for the line-following robot, utilizing the Arduino nano board and a custom setup. Below is the key portion of the code robot, which controls the motors based on sensor inputs.

```
#include <BeeLineSensorPro.h>

#define M1 4
#define M1pwm 5
#define M2 7
#define M2pwm 6

BeeLineSensorPro sensor =
BeeLineSensorPro((unsigned char[]) {
  A0, A1, A2, A3, A4, A5, A6, A7
}, LINE_BLACK);

void mdrive(int m1, int m2); // Function
declaration

void setup() {
  pinMode(M1, OUTPUT);
  pinMode(M1pwm, OUTPUT);
  pinMode(M2, OUTPUT);
  pinMode(M2pwm, OUTPUT);
  Serial.begin(115200);
  delay(2000);
  for (int i=0;i<350;i++){
    sensor.calibrate();
```

```
    mdrive(120,-120);

  }
  mdrive(0,0);
  delay(2000);
}

float kP=0.18;
float kD=1.4;
int last_value;

void loop() {
  int err=sensor.readSensor();
  for(int i=0;i<8;i++){
    Serial.print(sensor.values[i]);
    Serial.print('\t');
  }
  Serial.println(err);
  int m1=255;
  int m2=255;

  int diff=err*kP + (err - last_value)*kD;

  last_value=err;
  mdrive(m1+diff,m2-diff);
}
void mdrive(int m1, int m2) {
  // Motor 1 control
  if (m1 > 0) {
    if (m1 > 255) {
      m1 = 255;
    }
    // Forward
```

```
    digitalWrite(M1, HIGH);
    analogWrite(M1pwm, 255 - m1);
  } else {
    if (m1 < -255) {
      m1 = -255;
    }
    // Backward
    digitalWrite(M1, LOW);
    analogWrite(M1pwm, -m1);
  }

  // Motor 2 control
  if (m2 > 0) {
    if (m2 > 255) {
      m2 = 255;
    }
    // Forward
    digitalWrite(M2, HIGH);
    analogWrite(M2pwm, 255 - m2);
  } else {
    if (m2 < -255) {
      m2 = -255;
    }
    // Backward
    digitalWrite(M2, LOW);
    analogWrite(M2pwm, -m2);
  }
}
```

1.6 Challenges and Modifications

• **Sensor Integration Challenges**:
The initial sensor configuration faced limitations due to compatibility issues with the existing library, which restricted the functionality of multiple sensors operating simultaneously. As a result, an alternative sensor solution was adopted to ensure seamless operation and improved performance.

• **Motor Control Instability**:
During the initial testing phase, the motor control system exhibited inconsistent behavior. This was traced back to imperfections in the soldering joints, which were subsequently resolved by refining the soldering process and incorporating heat-shrink sleeves for enhanced durability and stability.

• **Connectivity Issues**:
The system encountered intermittent malfunctions in both motor and sensor operations, primarily caused by inadequate connections and loose wiring. These issues were mitigated by redesigning the layout to optimize component placement and ensuring all connections were securely fastened to prevent disruptions.

Project Timeline

- Initial Planning (3rd – 10th January 2025) - We discussed how to buy the components through our WhatsApp group - Two team members went and brought the components form DUINO - Discussed the initial plan during the session on Project Discussion Group

- Initial Design (16th January – 5th February 2025) - We discussed the initial design and made attempts to build the robot and made several changes to the code during this time

- Testing and Simulation (11th – 13th February 2025) - We tested the robot for the first time and tested

- Final Tuning and Component Adjustments (16th March 2025) - Tested the robot and tuned for the final round on the day before “The RACE”

Conclusion

In summary, this project involved the design, development, and comparative evaluation of two distinct line-following robot prototypes, each employing a different strategy to balance cost, performance, and complexity. The first design, equipped with advanced components and enhanced processing capabilities, enabled the implementation and optimization of a sophisticated control system, resulting in exceptional accuracy and stability when navigating intricate paths. The second design, while simpler and more cost-effective, achieved the core functionality of line following and served as a reliable platform for testing and experimentation.

Testing both prototypes provided critical insights into how hardware selection influences performance, stability, and reliability. The advanced design underscored the advantages of high-quality components for handling complex trajectories, while the simpler design demonstrated the viability of creating functional, low-cost autonomous systems using basic elements.

The integration of a control algorithm played a pivotal role in both designs, facilitating real-time adjustments and error correction, which significantly enhanced the robots' ability to navigate sharp turns and curved paths. The process of fine-tuning the algorithm's parameters highlighted the importance of achieving a balanced control system for smooth and responsive operation. Future enhancements could involve the integration of more advanced sensor technologies, improvements to the structural design for greater stability, and the exploration of adaptive control methods, such as machine learning, for dynamic path correction. This project not only deepened our understanding of autonomous navigation systems but also

established a foundation for further research into cost-effective robotics solutions with practical applications.

References

1. Arduino. (n.d.). *PID control library*. [Online]. Available: https://www.arduino.cc/reference/en/libraries/pid/

2. Robotics Back-End. (2021). *How to build a line-following robot*. [Online]. Available: https://roboticsbackend.com/how-to-build-a-line-following-robot/

3. Electronics Hub. (2022). *Line follower robot using Arduino*. [Online]. Available: https://www.electronicshub.org/line-follower-robot-using-arduino/

4. Instructables. (2020). *Build a line-following robot with PID control*. [Online]. Available: https://www.instructables.com/Build-a-Line-Following-Robot-with-PID-Control/

5. SparkFun Electronics. (n.d.). *Line sensor array*. [Online]. Available: https://www.sparkfun.com/products/9453

6. IEEE Spectrum. (2021). *How to build a robot: Line-following robot tutorial*. [Online]. Available: https://spectrum.ieee.org/how-to-build-a-robot-line-following-robot-tutorial

Team Members

– **Team Leader:** P.A.T. Ninduwara - E/23/246

– **Team Members:**

1. K.N.S. Perera - E/23/268
2. P.M.A.A. Navodya - E/23/239
3. T.I.S. Peiris - E/23/262
4. A. Rajeeth - E/23/289
5. T.P.L. Nambuge- E/23/234
6. W.A.I. Prathibha -E/23/277
7. S. Paveenan – E/23/256