TDD-FDP-CCE

Design Line Follower Mobile Robot (Technical Design Document)

Project Name			
Student	[1] Tran Van Phuoc [2] Quynh Nhu [3] Nguyen Quoc Bao	**ID**	1610111 1610000
Major	Electronics and Communication Engineering	**Supervisor**	Assoc. Prof. Phan Van Ca

Copyright ©2021 CCE Department.

Contents

▪ Revision History

Version	Date	Content of revision	Author	Supervisor
1.0	02-01-2021			
1.1				
1.2				
1.3				
1.4				
1.5				
2.0				

▪ Terms and abbreviations

[UART]	Universal asynchronous receiver-transmitter.
[USB]	Universal Serial Bus.
[IDE]	Integrated development environment.

▪ References

[1] A. A. Galadima, "Arduino as a learning tool," 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, 2014, pp. 1-4, doi: 10.1109/ICECCO.2014.6997577.

[2] V. Oza and P. Mehta, "Arduino Robotic Hand: Survey Paper," 2018 International Conference on Smart City and Emerging Technology (ICSCET), Mumbai, 2018, pp. 1-5, doi: 10.1109/ICSCET.2018.8537312.

▪ List of Tables & Figures

1 Introduction

1.1 INTRODUCTION TO LINE FOLLOWER ROBOTS

A line follower robot is a type of autonomous mobile robot designed to follow a path or line on the ground. This path is usually represented as a high-contrast line, such as black on white or white on black. The robot uses sensors to detect the line and adjusts its motion to stay on track. These robots are often used to demonstrate fundamental concepts in robotics, such as sensor integration, control systems, and basic navigation. Line follower robots are widely utilized in educational settings to teach about automation and robotics, as well as in industrial applications for material handling and logistics. The movement of these robots is primarily controlled through feedback from sensors that continuously detect the line and send data to the motor controllers.

1.2 PRINCIPLE OF OPERATION

- **Basic Functionality**: The primary working principle of a line follower robot is based on feedback control. The sensors detect the line, and the robot's microcontroller processes this data to adjust the speed and direction of the wheels or motors.
- **Sensor Mechanism**: The most common sensors used in line follower robots are infrared (IR) sensors or Light Dependent Resistors (LDRs), which detect variations in light intensity as the robot moves over the line.
- **Control Algorithms**: Typically, simple control algorithms like On/Off control or more advanced proportional-integral-derivative (PID) controllers are used to maintain the robot's course along the line.

1.3 COMPONENTS OF A LINE FOLLOWER ROBOT

- **Sensors**: Infrared sensors, ultrasonic sensors, or color sensors that help the robot detect the line or obstacles.
- **Microcontroller**: The brain of the robot, such as Arduino, Raspberry Pi, or other microcontrollers, that processes sensor data and commands the motors.
- **Motors and Motor Drivers**: DC motors or stepper motors are often used to provide movement. Motor drivers like L298N control the speed and direction of the motors based on signals from the microcontroller.
- **Power Supply**: Batteries or rechargeable power sources are typically used to supply power to the motors, sensors, and microcontroller.

1.4 APPLICATIONS OF LINE FOLLOWER ROBOTS

- **Industrial Automation**: Line follower robots are extensively used in Automated Guided Vehicle (AGV) systems in factories and warehouses. These robots can transport materials from one location to another, reducing manual labor and enhancing operational efficiency.
- **Logistics and Delivery Systems**: In logistics, line follower robots can be used to navigate through warehouses, carrying goods along predefined paths. In more advanced settings, they are being developed to deliver small packages autonomously within stores or offices.
- **Educational Applications**: Line follower robots are a popular project in educational programs, helping students learn about sensors, actuators, and basic robotics principles. They are often used in competitions where students build and program their own robots.
- **Public and Private Sectors**: In industries like agriculture, healthcare, or even airports, these robots can be deployed to move equipment or supplies along predetermined routes, improving operational efficiency and reducing human error.

1.5 STATE OF RESEARCH AND DEVELOPMENT IN LINE FOLLOWER ROBOTS WORLDWIDE

- **Technological Advances**: In recent years, significant progress has been made in improving the sensors and control algorithms used in line follower robots. The development of more accurate infrared sensors, as well as the use of machine learning and artificial intelligence, has led to robots that can handle more complex paths and environments.
- **AI Integration**: Artificial intelligence (AI) is now being integrated into line follower robots to enhance their capabilities. AI algorithms allow these robots to adapt to new, unknown, or changing environments, making them more flexible and robust in dynamic settings.
- **Control Systems and Algorithms**: Advanced control techniques, such as fuzzy logic and PID controllers, are now commonly used to improve the performance of these robots. Moreover, machine learning models are being developed to allow robots to learn optimal paths and behaviors through experience.
- **Commercial and Industrial Use**: Many companies are working on industrial-grade AGVs that are based on line-following principles. These robots can follow a line through a large warehouse or factory floor, adapting to changing environments and delivering goods efficiently.

1.6 RESEARCH AND DEVELOPMENT OF LINE FOLLOWER ROBOTS IN VIETNAM

- **Industrial Applications**: The application of line follower robots is beginning to gain traction in Vietnam, especially in the manufacturing sector. Many companies are investing in robotic systems to automate material handling processes, improve efficiency, and reduce labor costs.
- **Academic Research**: Vietnamese universities and research institutions are increasingly involved in the development of line follower robots. These robots are often part of robotics and automation programs, where students and researchers experiment with various sensors, algorithms, and hardware configurations.
- **Challenges**: Despite the growing interest, there are several challenges to the widespread adoption of line follower robots in Vietnam, including the high initial costs, limited availability of advanced components, and a need for more trained professionals in the field of robotics and automation.
- **Opportunities**: With the ongoing push toward automation in industries like manufacturing, logistics, and agriculture, Vietnam presents a promising market for the development and implementation of line-following robots. Moreover, research in robotics education and DIY robotics kits is gaining popularity, creating opportunities for students to build and experiment with their own robots.

1.7 OBJECTIVES AND SCOPE OF THE STUDY

- **Objectives**: This study aims to design and implement a line follower robot that can efficiently follow a simple or complex path using infrared sensors and basic control algorithms. The robot will be tested in different environments to assess its performance and adaptability to various conditions.
- **Scope of the Study**: The study will focus on the design, prototyping, and programming of a line follower robot using an Arduino microcontroller and IR sensors. The control algorithms used will include basic On/Off control and PID control to evaluate the robot's ability to follow the line under different conditions

2 Concept/Technology

2.1 CUSTOMER NEEDS

This section outlines the requirements and expectations of the customer or end user for the system or product you are developing. In the case of a line follower robot, customer needs might include:

- High accuracy: The robot should be able to track the path precisely without deviating from the line.
- Ability to operate in different environments: Customers may want the robot to function effectively on various surfaces and under different lighting conditions.
- Cost-effectiveness: The line follower robot needs to be designed with a reasonable cost while maintaining performance and durability.
- Ease of maintenance and upgrades: The system should be easy to maintain, repair, and upgrade when necessary.
- Flexibility and scalability: The system should have the potential to support additional features in the future, such as obstacle detection or the ability to follow more complex paths.

2.2 ENGINEERING REQUIREMENTS

This section details the technical requirements needed to develop the system to meet customer needs.

2.2.1 Functions

Functions are the operational requirements the system must fulfill. For a line follower robot, these functions may include:

- Path recognition and following: The robot must be able to accurately and steadily follow a line regardless of environmental or lighting conditions.
- Motor control: The system should be capable of controlling motors to adjust the speed and direction of the robot based on sensor inputs.
- Feedback control: The robot should have a feedback control system such as a PID controller to maintain stability and minimize errors while following the line.
- Navigation in complex environments: The robot should be able to navigate through curves and small obstacles, if present.
- Scalability: The system should support additional features in the future, such as obstacle avoidance or the ability to follow more complex paths

2.2.2 Non-functions

Non-functional requirements refer to aspects that are not directly related to the core functionality but are crucial for the overall performance and success of the system. Non-functional requirements for a line follower robot might include:

- Reliability and durability: The robot needs to operate consistently over extended periods without encountering hardware or software issues.
- Energy efficiency: The robot should be energy-efficient, ensuring it can operate for long periods on a single battery charge or set of batteries.
- Expandability: The system should be able to integrate new features or technologies without requiring a complete overhaul of the hardware or software.
- Ease of use and maintenance: The system should be user-friendly and easy to maintain, ensuring that users can replace parts or upgrade features without difficulty.
- Open and flexible architecture: The system should support open connectivity standards, enabling integration with new sensors or systems in the future.

3 System Architecture

3.1 OVERVIEW OF THE SYSTEM ARCHITECTURE

The architecture of the line follower robot consists of several key components that work together to achieve the desired functionality of autonomous navigation along a predefined path. The system includes both hardware and software elements, each designed to meet the engineering requirements and fulfill the customer needs identified in Chapter 2. The main components of the system include sensors, actuators (motors), the microcontroller, and the software that controls the system.

3.2 HARDWARE ARCHITECTURE

This section will provide a detailed description of the hardware components used in the line follower robot.

- **Microcontroller**:
  The brain of the robot, typically an Arduino, Raspberry Pi, or any suitable microcontroller, processes sensor inputs and sends commands to the motors. The microcontroller interprets the sensor data to control the movement of the robot.
- **Sensors**:
  The robot uses sensors (usually infrared or optical sensors) to detect the line. The sensors continuously monitor the environment and detect the presence or absence of the line to send feedback to the microcontroller.
  - **Infrared Sensors**: These sensors are typically used to detect the line. They are placed underneath the robot to sense the contrast between the line and the surface. Depending on whether the sensor detects the line or not, the microcontroller adjusts the robot's movement.
- **Motors and Motor Drivers**:

  The motors are responsible for moving the robot. DC motors or stepper motors are usually employed, with motor drivers like the L298N controlling the direction and speed of the motors. The motor drivers receive control signals from the microcontroller and ensure smooth operation of the motors.

- **Power Supply**:

  The robot requires a power source to operate the sensors, motors, and microcontroller. A rechargeable battery, such as a Li-ion or Li-poly battery, is typically used to provide the necessary voltage and current to all components

3.3 SOFTWARE ARCHITECTURE

The software architecture governs the behavior of the robot, processing the sensor data and controlling the motors to ensure the robot stays on the path.

- **Control Algorithms**:
  The software includes algorithms to process sensor data and make real-time decisions on the robot's movement. Common algorithms include:
    - **On/Off Control**: A simple approach where the robot adjusts its direction when a sensor detects that it is off the line.
    - **PID Control**: A more advanced algorithm that adjusts the robot's speed and direction based on the error between the desired position and the current position. This allows for smoother and more precise following of the line.
- **Sensor Data Processing**:
  The software continuously reads inputs from the sensors, processes the data to determine if the robot is on the line or not, and sends commands to the motor drivers to make appropriate adjustments. If the sensors detect that the robot has veered off the line, corrective actions are taken to bring the robot back to the path.
- **Communication Protocol**:
  If the robot is part of a larger system or is intended to provide feedback to an external system, communication protocols like UART or I2C may be implemented to allow for data exchange between the robot and other devices (e.g., a monitoring system).

3.4 SYSTEM INTEGRATION

This section explains how the different components work together as a cohesive system.

- **Data Flow**:
  The sensors constantly feed data to the microcontroller. Based on the sensor inputs, the microcontroller processes the information and sends appropriate commands to the motor driver to adjust the robot’s movement. The motor drivers then control the motors, allowing the robot to stay on the line
- **Feedback Loop**:
  A key aspect of the system is the feedback loop, where the robot continuously monitors its position relative to the line. The feedback allows for real-time adjustments to maintain optimal performance, ensuring that the robot follows the path with minimal deviation.

- **System Reliability**:

  The architecture should ensure that each component operates reliably under different conditions. The robustness of the hardware and software must be tested to ensure that the robot functions efficiently over long periods, even in dynamic environments.

3.5 SUMMARY OF SYSTEM ARCHITECTURE

The system architecture of the line follower robot combines several hardware and software elements that work together to achieve the primary objective of autonomous navigation along a path. The architecture includes sensors for detecting the line, a microcontroller for processing data, motor drivers for controlling movement, and control algorithms that ensure the robot follows the path with minimal error.

4 Detailed Design

4.1 OVERVIEW OF THE DESIGN

This chapter describes the detailed design of the line-following robot system, including the hardware components used, how they are connected, and how they work together to perform the line-following function. The main components of the system include: **Arduino Uno R3 Microcontroller**, **L298N Motor Driver**, **DC Motors**, **TCRT5000 Sensors**, **18650 Batteries**, and other supporting parts.

4.2 HARDWARE COMPONENTS

4.2.1 Arduino Uno R3 Microcontroller

The Arduino Uno R3 serves as the "brain" of the system, processing signals from the sensors and controlling the motors. It receives input from the TCRT5000 sensors to detect the line and makes decisions about the robot's movement. Arduino Uno also manages signals from buttons and controls the motor driver.

- **Connections with other components**:
  - The Arduino’s GPIO (General Purpose Input/Output) pins will connect to the TCRT5000 sensors to receive signals and decide the robot’s movement.
  - It will send control signals to the L298N motor driver to adjust the speed and direction of the motors.

Figure 1: uno r3 MCU

4.2.2 L298N Motor Driver

The L298N is a dual H-Bridge motor driver capable of controlling two DC motors. It receives control signals from the Arduino and drives the motors in different directions to move the robot along the line.

- **Connections with other components**:
  - Pins IN1, IN2, IN3, IN4 on the L298N connect to the GPIO pins on the Arduino to receive control signals from the microcontroller.
  - Output pins OUT1, OUT2, OUT3, OUT4 of the L298N are connected to the two DC motors to control their speed and direction.
  - ENA and ENB pins of L298N will be used to control the motor speed through PWM signals

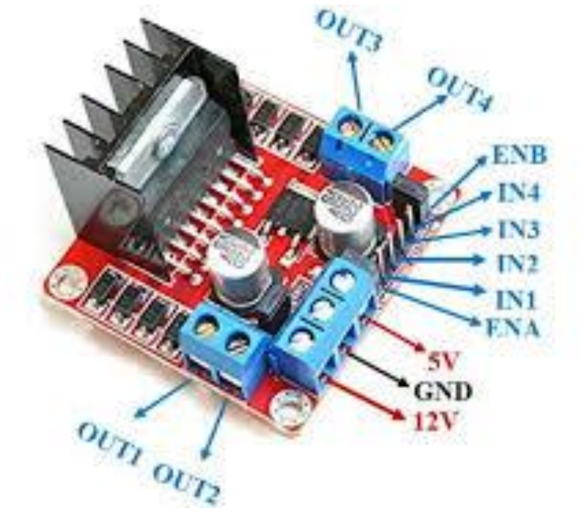

Figure 2: Driver L298N

4.2.3 DC Motors and Wheels

The robot uses two yellow DC motors with wheels to provide movement. The DC motors generate rotational motion, which drives the wheels, enabling the robot to follow the line.

- **Connection with the L298N motor driver**:
    - The DC motors are connected to the OUT1 and OUT2 pins (for the left motor) and OUT3 and OUT4 pins (for the right motor) of the L298N.
    - PWM control (ENA, ENB) adjusts the motor speed.

Figure 3: DC MOTOR V1 AND WHEELS

4.2.4 TCRT5000 Sensors

The TCRT5000 is an optical sensor used to detect the line. The system uses a sensor array consisting of 5 TCRT5000 sensors placed in a line beneath the robot to track the line and detect changes in surface color (to differentiate between the line and the background).

- **Connections with Arduino**:
    - Each TCRT5000 sensor has an output signal pin (OUT) that the Arduino reads. These signals help decide whether the robot is on the track and if it needs to adjust its direction.
    - The sensors are connected to the analog or digital input pins of Arduino to receive the signals.

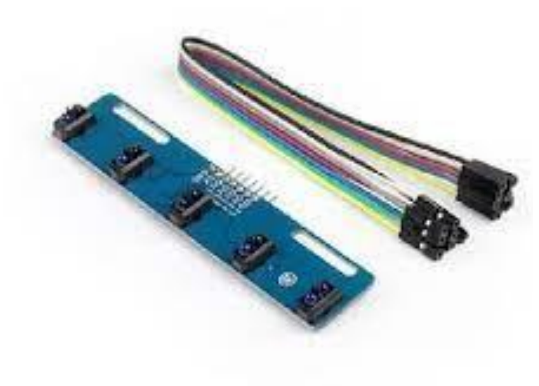

Figure 4: IR SENSOR ARRAY

4.2.5 18650 Batteries

Two 18650 lithium-ion batteries provide power to the entire system. Each battery is connected in series (for 12V) or in parallel (to maintain 3.7V) depending on the system's power requirements.

- **Connection with components**:
  - The batteries provide power to the Arduino, L298N motor driver, and DC motors.
  - A battery protection module is used to ensure safety while using the 18650 cells, preventing overcharging or overheating

Figure 5: Battery

4.3 CIRCUIT DESIGN

In the circuit design, all components are interconnected via the GPIO pins of the Arduino and the L298N motor driver.

- **TCRT5000 Sensors**: Connected to the input pins of the Arduino to receive signals from the line and send feedback to the Arduino for processing.
- **L298N Motor Driver**: Receives control signals from Arduino to drive the DC motors in the correct direction and speed.
- **DC Motors**: Provide movement for the robot, controlled by L298N.
- **18650 Batteries**: Provide power to all components

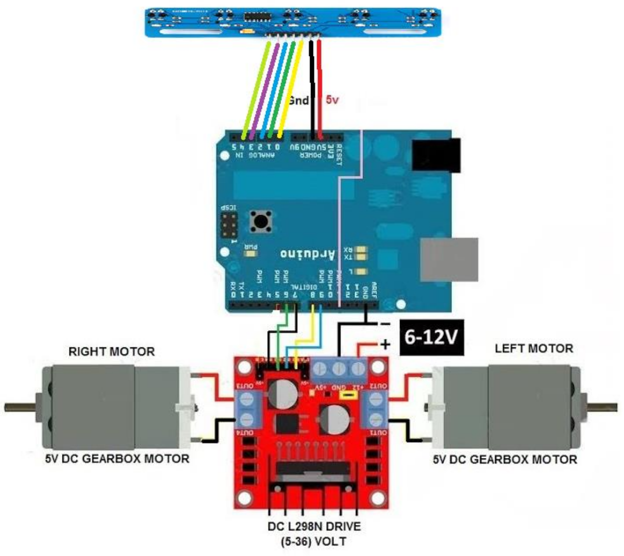

Figure 6: Wiring diagram

4.4 SOFTWARE DESIGN

The control software processes the data from the sensors and determines the robot's actions. Common algorithms include:

- **Simple Control Algorithm**: Based on sensor input, the robot adjusts its direction (turn left, right, move forward, or reverse) to stay on track.
- **PID Control Algorithm**: A more advanced algorithm used to maintain the robot's stability as it follows the line, ensuring it doesn't drift too far from the path.

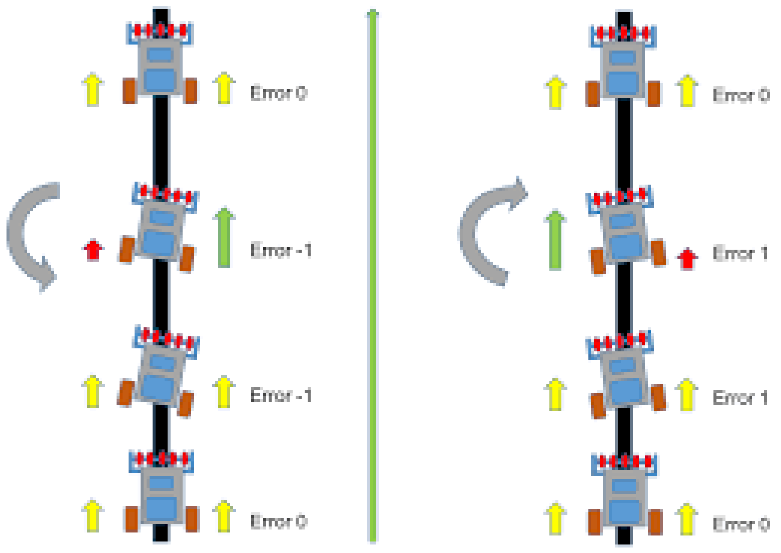

Figure 7: PID Controlled

4.5 SYSTEM INTEGRATION

After designing the hardware and software, system integration is crucial to ensure the components work together seamlessly. The components operate as follows:

- **Sensors** detect the line and send signals to Arduino.
- **Arduino** processes the sensor data and sends control commands to the **L298N motor driver**.
- **L298N** controls the **DC motors** to move the robot along the line.

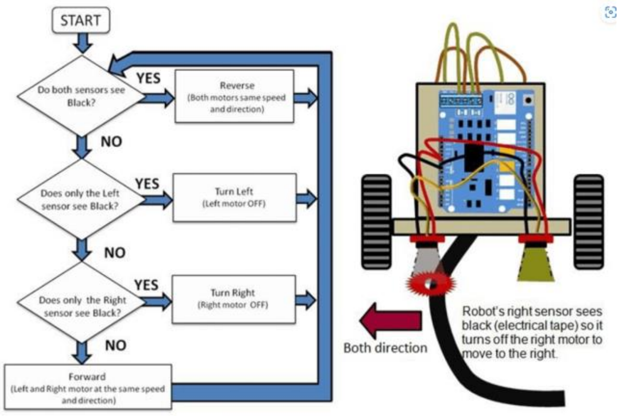
START
Do both sensors see Black?
YES
Reverse (Both motors same speed and direction)
NO
Does only the Left sensor see Black?
YES
Turn Left (Left motor OFF)
NO
Does only the Right sensor see Black?
YES
Turn Right (Right motor OFF)
NO
Forward (Left and Right motor at the same speed and direction)
Both direction
Robot's right sensor sees black (electrical tape) so it turns off the right motor to move to the right.

Figure 8: Example of system integration

4.6 Testing and Calibration

Finally, testing and calibration of the components are necessary to ensure the system operates as expected. Testing procedures include:

- Checking the accuracy of the TCRT5000 sensors.
- Verifying the motor control functionality of the L298N.
- Ensuring compatibility between hardware and software

5 Final Product

5.1 PRODUCT OVERVIEW

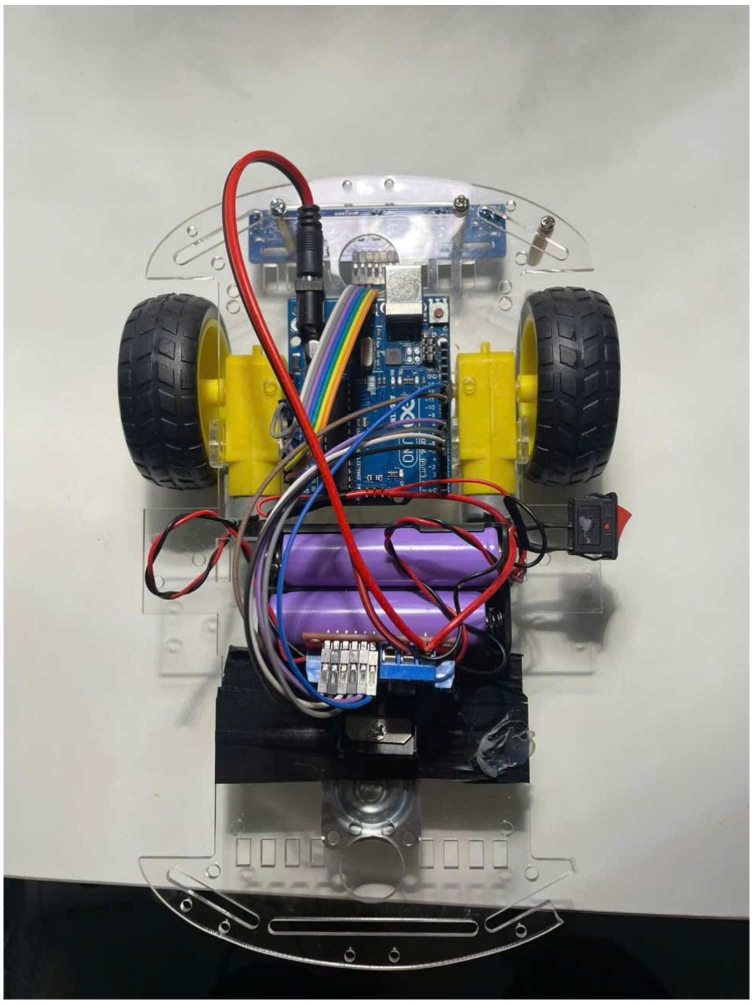

Figure 9. Overview of mobile robot

The final product of this project is an autonomous line-following robot designed to navigate along a predefined path. The robot consists of a microcontroller (Arduino Uno R3), two DC motors, an L298N motor driver, five TCRT5000 sensors for line detection, and two 18650 lithium-ion batteries. The robot's primary function is to detect and follow a black line on a white surface, adjusting its path as necessary to stay on course

5.2 System Features and Specifications

- **Dimensions and Structure**: The robot has a compact and lightweight design, measuring 20 cm in length, 15 cm in width, and 10 cm in height. The chassis is built using plastic material to ensure durability and low weight.
- **Key Components**:
  - **Arduino Uno R3**: Acts as the brain of the robot, processing sensor data and controlling the motors.
  - **L298N Motor Driver**: Controls the speed and direction of the motors based on signals from the Arduino.
  - **DC Motors and Wheels**: Provide the motion for the robot, enabling it to follow the line.
  - **TCRT5000 Sensors**: Detect the presence of the black line and send signals to the Arduino to adjust the robot's movement accordingly.
  - **18650 Batteries**: Supply power to the entire system, ensuring long operational hours.

5.3 Product Performance

The robot performs well in detecting and following the line with an accuracy of approximately 80-90%. The sensors effectively distinguish the black line from the white background, and the robot adjusts its path to stay centered on the line. The robot's speed is consistent, and it successfully navigates around sharp turns and curves with minimal deviation. During testing, the robot was able to continuously follow the line for up to 2 hours on a fully charged battery.

5.5 Product Testing and Results

Various tests were conducted to ensure the robot functions as intended:

- **Functionality Test**: The robot successfully followed the line in different environments, such as on curved paths and with varying light conditions.
- **Endurance Test**: The robot ran for 2 hours on a single charge of the 18650 batteries, with no significant performance degradation.
- **Performance Against Objectives**: The robot met all the objectives outlined at the beginning of the project, including accurate line-following, smooth turning, and efficient power usage.

5.6 Challenges Encountered and Solutions

Throughout the development of the project, several challenges were faced:

- **Sensor Calibration**: Initially, the sensors were too sensitive to light changes, causing the robot to deviate from the line. This was solved by adjusting the sensor's threshold values in the code.
- **Motor Control Issues**: The motors sometimes did not respond as expected due to voltage fluctuations. This was resolved by adding capacitors to stabilize the power supply and using PWM control to regulate motor speed.

5.7 Conclusion and Future Work

The final product meets the primary goal of creating an efficient and reliable line-following robot. The design and implementation were successful, and the robot demonstrated stable performance in real-world testing. Future improvements may include adding more advanced sensors, such as infrared or camera-based systems, to enhance the robot's ability to follow more complex paths. Additionally, increasing the robot's autonomy by adding obstacle detection and avoidance features could open up new applications in various industries, including automation and robotics education.

6 Appendix

6.1 Division of Labor

ID	Activity	Description	Deliverables/ Checkpoints	Duration (Days)	People	Resources	Predecessors
1	Programing						
2	Design chsis, v.v						
3	Order component and assembly						
4	Make final report						

6.2 Bill of Material

ID	Parts/Components	Amount	Price per Unit	Total
1	Arduino UNO board	1	110000	110000
2	Mica frame	1	30000	30000
3	Ir sensor array	1	32000	32000
4	L298n	1	25000	25000
5	Dc motor and wheels	2	18000	36000
6	Optional wheel	1	10000	10000
7	Battery	2	20000	40000
8	Battery holder	1	12000	12000
9	Wiring	1	17000	17000
10	Switch	1	5000	5000

6.3 Gantt Chart

ID	Task name	Start	Finish	Duration	Layout
1	Project planning and reachh	1	2	1	1.1 Define project scope and objectives 1.2 Research line following robot concepts 1.3 Identify necessary components
2	Component Procurement and Assembly	2	5	3	2.1 Order components 2.2 Assemble chassis 2.3 Install electronics
3	Software Development	6	8	2	3.1 Develop basic motor control 3.2 Implement line following algorithm 3.3 Integrate sensor readings
4	Testing and Calibration	8	10	2	4.1 Initial testing on straight line 4.2 Calibrate sensors and adjust parameters 4.3 Test on curved lines and obstacles
5	Documentation and Reporting	10	11	1	5.1 Write project report

6.4 Programing code

```
// IR Sensors
int sensor1 = A0;      // Left most sensor
int sensor2 = A1;
int sensor3 = A2;
int sensor4 = A3;
int sensor5 = A4;     // Right most sensor

// Initial Values of Sensors
int sensor[5] = {0, 0, 0, 0, 0};

// Motor Variables
int ENA = 11;
int IN1 = 7;
int IN2 = 6;
int IN3 = 5;
int IN4 = 4;
int ENB = 10;



// Initial Speed of Motor
int initial_motor_speed = 67; // Reduced speed for slower movement

// PID Constants
float Kp = 10;  // Adjusted PID values for smoother operation
float Ki = 2.2;
float Kd = 90;

float error = 0, P = 0, I = 0, D = 0, PID_value = 0;
float previous_error = 0, previous_I = 0;

void setup() {
  pinMode(sensor1, INPUT);
  pinMode(sensor2, INPUT);
  pinMode(sensor3, INPUT);
  pinMode(sensor4, INPUT);
  pinMode(sensor5, INPUT);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);

  Serial.begin(9600);                     // Setting serial monitor at a
default baud rate of 9600
  }

void loop() {
  readSensor();
  Serial.println(error);

  if(error == 222|| error == 999) BotStop();
  else {
    PIDControl();}

}

void readSensor(){

  sensor[0] = digitalRead(sensor1);
  sensor[1] = digitalRead(sensor2);
  sensor[2] = digitalRead(sensor3);
  sensor[3] = digitalRead(sensor4);
  sensor[4] = digitalRead(sensor5);

// full đen
   if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 0) && (sensor[3]
== 0) && (sensor[4] == 0)) error = 222;

// full trắng
   else if ((sensor[0] == 1) && (sensor[1] == 1) && (sensor[2] == 1) &&
(sensor[3] == 1) && (sensor[4] == 1)) error = 999;

// xe không lệch
   else if ((sensor[0] == 1) && (sensor[1] == 1) && (sensor[2] == 0) &&
(sensor[3] == 1) && (sensor[4] == 1)) error = 0;

// lệch trái ít
   else if ((sensor[0] == 1) && (sensor[1] == 1) && (sensor[2] == 0) &&
(sensor[3] == 0) && (sensor[4] == 1)) error = 1;
   else if ((sensor[0] == 1) && (sensor[1] == 1) && (sensor[2] == 1) &&
(sensor[3] == 0) && (sensor[4] == 1)) error = 1.5;


// lệch trái nhiều
   else if ((sensor[0] == 1) && (sensor[1] == 1) && (sensor[2] == 1) &&
(sensor[3] == 0) && (sensor[4] == 0))  error = 2;
   else if ((sensor[0] == 1) && (sensor[1] == 1) && (sensor[2] == 1) &&
(sensor[3] == 1) && (sensor[4] == 0)) error = 3;

// lệch phải ít
   else if ((sensor[0] == 1) && (sensor[1] == 0) && (sensor[2] == 0) &&
(sensor[3] == 1) && (sensor[4] == 1)) error = -1;
   else if ((sensor[0] == 1) && (sensor[1] == 0) && (sensor[2] == 1) &&
(sensor[3] == 1) && (sensor[4] == 1)) error = -1.5;


// lệch trái nhiều
   else if ((sensor[0] == 0) && (sensor[1] == 0) && (sensor[2] == 1) &&
(sensor[3] == 1) && (sensor[4] == 1))  error = -2;
   else if ((sensor[0] == 0) && (sensor[1] == 1) && (sensor[2] == 1) &&
(sensor[3] == 1) && (sensor[4] == 1)) error = -3;

}
void PIDControl(){

 Serial.println("Hi, im pid control");
  P = error;
  I = I + previous_I;
  D = error - previous_error;

  PID_value = (Kp * P) + (Ki * I) + (Kd * D);

  previous_I = I;
  previous_error = error;

  int left_motor_speed = initial_motor_speed + PID_value;
  int right_motor_speed = initial_motor_speed - PID_value;

  // The motor speed should not exceed the max PWM value
  left_motor_speed = constrain(left_motor_speed, 0, 255);
  right_motor_speed = constrain(right_motor_speed, 0, 255);

  analogWrite(ENB, left_motor_speed); //Left Motor Speed
  analogWrite(ENA, right_motor_speed); //Right Motor Speed
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
void BotStop(){

     analogWrite(ENA,255);
     analogWrite(ENB,255);
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, LOW);
     digitalWrite(IN3, LOW);
     digitalWrite(IN4, LOW);
}
```