“BADERIA GLOBAL INSTITUTE OF ENGINEERING & MANAGEMENT”

A

MINOR PROJECT-II REPORT

ON

Line Following Robot Using PID Algorithm

Submitted for the partial fulfillment of
The requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & COMMUNICATION ENGINEERING

Guided by

MR. Vijay Bhatt

DEPARTMENT OF ELECTRONICS
&
COMMUNICATION ENGINEERING

Submitted by

Vivek Vishwakarma - Roll No. 0225EC211134
Yuvraj Singh Lodhi - Roll No. 0225EC211138
Tarun Bisen - Roll No. 0225EC211127
Sohit Kushwaha - Roll No. 0225EC211124
Ankit Vishwakarma - Roll No. 0225EC211018
Yaman Kureel - Roll No. 0225EC211135

Department of Electronics & Communication Engineering
BADERIA GLOBAL INSTITUTE OF ENGINEERING & MANAGEMENT
Paten by pass road Jabalpur (M.P.)

“BADERIA GLOBAL INSTITUTE OF ENGINEERING & MANAGEMENT”

DECLARATION

I/We, hereby declare that this report entitled “**Line Following Robot Using PID Algorithm**”
is an authentic record of our own work carried out toward the partial fulfillment for the award of degree of Bachelor of Technology in Electronics & Communication Engineering, under the guidance of **MR. Vijay Bhatt** (Asst. Prof. Department of EC BGIOEM Jabalpur) during Jan to May 2024.

Vivek Vishwakarma - Roll No. 0225EC211134
Yuvraj Singh Lodhi - Roll No. 0225EC211138
Tarun Bisen - Roll No. 0225EC211127
Sohit Kushwaha - Roll No. 0225EC211124
Ankit Vishwakarma - Roll No. 0225EC211018
Yaman Kureel - Roll No. 0225EC211135

BADERIA GLOBAL INSTITUTE OF ENGINEERING &

MANAGEMENT

<u>CERTIFICATE</u>

This is certified that "**Vivek Vishwakarma, Yuvraj Singh Lodhi, Tarun Bisen, Sohit Kushwaha, Ankit Vishwakarma, Yaman Kureel**" has completed a Major Project on "**Line Following Robot Using PID Algorithm**" for the partial fulfillment for award of degree of Bachelor of Technology in Electronics & Communication Engineering.

Date: 05-07-2024

MR. Vijay Bhatt
Asst. Prof. (EC Deptt.)

Dr. Prateek Mishra
H.O.D. (EC Deptt.)
BGIOEM Jabalpur
Date: _05-07-2024

Dr. Rajeev Khatri
Director
BGIOEMJabalpur
Date: _05-07-2024____

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL

CERTIFICATE

This is to certify that

“**Vivek Vishwakarma, Yuvraj Singh Lodhi, Tarun Bisen, Sohit Kushwaha, Ankit Vishwakarma, Yaman Kureel**”
Has successfully completed his/her Major project- II on

Line Following Robot Using PID Algorithm

During year 2024 in the partial fulfillment for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & COMMUNICATION ENGINEERING

INTERNAL EXAMINER **EXTERNAL EXAMINAR**

ACKNOWLEDGEMENT

I wanted to express my appreciation to Mr. Saurabh Baderia, Chairman, and Dr. Rajiv Khatri, Director, for supporting and giving resources for institutional research.
I owe a great deal of thanks to Dr. Prateek Mishra, head of the EC department, for providing the necessary resources for this effort. I would like to convey my sincere admiration and thanks to my advisor and mentor, Mr. Vijay Bhatt, for his inspirational leadership and unwavering support during the completion of my B-Tech dissertation. I also want to thank him for his thoughtful remarks and recommendations, which helped me keep improving my comprehension.
I'm also grateful to the teachers and staff of the EC department for their invaluable advice they gave me when I was working on my project. Additionally, I want to express my gratitude to my friends who have always served as an inspiration to me in my work. I'd want to end by expressing my sincere thanks to my parents. My commitment to work would not have been feasible without their desires, moral support, and encouragement. Above all, I give thanks to God Almighty, who gave me the perseverance and strength to finish the task.

Vivek Vishwakarma - Roll No. 0225EC211134
Yuvraj Singh Lodhi - Roll No. 0225EC211138
Tarun Bisen - Roll No. 0225EC211127
Sohit Kushwaha - Roll No. 0225EC211124
Ankit Vishwakarma - Roll No. 0225EC211018
Yaman Kureel - Roll No. 0225EC211135

Chapter 1
INTRODUCTION

1.1 Introduction

The rapid advancements in robotics and automation have led to the development of various autonomous systems, one of which is the line-following robot. A line-following robot is an autonomous mobile robot designed to follow a predefined path, usually represented by a line on the floor. This capability makes it highly useful in numerous applications such as manufacturing, logistics, and service industries where automated transportation and material handling are required.

The performance and efficiency of a line-following robot significantly depend on its control system. One of the most effective control algorithms used in such systems is the Proportional-Integral-Derivative (PID) control algorithm. The PID control algorithm is renowned for its ability to provide precise and stable control by continuously adjusting the control input based on the error between the desired and actual positions of the robot.

The primary objective of this project is to design and implement a line-following robot that leverages the PID control algorithm to achieve accurate and stable navigation along a predefined path. The PID algorithm adjusts the robot's steering and speed by minimizing the error detected by the sensors, thereby ensuring smooth and reliable movement along the line.

In this project, we will explore the fundamental components required to build the line-following robot, including the microcontroller, motors, sensors, and chassis. We will then delve into the principles of the PID control algorithm and how it is applied to the robot's control system. The implementation will involve programming the microcontroller to read sensor data, calculate the error, and adjust the motor speeds accordingly using the PID control equation.

Through systematic testing and validation, we aim to demonstrate the effectiveness of the PID algorithm in maintaining the robot's adherence to the line, even in challenging scenarios such as curves and intersections. The successful implementation of this project will not only highlight the capabilities of PID control in robotics but also provide a foundation for further enhancements and applications in autonomous navigation systems.

This introduction sets the stage for a comprehensive exploration of the design, implementation, and testing of a line-following robot using the PID algorithm, showcasing its potential and practical applications in real-world scenarios.

Chapter 2
PROJECT OBJECTIVES

2.1 Project Objectives

- Design a robot capable of following a line autonomously

Designing a robot capable of autonomously following a line involves a combination of hardware and software components. The goal is to create a system that can detect a line on the floor and adjust its movements to stay on that line using a PID control algorithm. The design process can be broken down into several key steps: selecting hardware components, setting up the physical robot, implementing the PID control algorithm, and integrating the software and hardware to ensure smooth operation.

- Implement a PID control algorithm to improve the robot's line-following accuracy and stability.

Implementing a PID control algorithm to improve the robot's line-following accuracy and stability involves tuning the Proportional-Integral-Derivative parameters to dynamically adjust the robot's motor speeds based on the error signal from the line detection sensors. The proportional component corrects the robot's position by providing an output proportional to the current error, the integral component accumulates past errors to address any persistent biases, and the derivative component predicts future errors by considering the rate of change of the error, thereby damping oscillations. By continuously adjusting these parameters, the PID algorithm ensures that the robot can swiftly and smoothly correct its path, maintaining a consistent and accurate trajectory along the line even in the presence of curves, intersections, or varying line thickness. This results in enhanced stability and precision in the robot's line-following behavior, making it more reliable and effective in navigating predefined paths.

- Test the robot's performance in various scenarios and environments.

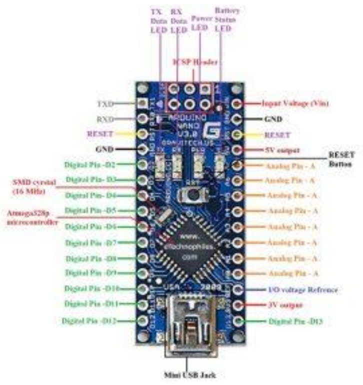
Testing the robot's performance in various scenarios and environments is essential to ensure its robustness and reliability. The robot should be evaluated on straight lines to measure its ability to maintain a steady path without deviation. Curved paths present a greater challenge, requiring the PID algorithm to dynamically adjust motor speeds to navigate turns smoothly. Intersections and forks test the robot's decision-making capabilities and its ability to choose and follow the correct path. Additionally, varying line thickness and colors, as well as different surface textures, should be tested to ensure the sensors and control algorithm can adapt to different conditions. Testing in environments with potential obstacles will evaluate the robot's response to unexpected changes in the path. Comprehensive testing across these diverse scenarios will help identify any weaknesses in the system and provide opportunities to fine-tune the PID parameters for optimal performance, ensuring the robot operates effectively in real-world applications.

Chapter 3
SYSTEM COMPONENTS

3.1 Hardware Components

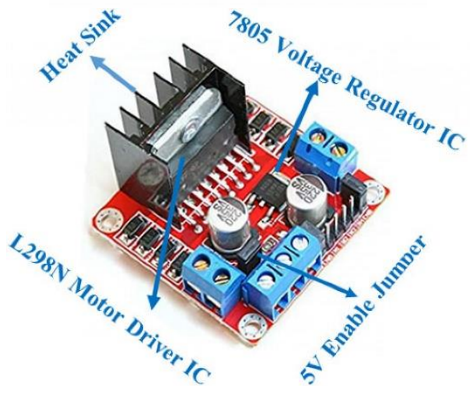
1 Microcontroller

- **Arduino Nano**: A widely used microcontroller that offers sufficient processing power, multiple I/O pins, and a user-friendly programming environment.



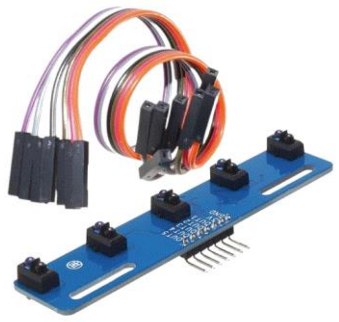
2 Motors and Motor Drivers

- **DC Motors**: Provide the necessary propulsion for the robot. The speed and direction of these motors can be controlled via the motor driver.
- **Motor Driver (L298N)**: Allows control of the motors by the microcontroller, enabling forward and reverse motion as well as speed control.



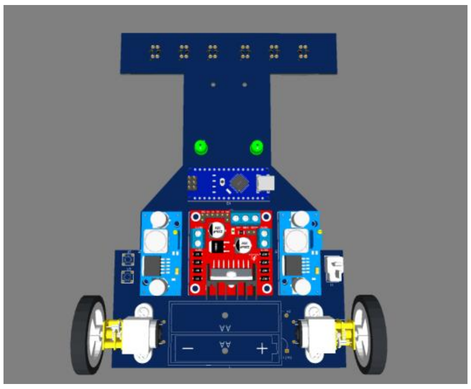
3 Sensors

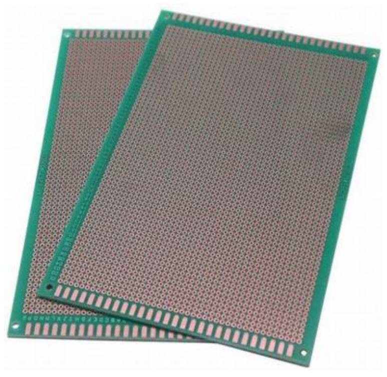
- **Infrared (IR) Sensors**: Detect the presence of a line by differentiating between the color of the line (typically black) and the background (typically white).



4 Chassis

- **Robot Base**: The physical structure that holds all components together, including slots for mounting the motors, sensors, and microcontroller.





5 Power Supply

- **Battery Pack**: Provides the necessary power to the microcontroller and motors.

- **Buck Converter:** A Buck Converter, also known as a step-down converter, is a type of DC-DC converter that efficiently reduces the input voltage to a lower output voltage.

- **Boost Converter:** A Boost Converter, also known as a step-up converter, is a type of DC-DC converter that increases (boosts) the input voltage to a higher output voltage.

3.2 Software Components

- **Arduino IDE**: For programming the microcontroller

1.6.4
ARDUINO
AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits

- **PID Library**: To implement the PID control algorithm efficiently on an Arduino Nano for a line-following robot, the Arduino PID Library can be utilized. This library simplifies the process of implementing PID control by providing a standardized framework for calculating and applying the PID control loop. Here's a step-by-step guide on using the PID library in your project.

Chapter 4

DESIGN AND IMPLEMENTATION

4.1 Hardware Setup

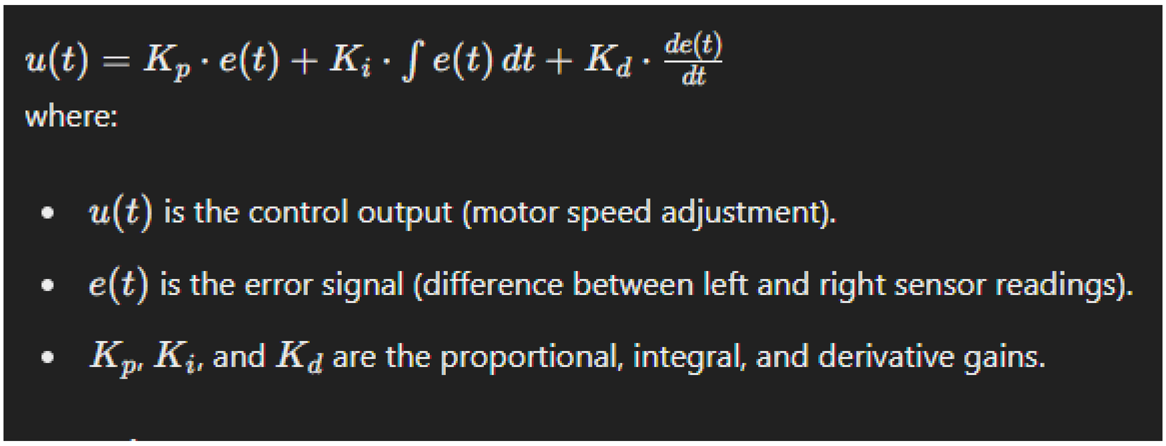
1. **Assemble the Chassis**: Attach the wheels and DC motors to the robot chassis.
2. **Place the IR Sensors**: Install the IR sensors at the front of the robot to detect the line.
3. **Connect the Microcontroller**: Wire the microcontroller to the motor driver and the IR sensors.
4. **Power Supply**: Connect the battery pack to power the microcontroller and motors.

4.2 PID Control Algorithm

The PID control algorithm adjusts the speed and direction of the motors based on the error signal from the IR sensors. The error is the difference between the desired position of the line (centered between the sensors) and the actual position.

- **Proportional (P) Control**: Corrects the error in proportion to its magnitude.
- **Integral (I) Control**: Accounts for the accumulation of past errors.
- **Derivative (D) Control**: Predicts future errors based on the rate of change.

The PID control equation is:



$$u(t) = K_p \cdot e(t) + K_i \cdot \int e(t)\,dt + K_d \cdot \frac{de(t)}{dt}$$

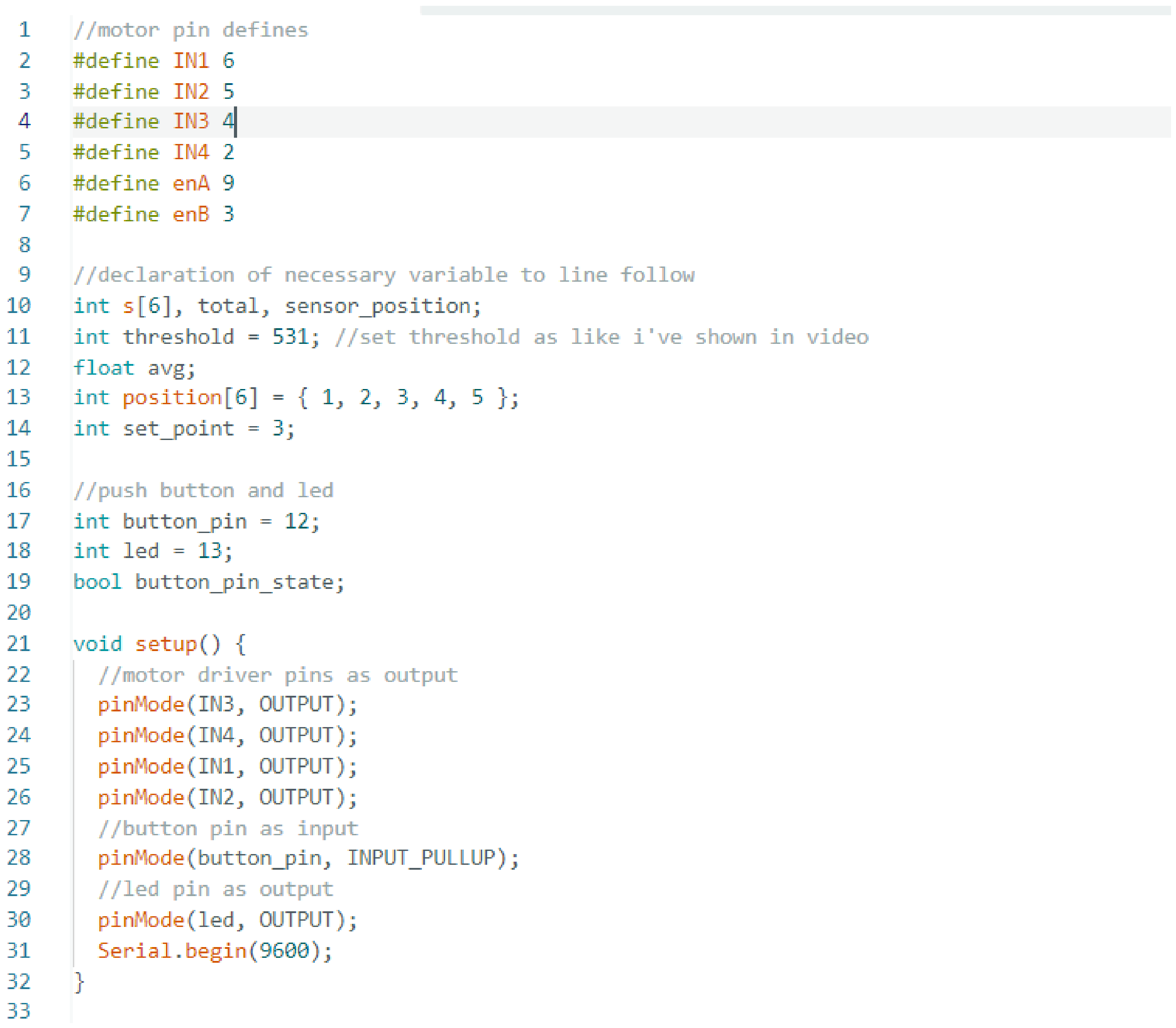
where:

- $u(t)$ is the control output (motor speed adjustment).
- $e(t)$ is the error signal (difference between left and right sensor readings).
- K_p, K_i, and K_d are the proportional, integral, and derivative gains.

4.3 Software Implementation

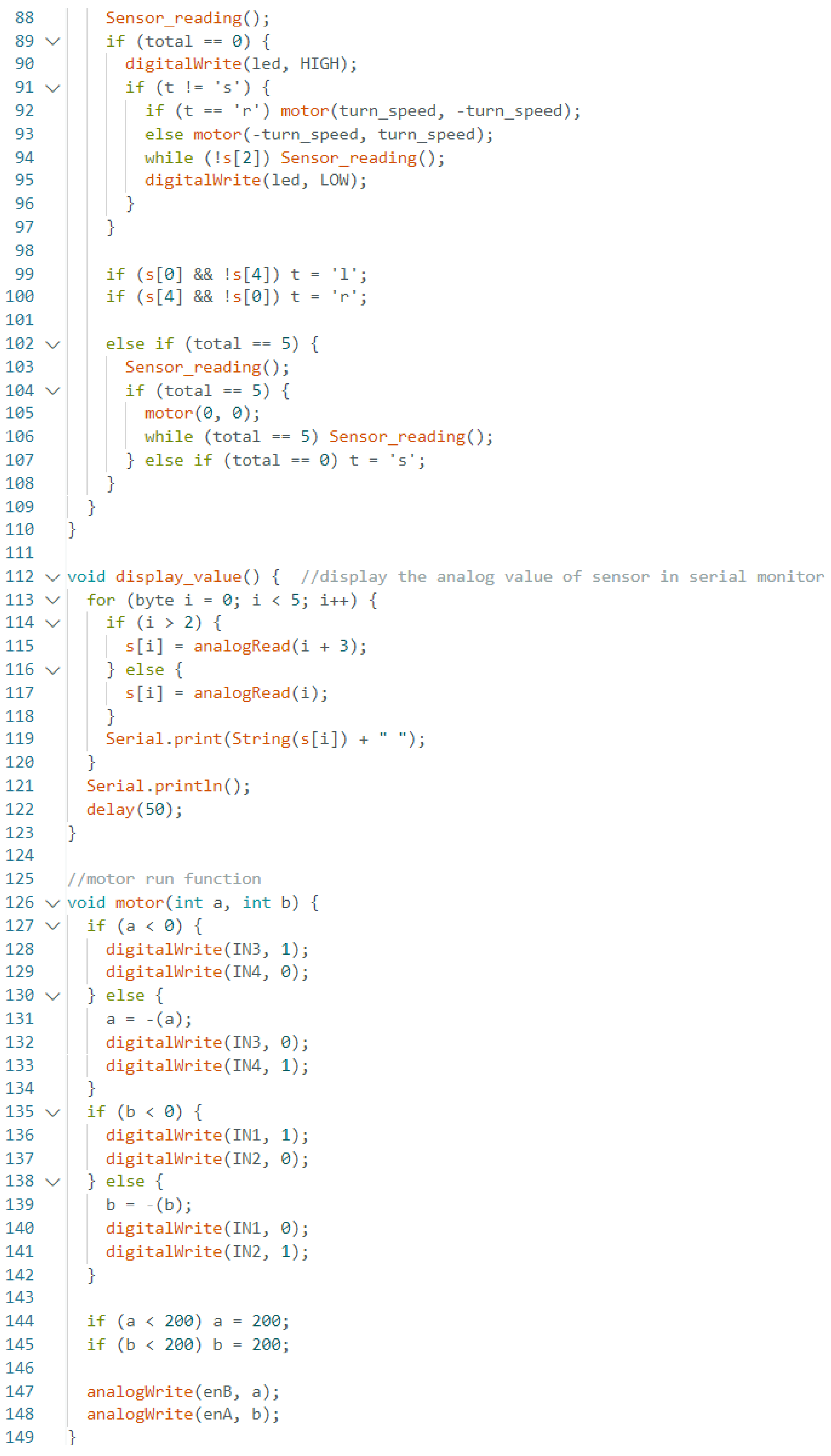
1. **Initialize the Arduino IDE**: Install the necessary libraries and set up the development environment.
2. **Read Sensor Values**: Continuously read values from the IR sensors to detect the line.
3. **Calculate Error**: Compute the error signal based on the difference in sensor readings.
4. **Compute PID Output**: Use the PID algorithm to calculate the control output.
5. **Control Motors**: Adjust the motor speeds based on the PID output to keep the robot on the line.

The Arduino code for the PID control is as follows:



```
//motor pin defines
#define IN1 6
#define IN2 5
#define IN3 4
#define IN4 2
#define enA 9
#define enB 3

//declaration of necessary variable to line follow
int s[6], total, sensor_position;
int threshold = 531; //set threshold as like i've shown in video
float avg;
int position[6] = { 1, 2, 3, 4, 5 };
int set_point = 3;

//push button and led
int button_pin = 12;
int led = 13;
bool button_pin_state;

void setup() {
  //motor driver pins as output
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  //button pin as input
  pinMode(button_pin, INPUT_PULLUP);
  //led pin as output
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

```

```
void loop() {
  button_pin_state = digitalRead(button_pin);
  display_value();

  //when button is pressed then the robot will start to follow lines.
  while (button_pin_state != 1) {
    //blink led
    for (int i = 0; i < 3; ++i) {
      digitalWrite(led, HIGH);
      delay(100);
      digitalWrite(led, LOW);
      delay(100);
    }
    PID_LINE_FOLLOW();  //line follow using PID_Value
  }
}

void Sensor_reading() {
  sensor_position = 0;
  total = 0;
  for (byte i = 0; i < 5; i++) {  //snesor data read from A0, A1, A2, A6, A7
    if (i < 2) {
      s[i] = analogRead(i + 3);
    } else {
      s[i] = analogRead(i);
    }
    if (s[i] < threshold) s[i] = 1;  //analog value to digital conversion
    else s[i] = 0;
    sensor_position += s[i] * position[i];
    total += s[i];
  }
  if (total) avg = sensor_position / total;  //average value
}

void PID_LINE_FOLLOW() {

  int kp = 50, kd = 500, PID_Value, P, D;
  float error, previous_error;
  int base_speed = 200, left_motor_speed, right_motor_speed, turn_speed = 100;
  char t;

  while (1) {
    Sensor_reading();
    error = set_point - avg;
    D = kd * (error - previous_error);
    P = error * kp;
    PID_Value = P + D;
    previous_error = error;

    //adjusting motor speed to keep the robot in line
    right_motor_speed = base_speed - PID_Value;  //right motor speed
    left_motor_speed = base_speed + PID_Value;  //left motor speed
    motor(left_motor_speed, right_motor_speed);              //when robot in straight position
```

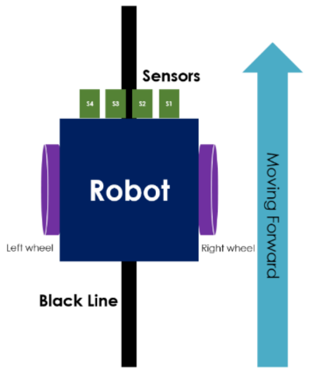


```
    Sensor_reading();
    if (total == 0) {
      digitalWrite(led, HIGH);
      if (t != 's') {
        if (t == 'r') motor(turn_speed, -turn_speed);
        else motor(-turn_speed, turn_speed);
        while (!s[2]) Sensor_reading();
        digitalWrite(led, LOW);
      }
    }

    if (s[0] && !s[4]) t = 'l';
    if (s[4] && !s[0]) t = 'r';

    else if (total == 5) {
      Sensor_reading();
      if (total == 5) {
        motor(0, 0);
        while (total == 5) Sensor_reading();
      } else if (total == 0) t = 's';
    }
  }
}

void display_value() {  //display the analog value of sensor in serial monitor
  for (byte i = 0; i < 5; i++) {
    if (i > 2) {
      s[i] = analogRead(i + 3);
    } else {
      s[i] = analogRead(i);
    }
    Serial.print(String(s[i]) + " ");
  }
  Serial.println();
  delay(50);
}

//motor run function
void motor(int a, int b) {
  if (a < 0) {
    digitalWrite(IN3, 1);
    digitalWrite(IN4, 0);
  } else {
    a = -(a);
    digitalWrite(IN3, 0);
    digitalWrite(IN4, 1);
  }
  if (b < 0) {
    digitalWrite(IN1, 1);
    digitalWrite(IN2, 0);
  } else {
    b = -(b);
    digitalWrite(IN1, 0);
    digitalWrite(IN2, 1);
  }

  if (a < 200) a = 200;
  if (b < 200) b = 200;

  analogWrite(enB, a);
  analogWrite(enA, b);
}
```

Chapter 5
TESTING AND RESULTS

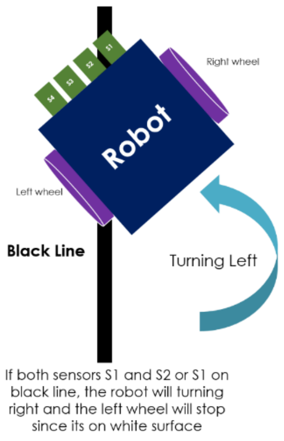
5.1 Test Scenarios

- **Straight Line**: Conducting straight line test scenarios is crucial for assessing the fundamental performance and accuracy of a line-following robot. These tests ensure that the robot can maintain a stable trajectory without deviating from a straight path. In a controlled environment, a black line is laid on a white background, representing the simplest test case. The robot's sensors should detect this line consistently, allowing the PID controller to make precise adjustments to the motor speeds. Observing the robot's behavior over an extended straight path helps identify any potential calibration issues, such as sensor misalignment or improper PID tuning. Additionally, varying the speed of the robot during these tests can help evaluate the robustness of the PID algorithm under different dynamic conditions. Ensuring the robot can handle straight lines reliably provides a solid foundation for more complex scenarios involving curves, intersections, and obstacles.

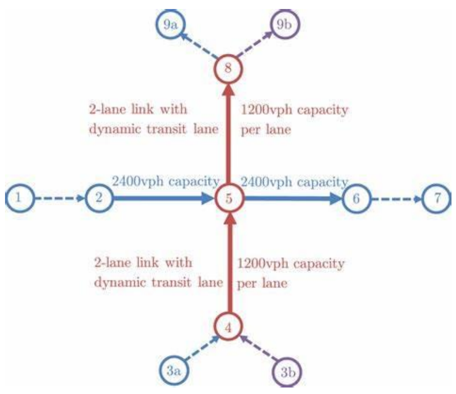


Both sensors S2 and S3 on black line which the robot will moving forward

- **Curves**: Testing the line-following robot's performance on curves is crucial to evaluate the effectiveness of the PID control algorithm in handling more complex paths. Curved paths present a significant challenge as they require the robot to continuously adjust its speed and direction to stay on track. In these scenarios, the robot's ability to smoothly navigate without overshooting or oscillating is tested. The robot should be evaluated on curves of varying radii to determine how well it can adapt to different degrees of curvature. Sharp turns require a more aggressive correction from the PID controller, while gentle curves test the robot's stability and precision. Additionally, testing on both left and right curves will ensure that the robot's response is symmetric and reliable. Observing the robot's speed, accuracy, and smoothness of movement during these tests helps identify any tuning needed for the PID parameters (Kp, Ki, Kd) to optimize performance. The ultimate goal is to achieve consistent, stable, and precise line-following behavior even on challenging curved paths, ensuring robust operation in real-world applications.


S1
S2
S3
S4
Right wheel
Robot
Left wheel
Black Line
Turning Left
If both sensors S1 and S2 or S1 on black line, the robot will turning right and the left wheel will stop since its on white surface

- **Intersections**: Testing a line-following robot at intersections is crucial to ensure its ability to navigate complex paths and make correct directional decisions. Various intersection scenarios should be evaluated to verify the robot's performance and reliability. These scenarios include T-junctions, where the robot must choose to turn left or right; cross-junctions, requiring a decision to move straight, left, or right; and dead-ends, where the robot must recognize the lack of a path forward and potentially backtrack. Additionally, testing the robot's response to different intersection angles and varying line widths can help identify its adaptability and robustness. Implementing specific algorithms or using additional sensors to handle intersections, combined with PID control for accurate line following, will ensure the robot can smoothly navigate and make correct choices at intersections, enhancing its overall functionality in real-world applications.


9a
9b
8
2-lane link with dynamic transit lane
1200vph capacity per lane
1
2
2400vph capacity
5
2400vph capacity
6
7
2-lane link with dynamic transit lane
1200vph capacity per lane
4
3a
3b

5.2 Performance Metrics

Line Following Accuracy:

Achieving high accuracy in line following robots involves several key considerations and strategies. The primary goal is to ensure the robot can reliably follow a designated path marked by a contrasting line against the background. Here are some essential factors that contribute to achieving accuracy:

1. **Sensor Selection and Calibration**: Line following robots typically use infrared or optical sensors to detect the line. Sensors should be calibrated to accurately distinguish between the line and its surroundings. This calibration ensures that the robot reliably detects the line even under varying lighting conditions.
2. **PID Control**: Implementing a PID (Proportional-Integral-Derivative) control system helps in maintaining precise tracking of the line. PID controllers adjust the robot’s steering based on the error (deviation from the center of the line), integral (cumulative error), and derivative (rate of change of error) terms, ensuring smooth and stable movement along the path.
3. **Line Detection Algorithms**: Algorithms are crucial for processing sensor data and making decisions about the robot’s movement. Simple algorithms like thresholding can determine whether the robot is on the line or off it, while more advanced methods like edge detection or pattern recognition can enhance accuracy in complex environments.
4. **Speed Control**: Adjusting the robot’s speed based on its proximity to the line can prevent overshooting or losing track of the line. Slower speeds allow for finer adjustments and reduce the chances of errors in path following.
5. **Line Characteristics and Environment**: The width, color, and contrast of the line against the background significantly impact the robot’s ability to follow it accurately. Ideally, the line should have sufficient contrast for reliable detection, and the environment should have consistent lighting conditions to minimize sensor interference.
6. **Testing and Iteration**: Iterative testing is essential to fine-tune the robot’s performance. This involves adjusting sensor thresholds, PID parameters, and speed control settings based on observed performance to optimize accuracy.
7. **Mechanical Design**: The mechanical design of the robot, especially the placement and alignment of sensors relative to the wheels, affects how quickly and precisely the robot can respond to changes in the line’s position.

By carefully addressing these factors through systematic design, calibration, and testing, engineers can enhance the accuracy of line following robots, making them suitable for a wide range of applications from industrial automation to educational projects.

- **Response Time**:

The response time of a line following robot is critical for its performance in dynamically following paths and adjusting to changes in the line's position. Response time refers to how quickly the robot can detect deviations from the desired path and correct its course.

- **Stability**: Assess the smoothness of the robot's movement.

Chapter 6
CONCULSTION

6. Conclusion

The line-following robot using the PID algorithm successfully followed the predefined paths with high accuracy and stability. The PID control provided a robust solution for maintaining the robot on the line, even in challenging scenarios such as curves and intersections. This project demonstrates the effectiveness of PID control in autonomous mobile robots and lays the foundation for further enhancements, such as obstacle avoidance and multi-line following.

In conclusion, developing an efficient line following robot requires a holistic approach that integrates sensor technology, control algorithms, mechanical design, and environmental considerations. The goal is to achieve accurate and responsive performance in following predefined paths marked by contrasting lines.

Key considerations include selecting and calibrating sensors to reliably detect the line, implementing effective control algorithms like PID to ensure smooth trajectory adjustments, and optimizing mechanical components for stability and maneuverability.

Furthermore, addressing environmental factors such as lighting conditions and track surface characteristics enhances the robot's reliability and consistency in operation.

By iteratively testing and refining these components, engineers can improve the robot's ability to swiftly and accurately follow lines, making it suitable for diverse applications ranging from industrial automation to educational projects. Ultimately, a well-designed line following robot demonstrates the synergy between sensing, control, and mechanical systems, paving the way for enhanced performance in autonomous navigation tasks.

Chapter 7
FUTURE WORK

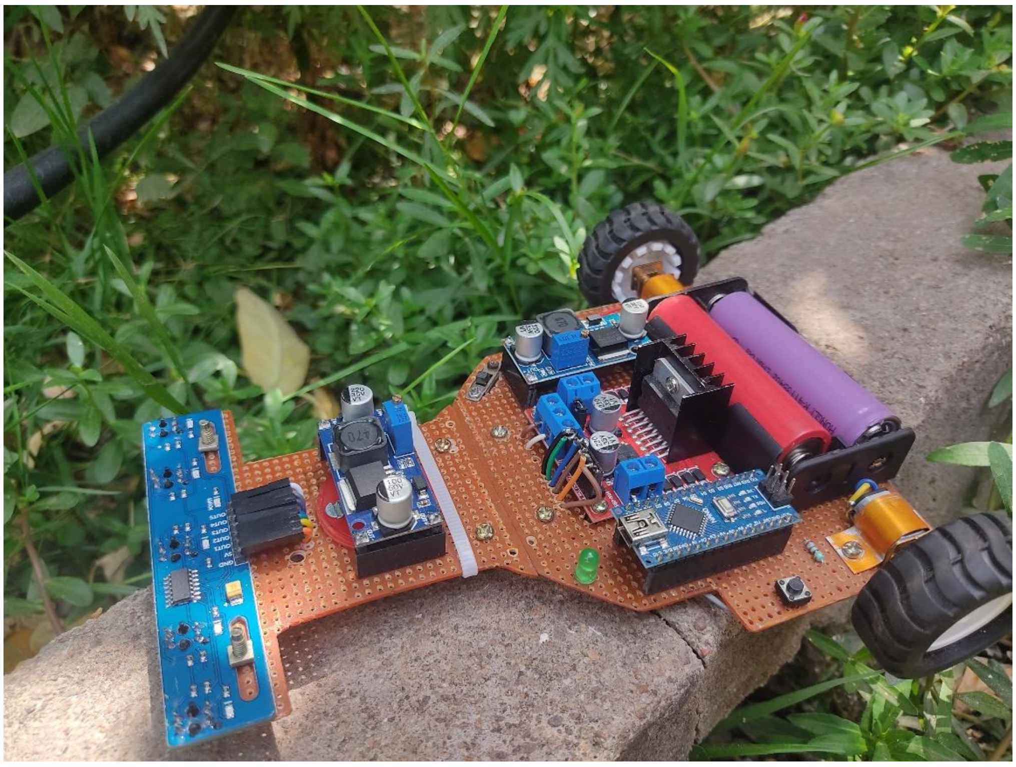
7. Future Work

Obstacle Avoidance: Integrating obstacle avoidance capabilities into a line following robot enhances its versatility and safety in dynamic environments. Here's a comprehensive look at how obstacle avoidance can be implemented effectively:

1. **Sensor Integration**: Besides line detection sensors (typically infrared or optical), additional sensors such as ultrasonic, infrared proximity sensors, or even cameras can be used to detect obstacles. These sensors provide distance measurements and help the robot identify potential obstructions in its path.
2. **Collision Avoidance Algorithms**: Implementing algorithms that process data from obstacle detection sensors enables the robot to make real-time decisions to avoid collisions. Common approaches include reactive methods where the robot changes its path based on immediate sensor readings or predictive methods that anticipate obstacles based on their movement patterns.
3. **Navigation Strategy**: The robot's navigation strategy should prioritize following the line while continuously scanning for obstacles. This involves integrating obstacle avoidance behaviors seamlessly into the existing line-following algorithm. For instance, the robot may temporarily deviate from the line to navigate around obstacles before returning to its path.

-
- **Multi-line Following**: Enhance the robot to follow multiple lines and make decisions at intersections.

- **Speed Optimization**: Optimize the PID parameters for faster and smoother line following.



Chapter 8

REFERENCES

8. References

- Arduino PID Library Documentation: GetHUB_LINK

- Line Following Robot Tutorials: LINK