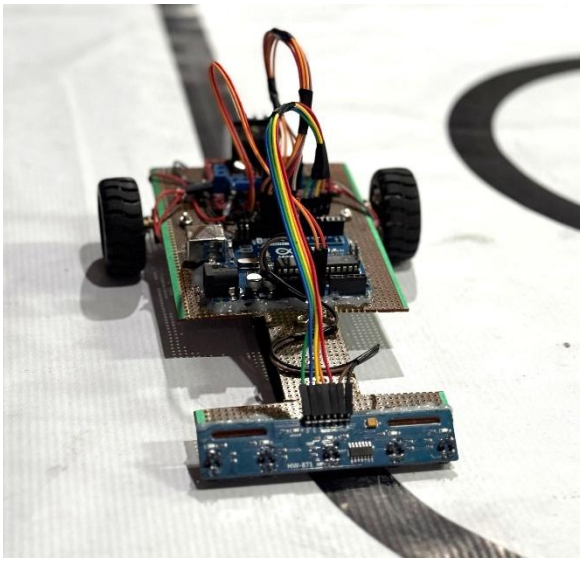
DESIGN AND DEVELOPMENT OF A LINE FOLLOWING ROBOT

EE-1010 Mini Project

Project Report



Team Members

NANDASIRI K.M.I.B. (E/23/236)

NAWARATHNA D.M. (E/23/241)

NULAR M.M.M.A. (E/23/249)

PERIS M.I.C. (E/23/260)

PERERA T.N.D. (E/23/270)

PERERA U.T.N. (E/23/271)

PREMASIRI L.N.N. (E/23/279)

Table of Contents

OUTLINE

A line-following robot is a robot system that moves along a predetermined path with sensors. It will follow a white line on a black surface or a black line on a white surface. Infrared sensors or light-dependent resistors detect the contrast in brightness between the line and surface, allowing the robot to remain on course. A microcontroller decodes the sensor information and drives the motors to remain on course. These robots are widely used in research studies and projects for learning about robotics, automation, and control systems. They are used industrially with practical applications such as navigating automatic vehicles in warehouses and factories.

ACKNOWLEDGEMENT

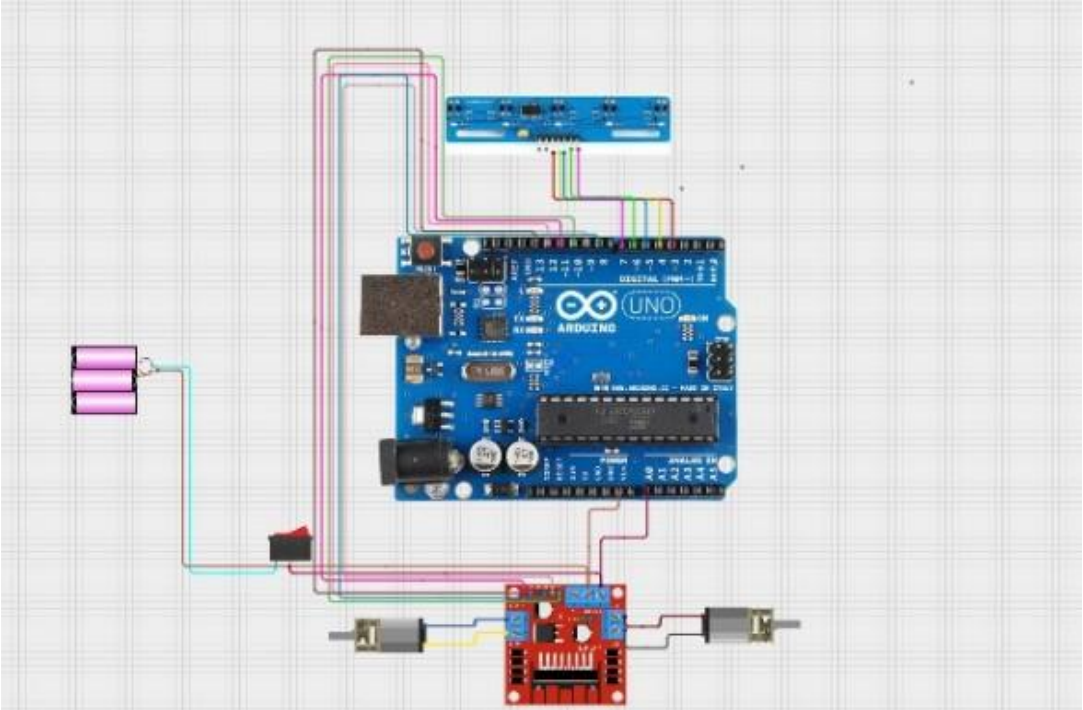
We are privileged to submit our gratitude to everyone who helped us towards the success of this project. First and foremost, we are thankful to our project guide Dr. Janaka Wijayakulasooriya from the bottom of our hearts for their constructive criticism, advice, and continuous encouragement. Their knowledge led us through the breach of the technology and theory boundaries of robotics.

We also appreciate the Department of Electrical and Electronic Engineering's provision of necessary resources, facilities, and laboratory support, which were critical to the research and development process. This project would not have been possible without these arrangements.

Also, we really appreciate the good words and recommendation of our batchmates, which inspired us to the project. Lastly, we appreciate the love and support of our families for accompanying us always, tolerating us and urging us at all times.

Thank you all for your feedback, which made this project worthwhile and rewarding to study.

CIRCUIT DIAGRAM



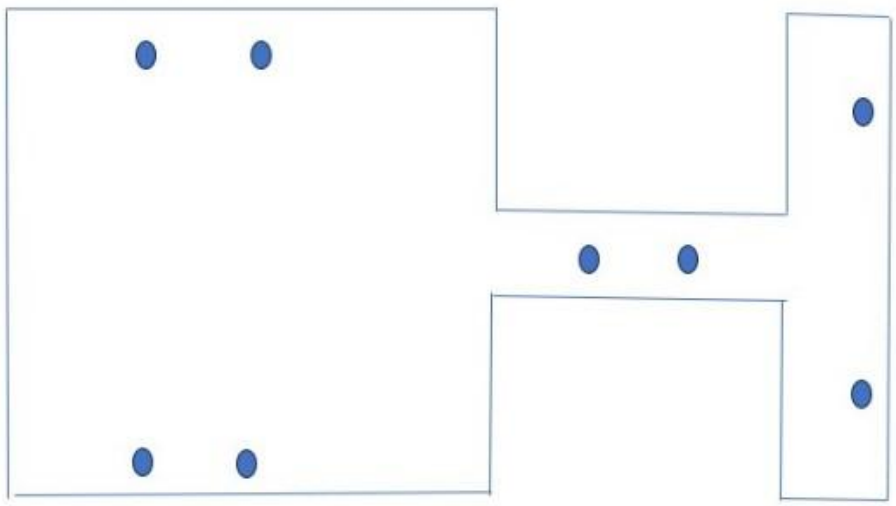
- Arduino Uno Microcontroller
- L298N Motor Driver Module
- HW 871 Five Sensor Array
- 200 RPM Metal Gear Motors
- 3.7V 1800mA 18650 Li-ion Rechargeable Batteries
- Push Lock Switch

HARDWARE LIST

Components

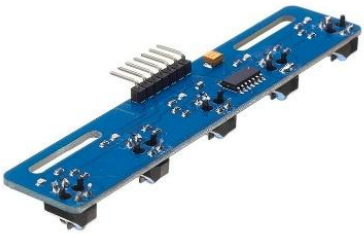
Chassis and Structure:

The body was constructed of a PCB dot board, which was selected due to its lightweight and flexibility. This material provided a solid surface to mount components on and supported straightforward wiring and circuit connections. The components were held in place by nuts and bolts, with the ability to easily remove or reconfigure components and maintain compactness and solidity of the structure. This was achieved while facilitating flexibility, which made it possible to easily assemble and make changes when needed.



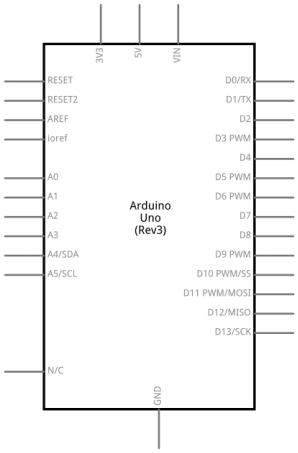
IR Sensor:

The Five Sensor Array HW-871 was employed for stable performance to recognize and follow the course. The sensor configuration enables the robot to successfully detect the line and cross turns and bends without deviation. Its effective detection range enhances responsiveness and stability to enable smooth movement and enhanced adaptation to shifting course conditions.



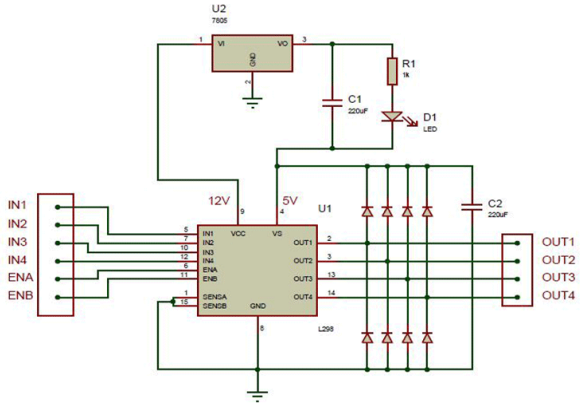
Arduino Uno:

The Arduino Uno was chosen due to its reliability, sufficient processing power, and simplicity of integration with essentials. Its sufficient digital and analog pins ensure a seamless integration with the HW-871 Five Sensor Array, ensuring flawless processing of data and effective control over the motion of the robot. Additionally, its universal compatibility and solid support make it the perfect choice for this project.



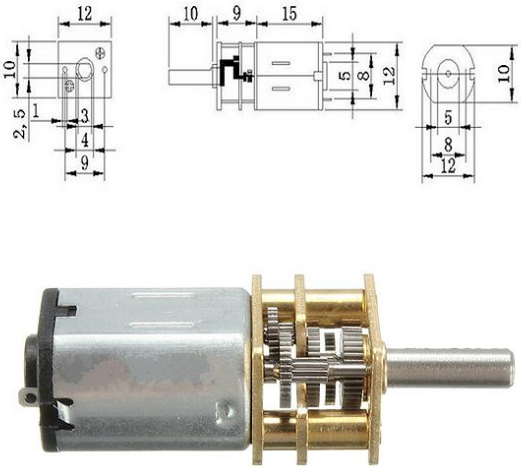
Motor Driver Module

L298N motor driver module was used in controlling the speed and direction of the motors. The module contains a double H-bridge configuration that is very efficient in controlling the motors and a heat sink to prevent overheating and ensure stability in performance. It is therefore a good, dependable module for continuous use in the line-following robot.



Motors:

We used 200 RPM metal gear motors to drive the wheels due to their provision of an appropriate balance between torque and speed. The motors feature smooth handling and constant outputs, making them adequate for efficient control for the line-following robot. Metal gears in the motors provide ruggedness, with small sizes allowing effective as well as stable design.



Power Supply

We powered the robot using three 3.7V 1800mAh rechargeable batteries, which offered a steady and effective power source. Their light weight ensures the mobility of the robot, and their rechargeable functionality ensures that they are an economical and eco-friendly option when used over a longer duration.

DEVELOPMENT STAGES

The line-following robot's design activity was organized, ranging from component choice to assembly, testing, and tuning. In every phase, emphasis was put on collective decision-making, problem-solving, and continuous improvement in refining the precision and efficiency of the robot for track following.

Component Selection and Planning

We began by conducting research and selecting essential components to build a stable and efficient line-following robot. A few of the essential components were an Arduino Uno for controlling, an HW-871 Five Sensor Array to detect lines, an L298N motor driver module to control speed and direction, 200 RPM metal gear motors for movement, and three 3.7V 1800mAh rechargeable batteries for power supply. Based on suggestions and previous testing, we chose a PCB dot board as the chassis, which gave us a lightweight but sturdy frame with evenly distributed weight.

Chassis Design and Assembly

To improve stability, we modified the chassis using a PCB dot board, chosen for its light weight and ease of customization. The components were positioned with significant regard to weight distribution, positioning them near the rear wheels so that the robot is stable when turning. The components were securely mounted on

the chassis using bolts and nuts, providing stability as well as easy modification or replacement if needed.

Sensor Testing and Calibration

The initial test round involved calibrating and testing a single sensor at a time to ensure accurate line detection. We experimented with having an array of IR sensors to provide line position feedback of the robot continuously. Various sensor arrays and placements were experimented and tested to obtain the best line detection accuracy, especially on curves and bends.

Coding and Control Systems

Control of the robot was implemented with a Proportional-Integral-Derivative (PID) control algorithm that allowed the robot to dynamically vary its speed based on sensor feedback. This allowed the robot to become more precise in line following, particularly eliminating deviation in straight lines and through curves. Through ongoing testing and tweaking of the code, we optimized the system to speed up on straight tracks and reduce speed on curves, optimizing the performance of the robot on the track.

Voltage Stabilization

In the testing, we observed that unstable battery voltage was making the motor driver module go haywire, leading to speed fluctuations. To counter this, we had the batteries connected in series so that they provided a stable voltage of 11.1 V. This adjustment stabilized the motor speed and improved the overall reliability of the robot.

Testing and Optimization

The final activity involved rigorous test runs of speed and following behavior of the robot. There were also fine tuning actions for even better performance like increasing speed straight and decreasing on the turns in order to provide the best overall completion time on the track. Ongoing tests and adjustments led us to learn a consistent repeatable performance under the needed speed and accuracy level.

Every phase of the design process contributed significantly to the refinement of the robot's line-following precision, stability, and speed. The project highlighted the significance of iterative testing, judicious component selection, and systematic problem-solving in being able to effectively achieve the design requirements.

CODE DEVELOPMENT

```
const int ENA = 10;  // PWM pin for Motor A
const int IN1 = 8;   // IN1 pin for Motor A (Forward)
const int IN2 = 9;   // IN2 pin for Motor A (Backward)

// Motor B
const int ENB = 11;   // PWM pin for Motor B
const int IN3 = 12;   // IN3 pin for Motor B (Forward)
const int IN4 = 13;   // IN4 pin for Motor B (Backward)

// New sensor pins
const int IRSensorLeft1 = 3;
const int IRSensorLeft2 = 4;
const int IRmiddle = 5;
const int IRSensorRight2 = 6;
const int IRSensorRight1 = 7;

// PID constants
float Kp = 40;
float Ki = 0;
float Kd = 28;

int baseSpeedL = 100; // Base speed for left motor
int baseSpeedR = 100; // Base speed for right motor

// PID variables
float error = 0, previousError = 0, integral = 0, derivative = 0;

void setup() {
    pinMode(ENA, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    pinMode(IRSensorLeft1, INPUT);
    pinMode(IRSensorLeft2, INPUT);
    pinMode(IRmiddle, INPUT);
    pinMode(IRSensorRight2, INPUT);
    pinMode(IRSensorRight1, INPUT);
    Serial.begin(115200);
}

void loop() {
    int s1 = digitalRead(IRSensorLeft1);
    int s2 = digitalRead(IRSensorLeft2);
    int s3 = digitalRead(IRmiddle);
    int s4 = digitalRead(IRSensorRight2);
    int s5 = digitalRead(IRSensorRight1);

    // Calculate error based on sensor readings
    error = (s1 * -4) + (s2 * -2) + (s3 * 0) + (s4 * 2) + (s5 * 4);

    // Reset integral on sharp turns
    if (abs(error) > 4) {
        integral = 0;
    }

    integral += error;
    derivative = error - previousError;
    float correction = (Kp * error) + (Ki * integral) + (Kd * derivative);

    int leftMotorSpeed = baseSpeedL + correction;
    int rightMotorSpeed = baseSpeedR - correction;

    // Slow down sharply on high error values for better cornering stability
    if (abs(error) > 4) {
        leftMotorSpeed *= 0.6;
        rightMotorSpeed *= 0.6;
    }

    // Ensure speeds are within valid range
    leftMotorSpeed = constrain(leftMotorSpeed, -150, 150);
    rightMotorSpeed = constrain(rightMotorSpeed, -150, 150);

    moveMotors(leftMotorSpeed, rightMotorSpeed);
    previousError = error;

    delay(10);
}

void moveMotors(int leftSpeed, int rightSpeed) {
    // Left Motor Control
    if (leftSpeed > 0) {
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
    } else if (leftSpeed < 0) {
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, HIGH);
        leftSpeed = -leftSpeed;  // Convert to positive for PWM
    } else {
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, LOW);  // Stop motor
    }

    // Right Motor Control
    if (rightSpeed > 0) {
        digitalWrite(IN3, HIGH);
        digitalWrite(IN4, LOW);
    } else if (rightSpeed < 0) {
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, HIGH);
        rightSpeed = -rightSpeed;  // Convert to positive for PWM
    } else {
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, LOW);  // Stop motor
    }

    // Apply PWM signals
    analogWrite(ENA, constrain(leftSpeed, 0, 150));
    analogWrite(ENB, constrain(rightSpeed, 0, 150));
}
```

Overview

The robot is controlled by a PID (Proportional-Integral-Derivative) control system, making the motor speeds and directions controllable to maintain the robot on course. The robot utilizes the HW-871 Five Sensor Array to detect the line, and the data from the sensors is processed to determine the error. The PID controller controls the motor speeds accordingly to move the robot to its position and maintain it on course. The code also has dynamic speed control, sensor calibration, and in-built windup prevention to enhance performance. Three series of 3.7V 1800mAh rechargeable batteries also provide a stable 11.1 V supply to maintain constant motor performance when working.

1.) setup()

When the robot is powered on, the setup() function runs once to get everything ready. It initializes the motors and sensors, making sure the system is prepared for operation. Additionally, it starts serial communication at 9600 baud for debugging purposes.

Key Tasks:

- Sets up the motor control pins (ENA, IN1, IN2, ENB, IN3, IN4) as outputs.
- Configures the sensor input pins (D3 to D7) to read data from the IR sensors.
- Starts serial communication to monitor sensor readings and debug the system.
- Calls the calibrate Sensors() function, which helps the sensor array recognize minimum and maximum values for more accurate readings.

2.) loop()

Once the setup is complete, the loop keeps running non-stop, making sure the robot follows the line smoothly. It does this by:

- Checking the sensor readings to see where the line is.
- Figuring out how far off-center the robot is (error calculation)
- Using a PID control system to make precise speed adjustments.
- Slowing down on sharp turns to stay stable.
- Calling moveMotors() to apply the right corrections.

3.) readSensors()

The IR sensors are like the robot's eyes, helping it see the track.

- Each sensor detects if it's over the black line or not.
- Depending on which sensors are activated, the robot calculates how far it has drifted from the center.
- The further off the track it is, the bigger the correction needed.

4.) PID Control

The PID controller is like a brain that keeps the robot steady and responsive. Here's how it works:

- Proportional (P): Reacts to how far off the robot is. The bigger the error, the stronger the correction.
- Integral (I): Helps with long-term corrections (but is set to zero here to avoid unnecessary drift).
- Derivative (D): Predicts how fast the error is changing and smooths out sudden movements.

If the error is too high (meaning a sharp turn is needed), the robot automatically slows down for better stability.

Innovative Aspects of the Current Code

1.) PID-Based Dynamic Control for Precision Navigation

Instead of using simple on-off logic, this robot implements a Proportional-Integral-Derivative (PID) controller to ensure smooth and precise movement.

The proportional term corrects the error based on how far the robot is from the center, the derivative term prevents sudden jerks, and the integral term (currently set to zero) can help correct long-term drift.

This advanced control system minimizes overshooting and oscillations, allowing the robot to handle sharp turns and complex paths effectively.

2.) Adaptive Speed Control for Sharp Turns

The code automatically slows down the robot when it detects a large positional error.

By multiplying speed by 0.6 during sharp turns, it prevents skidding and increases stability, which is especially useful for high-speed operation on curved tracks.

This feature makes the robot more intelligent and adaptable compared to traditional line-followers that maintain a constant speed.

3.) Weighted Sensor-Based Error Calculation

Instead of treating all sensors equally, the error is calculated using a weighted system:

- The outermost sensors have higher weights (-4 and +4).
- The inner sensors have lower weights (-2 and +2).
- The middle sensor is neutral (0).

This prioritizes central sensors while still allowing the robot to react when it drifts to the edges.

It enhances accuracy in following the line, reducing unnecessary oscillations.

4.) Modular and Expandable Code Structure

The code is structured in a modular way, making it easy to modify and expand:

Separate functions for sensor reading, PID computation, and motor movement.

If new features (like obstacle detection or Bluetooth control) need to be added, they can be integrated without major modifications.

This makes the system scalable for future enhancements, such as AI-based line-following or machine learning optimizations.

5.) Efficient Motor Control with PWM Optimization

The moveMotors() function intelligently controls motor direction based on speed values.

Instead of using full-speed ON/OFF switching, it applies Pulse Width Modulation (PWM) to fine-tune motor speeds.

This ensures precise speed variations rather than abrupt movements, which improves the overall performance of the robot.

6.) Real-Time Debugging with Serial Communication

The Serial.begin(115200); line enables high-speed serial communication for debugging.

Real-time data such as sensor readings, error values, and PID outputs can be printed to a serial monitor for performance analysis.

This allows engineers to fine-tune parameters quickly, making the development process more efficient.

TESTING

To fully assess the robot's efficiency, two tracks with varying complexity were put to test. The intention was to test the precision, velocity, and responsiveness of the robot across various track conditions.

1. **Track 1 - Straight Path Efficiency**
   The first track predominantly consisted of straight lines with minimal curves. This track was used to examine the speed, stability, and precision of the robot in following the line with minimal corrections. The test on this track provided a baseline of the robot's performance on basic, linear tracks.
2. **Track 2 - Complex Curves and Turns**
   The second path contained a number of tight bends and turns that challenged the robot's ability to manage speed, maintain direction, and react swiftly to speed and direction changes. The second path also tested the precision of the robot's sensors and motor control and assisted in analyzing deficiencies in tracking complicated paths.

By subjecting the robot to both tracks, we were able to discover a lot about its performance under different conditions. The experiments showed the robot's ability to balance speed and precision even when there is growing complexity in the tracks. The testing also gave avenues for improvement, such as sensor calibration optimization and motor control improvement to enable faster responsiveness in sharp turns.

DESIGN CHALLENGES AND SOLUTIONS

1.) **Material:** The line-following robot was initially built from a kit chassis, likely acrylic in material, which provided a robust and pre-formed structure. However, this material proved challenging to modify, limiting flexibility in design adjustment and component positioning.

Solution: The chassis was redesigned from a PCB Dot board, which offered greater customizability, less intricate integration of electronics, and a more compact design. This change allowed improved wiring management and weight and the maintenance of necessary structural integrity for robot operation.

2.) **Chassis Weight**: The robot was first built with a kit chassis, which is most likely to be acrylic. While being light and easy to build, it limited customization as well as non-optimization for the component layout.

Solution: The chassis was re-designed with an H-shaped PCB board, in which one of the rectangular parts is bigger than the other. This enhanced weight distribution so that motors and electronics could be placed more efficiently. The bigger part served as a stabilizer, while the smaller part eliminated unnecessary weight, making the robot more agile and performative.

3.) **Motor Selection**: The robot initially used two 3V-6V DC gear motors**.** However, these motors had limitations in terms of speed control and torque, leading to inconsistent movement and difficulties in maintaining a smooth trajectory along the line.

Solution: The motors were upgraded to 200 RPM metal gear motors, which provided greater speed stability, greater torque, and greater longevity. This upgrade provided greater smooth movement, greater control of turns, and overall better performance of the line-following robot.

4.) **Sensor Selection**: Initially, the project employed a QTR-8RC Reflectance Sensor Array with adequate detection ability. Its eight-sensor design, though, proved to be more complicated than it was needed for the track, and a few of its sensors remained unused, thus complicating calibration as well as processing.

Solution: A 5-sensor IR array substituted for the initial sensor system in a balanced trade-off between precision and effectiveness. It provided accurate line detection along with signal simplification of complexity in processing, which facilitated quicker response time and higher global navigation along the track.

5.) **Sensor Placement**: Another of the principal issues was positioning the sensor array in a manner that would pick up curves in the course early and provide accurate readings along the whole track. If the sensors were too distant from the line or too high on the chassis, they would potentially fail to read the line while navigating curves or sharp turns.

Solution: For this, the HW-871 Five Sensor Array was fitted at the front of the chassis, as close as possible to the track. The sensors were fitted lower down, giving improved detection of the line and allowing the robot to anticipate bends in the track. This improved the responsiveness and overall performance of the robot in following the line.

6.) **Sensor Calibration**: The HW 871 Five Sensor Array was utilized to detect the line through measuring reflected infrared signals. Varying ambient light, however, affected sensor reading, creating detection inconsistencies at times.

Solution: Calibration was performed to obtain the minimum and maximum values of all five sensors. This ensured that the system dynamically adjusted to accommodate different lighting conditions, with precise line detection. Normalizing sensor values maintained the robot in precise tracking and sensitive to path drift.

7.) **Motor Controller Changing:** Initially, we used an Drv8833 motor controller module to control the robot's motors. However, during testing, we changed the motor controller because we changed from Arduino Nano to Arduino Uno R3, leading to damage and rendering it unusable. The overheating issue highlighted the need for a more robust motor controller that could manage heat dissipation effectively.

Solution: To address this, we replaced the Drv8833 motor controller with an L298N module, which includes a built-in heat sink for improved thermal management. This upgrade enhanced the module's ability to dissipate heat, reducing the risk of overheating and allowing for more reliable and prolonged operation during testing.

8.) **Battery Selection:** The selection of the battery was very important in order to deliver power effectively without adding extra weight. A heavier battery would give more power but affect the speed and maneuverability of the robot. And also during the texting one battery overheated and we had to buy another one.

Solution: Three 3.7V 1800mAh rechargeable batteries in series were used to supply 11.1V, trading off power efficiency and lightness. This allowed for more than enough energy for extended use while maintaining a minimum weight for best performance. The rechargeable feature also allowed for repeated uses, reducing cost and environmental impact.

TECHNICAL SPECIFICATIONS

The line-following robot was designed with carefully selected components and configurations to ensure efficient line detection, stability, and guided navigation along the line. The following technical specifications encapsulate the robot's critical components:

1.) Microcontroller

Model: Arduino Uno

Function: The Arduino Uno is the main controller, controlling motor operations and interpreting sensor readings. It runs the PID control algorithm, varying the robot's speed and direction according to feedback from the sensor array to maintain precise line following.

2.) Sensors

Type: HW-871 Five Sensor Array

Function: The HW-871 Five Sensor Array detects the line by reading reflected infrared signals, allowing the robot to accurately identify its position on the path. It provides real-time feedback to the microcontroller, enabling precise adjustments for effective line following.

3.) Motor Driver Module

Model: L298N Motor Driver

Function: The L298N motor driver controls the speed and direction of the motors according to the instructions of the Arduino. It adds smoothness to the motion of the motor, particularly when it is turning, by accurate adjustment of the motor direction and speed.

4.) Motors

Type: 200 RPM Metal Gear Motors

Configuration: Two motors used for sole operation of left and right wheels

Function: Provides sufficient torque and speed to operate smoothly, allowing for effective control when going straight and in turns.

5.) Power Supply

Battery Type: Three 3.7V 1800mAh rechargeable lithium batteries stacked together to give a total of 11.1V.

Power Distribution: The batteries distribute power to the system, energizing both the motor driver and other components.

Voltage Stability: The 11.1V power supply ensures stable motor operation with sufficient voltage to ensure stable speed and reliability when in operation.

6.) Chassis

Material: PCB Dot Board

Configuration: H-shaped design with an asymmetrically larger side for increased stability. The motor and power modules are contained in the larger side, while the sensor array and control system are located in the smaller side. This configuration gives equal weight distribution, enhancing rear-wheel stability and enabling cornering more effectively.

7.) Control System

Algorithm: Proportional-Integral-Derivative (PID) Control

Function: The PID control algorithm is used to automatically adjust motor speed based on error detected by the sensor array. Speed is heightened during straight lines and reduced for turns, ensuring effective and accurate line following without instability

FUTURE DEVELOPMENT

Innovative Ideas to Enhance the Code

1.) **Automatic PID Tuning**

Idea: Implement an adaptive PID such that the code varies the Kp, Ki, and Kd values automatically based on track conditions, i.e., straights or turns.

Benefit: This would enable the robot to adapt its performance in real-time and behave in response to more intricate or dynamic track conditions, enabling smoother travel and more effective line-follower operations.

2.) **Fuzzy Logic Control Implementation**

Idea: Add a fuzzy logic controller as an alternative to traditional PID, where the code evaluates line-following performance using flexible "if-then" rules for various conditions.

Benefit: Fuzzy logic could handle track variations more smoothly, especially in cases where exact values are less effective, such as navigating sharp curves.

3.) **Obstacle Detection and Avoidance**

Idea: Add sensors (such as ultrasonic or proximity sensors) to detect objects in the robot's path and modify its movement to avoid collision.

Benefit: Such a feature would make the robot even more useful for use in real-life applications outside of line-tracking.

Innovative Ideas to Enhance the Hardware

1.) **Two IR Sensor Arrays for greater efficiency**

Idea: Two IR sensor arrays will be implemented to ensure improved efficiency of the robot. The first array will sense bends in advance, and the robot will be slowed down prior to the bend. The second array will be used to control speed variations to allow for smoother travel through bends. This configuration can readily decrease the time taken for path completion as well as the overall navigation accuracy.

Benefit: The robot will be quicker to respond with this approach, with better performance on tough courses with tight corners and complicated layouts.

2.) **Bluetooth Module for PID Tuning**

Idea: Add a Bluetooth module so PID constants can be tuned in real time with a contemporary smartphone app. PID constants (Kp, Ki, Kd) are tuned on a computer manually, which is slow. Tuning on the fly while testing with the Bluetooth module enables the fine-tuning process to be sped up.

Benefit: This will improve the testing process and be easier to PID tune so quick changes can be made based on conditions on the track.

3.) **Light Power Source and Chassis**

Idea: Utilize Lithium-Polymer (Li-Po) batteries, lighter than normal Lithium-Ion batteries, to keep the total weight of the robot as low as possible. The chassis shall be made with the best available materials like carbon fiber or lightweight alloys to keep the weight as low as possible.

Benefit: This will make the robot more agile, improve energy efficiency, and give a longer battery life with sufficient power to deliver at its best.

4.) **Real-Time Feedback Data with Wireless Communication**

Idea: Use wireless communication (i.e., Zigbee or Wi-Fi) to send real-time sensor feedback back to a monitor device (e.g., smartphone or laptop) in test mode. This can be employed to provide feedback regarding the performance and behavior of the robot and can be optimized and tuned in real time.

Benefit: Real-time data feedback enables quicker troubleshooting and optimization in the test mode, improving overall performance and reducing downtime.

5.) **Vision-Based Line Following:**

Upgrade from IR sensors to a camera with OpenCV for smarter navigation. The camera detects the line, adjusts the steering dynamically, and handles curves, broken lines, and varying surfaces more accurately. This makes the robot adaptable to different environments and reduces errors caused by sensor limitations. It also opens the door for future AI integration, allowing the robot to learn and improve its path-following skills over time.

These innovations collectively serve to make the project more efficient, flexible, and robust. They are evidence of careful design, careful planning, and good problem-solving techniques, all of which optimize the robot's overall performance and its ability to navigate difficult environments with accuracy.

BUDGET OF THE PROJECT

Item Name	Quantity	Unit Price	Price
Arduino Uno Revision 3	1	Rs.2600.00	Rs.2600.00
USB Type B cable,0.3m	1	Rs.250.00	Rs.250.00
5 Channel IR Tracking Sensor (HW-871)	1	Rs.980.00	Rs.980.00
12GA N20 200RPM 6V 6VDC Metal Gear Motor	2	Rs.1100.00	Rs.2200.00
MD0085-L298N Motor Driver Module	1	Rs.410.00	Rs.410.00
Jumper wire M/M 10CM	2	Rs.170.00	Rs.340.00
Jumper Wire M/F 10CM	2	Rs.160.00	Rs.320.00
Clear Glue Adhesive stick	2	Rs.30.00	Rs.60.00
Circuit wires 10/0.5mm	4	Rs.30.00	Rs.120.00
PCB Dot Board DIY	1	Rs.330.00	Rs.330.00
Caster Wheel	1	Rs.170.00	Rs.170.00
1800 mAh 3.7V Rechargeable battery	3	Rs.490.00	Rs.1470.00
Double Tape	1	Rs.290.00	Rs.290.00
Cello Tape (Small)	1	Rs.70.00	Rs.70.00
Battery Holder 18650-3	1	Rs.165.00	Rs.165.00
D-hole-Rubber-Wheel 43x19x3mm	2	Rs.170.00	Rs.340.00
N20-Motor-Seat Mounting Bracket	2	Rs.140.00	Rs.280.00
M3 16mm Phillips Screw	25	Rs.7.00	Rs.175.00
3.7V Rechargeable battery charger	1	Rs.700.00	Rs.700.00
Breadboard 400-Tie-Points	1	Rs.280.00	Rs.280.00
Insulation Tape Black	1	Rs.170.00	Rs.170.00
Screwdriver Phillips screw	1	Rs.210.00	Rs.210.00
Acrylic sheet Transparent	1	Rs.600.00	Rs.600.00
Paper cutter	1	Rs.150.00	Rs.150.00
Total			**Rs.12680.00**

REFERENCES

1. Miller, L. (2019). *Motor Controller Raspberry Pi & Arduino Configurations - Learn Robotics*. Learn Robotics. Available at: https://www.learnrobotics.org/blog/motor-controller-raspberry-piarduino-configurations/
2. Arduino (2021). *Arduino Uno*. Available at: https://www.arduino.cc/en/Main/ArduinoBoardUno [Accessed 14 Mar. 2025].
3. ProjectHub.arduino.cc. (n.d.). *Line Follower Robot (with PID controller)*. Available at: https://projecthub.arduino.cc/anova9347/line-follower-robot-withpid-controller-01813f
4. AllDatasheet (2020). *L298N Motor Driver Datasheet*. Available at: https://www.alldatasheet.com/view.jsp?Searchword=L298N [
5. ESCorner Channel. (n.d.). *PID LINE FOLLOWER ROBOT USING ARDUINO NANO*. [Online] Instructables. Available at: https://www.instructables.com/PID-LINE-FOLLOWER-ROBOT-USING-ARDUINO-NANO
6. Electronics Hub. (2021). *IR Sensor Array for Line Following Robots*. Available at: https://www.electronicshub.org/ir-sensor-array-for-line-following-robots