Nitte Meenakshi Institute of Technology
An Autonomous Institution under VTU, Belagavi
PB No. 6429, Yelahanka, Bengaluru 560-064, Karnataka, India

Mini Project Report

on

"LINE FOLLOWING ROBOT WITH OBSTACLE AVOIDANCE"

Submitted in partial fulfillment of the requirements for the award of degree of

MASTER OF COMPUTER APPLICATIONS

Of

Visvesvaraya Technological University (VTU)

By

Nishchitha M N – 1NT21MC063

Sachin R V – 1NT21MC084

Sakshith S Rao – 1NT21MC085

Shobitha U Nayak – 1NT21MC095

Shreelakshmi Narayan Bhat– 1NT21MC096

Vidya H– 1NT21MC109

Under the guidance of

Ms. Sowmya K

Assistant Professor

Dept of MCA, NMIT

Academic Year: 2022 - 23

Nitte Meenakshi Institute of Technology
Yelahanka, Bengaluru- 560064

Department of Master of Computer Applications

CERTIFICATE

This is to certify that **Nishchitha M N** *bearing* **1NT21MC063** *has completed her third semester Mini Project Work based on IOT entitled* ***"Line Following Robot with Obstacle Avoidance"*** *in partial fulfillment for the award of Master of Computer Applications degree, during the academic year 2022-2023 under my supervision.*

Signature of Internal Guide
Ms. Sowmya K
Assistant Professor
Department of MCA

Signature of HOD
Dr. Prasad N. H.
Dept. of MCA
NMIT

SEMESTER END EXAM

Examiners	**Signature with date**
1. ……..	**……..**
2……..	**……..**

Nitte Meenakshi Institute of Technology
Yelahanka, Bengaluru- 560064

Department of Master of Computer Applications

CERTIFICATE

This is to certify that **Sachin R V** *bearing* **1NT21MC084** *has completed his third semester Mini Project Work based on IOT entitled* ***"Line Following Robot with Obstacle Avoidance"*** *in partial fulfillment for the award of Master of Computer Applications degree, during the academic year 2022-2023 under my supervision.*

Signature of Internal Guide	**Signature of HOD**
Ms. Sowmya K	Dr. Prasad N. H.
Assistant Professor	Dept. of MCA
Department of MCA	NMIT

SEMESTER END EXAM

Examiners	**Signature with date**
1. ……..	**……..**
2……..	**……..**

Nitte Meenakshi Institute of Technology
Yelahanka, Bengaluru- 560064

Department of Master of Computer Applications

CERTIFICATE

This is to certify that **Sakshith S Rao** *bearing* **1NT21MC085** *has completed his third semester Mini Project Work based on IOT entitled* ***"Line Following Robot with Obstacle Avoidance"*** *in partial fulfillment for the award of Master of Computer Applications degree, during the academic year 2022-2023 under my supervision.*

Signature of Internal Guide	**Signature of HOD**
Ms. Sowmya K	Dr. Prasad N. H.
Assistant Professor	Dept. of MCA
Department of MCA	NMIT

SEMESTER END EXAM

Examiners	**Signature with date**
1. ……..	**……..**
2……..	**……..**

Nitte Meenakshi Institute of Technology
Yelahanka, Bengaluru- 560064

Department of Master of Computer Applications

CERTIFICATE

This is to certify that **Shobitha U Nayak** *bearing* **1NT21MC095** *has completed her third semester Mini Project Work based on IOT entitled* ***"Line Following Robot with Obstacle Avoidance"*** *in partial fulfillment for the award of Master of Computer Applications degree, during the academic year 2022-2023 under my supervision.*

Signature of Internal Guide	**Signature of HOD**
Ms. Sowmya K	Dr. Prasad N. H.
Assistant Professor	Dept. of MCA
Department of MCA	NMIT

SEMESTER END EXAM

Examiners	**Signature with date**
1. ……..	**……..**
2……..	**……**

Nitte Meenakshi Institute of Technology
Yelahanka, Bengaluru- 560064

Department of Master of Computer Applications

CERTIFICATE

This is to certify that **Shreelakshmi Narayan Bhat** *bearing* **1NT21MC096** *has completed her third semester Mini Project Work based on IOT entitled* ***"Line Following Robot with Obstacle Avoidance"*** *in partial fulfillment for the award of Master of Computer Applications degree, during the academic year 2022-2023 under my supervision.*

Signature of Internal Guide	**Signature of HOD**
Ms. Sowmya K	Dr. Prasad N. H.
Assistant Professor	Dept. of MCA
Department of MCA	NMIT

SEMESTER END EXAM

Examiners	**Signature with date**
1. ……..	**…….**
2……..	**…….**

Nitte Meenakshi Institute of Technology
Yelahanka, Bengaluru- 560064

Department of Master of Computer Applications

CERTIFICATE

This is to certify that **Vidya H** *bearing* **1NT21MC109** *has completed her third semester Mini Project Work based on IOT entitled* ***"Line Following Robot with Obstacle Avoidance"*** *in partial fulfillment for the award of Master of Computer Applications degree, during the academic year 2022-2023 under my supervision.*

Signature of Internal Guide	**Signature of HOD**
Ms. Sowmya K	Dr. Prasad N. H.
Assistant Professor	Dept. of MCA
Department of MCA	NMIT

SEMESTER END EXAM

Examiners	**Signature with date**
1. ……..	**…….**
2……..	**…….**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning the people who made it possible. With deep gratitude, we acknowledge all those guidance and encouragement, which served as beacon of light and crowned our efforts with success. We thank each one of them for their valuable support.

We express our kind thanks to the management of this Institution, Nitte Meenakshi Institute of Technology, Bengaluru for providing necessary facilities to carry out this project work successfully.

We express our kind thanks to our Principal, Dr. H.C Nagaraj, for providing necessary facilities and motivation to carry out the project work successfully.

We would like to express our sincere regards and heart full thanks to our HOD, Dr. Prasad N H, whose constant encouragement, guidance and support to carry out the project work successfully.

We would like to express our gratitude and humble thanks to our Guide, Ms. Sowmya K, for the constant encouragement, guidance and support to carry out the project work successfully.

We are grateful to all the teaching and the non-teaching staff of the Department of MCA, NMIT, Bengaluru, for providing us the support for completion of our work successfully.

Last but not the least; we would like to thank our parents and friends who have helped us directly and indirectly in all possible ways.

Nishchitha M N – 1NT21MC063

Sachin R V – 1NT21MC084

Sakshith S Rao – 1NT21MC085

Shobitha U Nayak – 1NT21MC095

Shreelakshmi Narayan Bhat– 1NT21MC096

Vidya H– 1NT21MC109

DECLARATION

I, **Nishchita M N,** student of III Semester of MCA, **Nitte Meenakshi Institute of Technology**, Bengaluru, bearing **1NT21MC063**, hereby declare that the project entitled "**Line Following Robot with Obstacle Avoidance"** has been carried out by me, under the supervision of, **Ms. Sowmya K, Assistant Professor,** and submitted in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** by the **Visvesvaraya Technological University** during the academic year **2022 - 2023**. This report has not been submitted to some other Organization/University for any award of degree or certificate.

Place: Bengaluru

Date:

Nishchita M N

1NT21MC063

DECLARATION

I, **Sachin R V,** student of III Semester of MCA, **Nitte Meenakshi Institute of Technology**, Bengaluru, bearing **1NT21MC084**, hereby declare that the project entitled "**Line Following Robot with Obstacle Avoidance"** has been carried out by me, under the supervision of, **Ms. Sowmya K, Assistant Professor,** and submitted in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** by the **Visvesvaraya Technological University** during the academic year **2022 - 2023**. This report has not been submitted to some other Organization/University for any award of degree or certificate.

Place: Bengaluru

Date:

Sachin R V

1NT21MC084

DECLARATION

I, **Sakshith S Rao,** student of III Semester of MCA, **Nitte Meenakshi Institute of Technology**, Bengaluru, bearing **1NT21MC085**, hereby declare that the project entitled "**Line Following Robot with Obstacle Avoidance"** has been carried out by me, under the supervision of, **Ms. Sowmya K, Assistant Professor,** and submitted in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** by the **Visvesvaraya Technological University** during the academic year **2022 - 2023**. This report has not been submitted to some other Organization/University for any award of degree or certificate.

Place: Bengaluru Sakshith S Rao

Date: 1NT21MC085

DECLARATION

I, **Shobitha U Nayak ,** student of III Semester of MCA, **Nitte Meenakshi Institute of Technology**, Bengaluru, bearing **1NT21MC095**, hereby declare that the project entitled "**Line Following Robot with Obstacle Avoidance"** has been carried out by me, under the supervision of, **Ms. Sowmya K, Assistant Professor,** and submitted in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** by the **Visvesvaraya Technological University** during the academic year **2022 - 2023**. This report has not been submitted to some other Organization/University for any award of degree or certificate.

Place: Bengaluru

Date:

Shobitha U Nayak

1NT21MC095

DECLARATION

I, **Shreelakshmi Narayan Bhat,** student of III Semester of MCA, **Nitte Meenakshi Institute of Technology**, Bengaluru, bearing **1NT21MC096**, hereby declare that the project entitled "**Line Following Robot with Obstacle Avoidance"** has been carried out by me, under the supervision of, **Ms. Sowmya K, Assistant Professor,** and submitted in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** by the **Visvesvaraya Technological University** during the academic year **2022 - 2023**. This report has not been submitted to some other Organization/University for any award of degree or certificate.

Place: Bengaluru Shreelakshmi Narayan Bhat

Date: 1NT21MC096

DECLARATION

I, **Vidya H,** student of III Semester of MCA, **Nitte Meenakshi Institute of Technology**, Bengaluru, bearing **1NT21MC109**, hereby declare that the project entitled "**Line Following Robot with Obstacle Avoidance"** has been carried out by me, under the supervision of, **Ms. Sowmya K, Assistant Professor,** and submitted in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** by the **Visvesvaraya Technological University** during the academic year **2022 - 2023**. This report has not been submitted to some other Organization/University for any award of degree or certificate.

Place: Bengaluru Vidya H

Date: 1NT21MC109

TABLE OF CONTENTS

TABLE OF FIGURES

1.INTRODUCTION

1.1 Problem Definition

Design and develop a Line Following Robot with Obstacle Avoidance using Arduino. The robot should be able to follow a black line on a white background while avoiding any obstacles in its path. The robot should be able to detect obstacles using sensors and make the necessary adjustments to avoid them while still following the line.

The robot should be able to perform the following tasks:

- Detect the black line using line sensors and follow it.
- Detect any obstacles in the path of the robot using obstacle sensors.
- Use the information obtained from the sensors to make the necessary adjustments to avoid the obstacles while still following the line.
- Use a motor driver to control the movement of the robot.
- Use an Arduino board to process the data obtained from the sensors and control the motor driver.
- Use a power source to provide the necessary power to the motor driver and the Arduino board.

1.2 Objectives of the Study

Designing and building a Line Following Robot with Obstacle Avoidance using Arduino requires clear objectives to ensure that the end product is effective and functional. The first objective is to design and build a robot that can accurately follow a black line on a white background. This involves selecting appropriate line sensors and programming the robot to respond to the data from these sensors.

The second objective is to equip the robot with obstacle sensors to enable it to detect any objects in its path. This will require careful selection of the sensors and programming of the robot to make the necessary adjustments to avoid the obstacles while still following the line.

The third objective is to program the Arduino board to process data from the sensors and control the motors to make the necessary adjustments. This will require a good understanding of programming concepts and techniques, as well as an ability to troubleshoot and debug code.

To ensure smooth operation of the robot, a motor driver must be used to control the movement of the robot. The motor driver must be carefully selected to ensure compatibility with the motors and provide sufficient power to operate the robot.

The chassis and frame of the robot are also critical components. The robot must be constructed in a way that is durable and reliable, with the ability to withstand the rigors of repeated use. This requires careful selection of materials and attention to detail in the construction process.

The power source is also an important consideration, as the motors and the Arduino board require a sufficient amount of power to operate for a reasonable amount of time. The power source must be selected carefully and must be able to provide the required amount of power without overheating or malfunctioning.

Testing is also an important part of the design and development process. The robot must be tested in various scenarios to ensure that it performs as expected and make any necessary modifications to improve its performance.

Finally, the design and development process should be well documented, and a user manual should be created to guide users on how to operate the robot effectively. This will ensure that the robot can be used by others and encourage the use of technology in solving real-world problems and promoting innovation.

1.3 Scope.

The scope of a Line Following Robot with Obstacle Avoidance using Arduino is broad and varied, with a range of potential applications in various industries. The primary scope of this robot is to provide an efficient and effective solution for navigating a path while avoiding obstacles. This could be useful in industries such as manufacturing, where automated robots are used for tasks, such as moving materials and products around the factory floor.

Another potential application is in the field of logistics, where robots can be used to move packages and products around a warehouse or storage facility. The ability to follow a line and avoid obstacles would be highly beneficial in this context, as it would reduce the risk of collisions and increase the efficiency of the process.

The scope of this robot is not limited to industrial applications, however. It could also be used in a range of educational settings, such as in robotics classes or as part of STEM programs. Building and programming a Line Following Robot with Obstacle Avoidance using Arduino is an excellent way to teach students about robotics and programming concepts, while also encouraging creativity and problem-solving skills.

The scope of this robot is also not limited to specific industries or applications. It could be adapted and modified to suit a range of different contexts and requirements, depending on the needs of the user. The versatility of this robot is one of its key strengths, as it can be customized to suit a range of different needs and applications.

In summary, the scope of a Line Following Robot with Obstacle Avoidance using Arduino is broad and varied, with potential applications in a range of different industries and educational settings. Its versatility and customizability make it an ideal solution for navigating a path while avoiding obstacles, and it has the potential to be adapted to suit a range of different contexts and requirements.

1.4 Methodology

The methodology for designing and building a Line Following Robot with Obstacle Avoidance using Arduino involves several key steps. The first step is to define the requirements and specifications of the robot, including the size, weight, power requirements, and the sensors and components required to achieve the desired functionality.

Once the requirements have been defined, the next step is to design the overall architecture of the robot, including the chassis, motor driver, and sensor placement. The design should take into account the requirements and specifications of the robot and ensure that all components are compatible and work together effectively.

The next step is to assemble the robot, which involves constructing the chassis and attaching the motors, sensors, and other components. Careful attention should be paid to the assembly process to ensure that all components are attached securely and in the correct orientation.

Once the robot has been assembled, the next step is to program the Arduino board to control the movement of the motors and process data from the sensors. This involves writing code to control the speed and direction of the motors, as well as code to read data from the line sensors and obstacle sensors.

Testing is a critical part of the methodology, and the robot should be tested in a range of different scenarios to ensure that it performs as expected. Testing should be done in a controlled environment, and any issues or problems should be identified and addressed before the robot is used in real-world applications.

Documentation is also an important part of the methodology, and the design and development process should be well-documented to ensure that others can replicate the process and build their own Line Following Robot with Obstacle Avoidance using Arduino. The documentation should include a detailed description of the design and development process, code samples, wiring diagrams, and other relevant information.

Finally, the methodology should include ongoing maintenance and support, to ensure that the robot remains functional and effective over time. This may involve replacing worn or damaged components, updating the code to fix bugs or improve performance, or providing support to users who have questions or issues with the robot.

In summary, the methodology for designing and building a Line Following Robot with Obstacle Avoidance using Arduino involves defining the requirements, designing the architecture, assembling the robot, programming the Arduino board, testing, documentation, and ongoing maintenance and support. Each of these steps is critical to the success of the project, and careful attention should be paid to each step to ensure that the robot is functional, effective, and reliable over time.

1.5 Hardware and Software Specifications

Hardware Requirements:

- **NODE MCU ESP8266**: NodeMCU is an open-source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). Strictly speaking, the term "NodeMCU" refers to the firmware rather than the associated development kits. Both the firmware and prototyping *board* designs are open source.

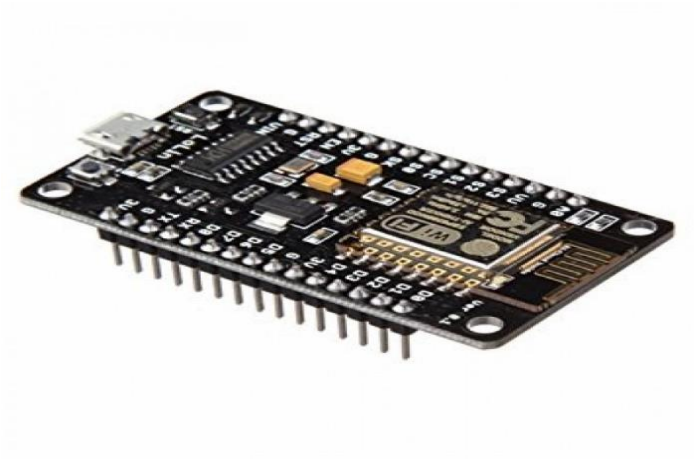

Figure 1.5. 1: NODE MCU ESP8266.

- **ARDUINO UNO:** The **Arduino Uno** is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc and initially released in 2010.[2][3] The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits.[1] The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable.[4] It can be powered by a USB cable or a barrel connector that accepts voltages between 7 and 20 volts, such as a rectangular 9-volt battery.

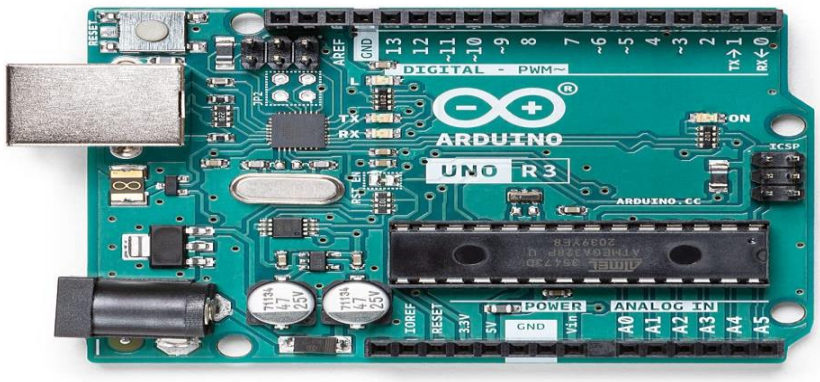

Figure 1.5. 2: Arduino UNO.

- **L298N MOTOR DRIVER SHIELD:** This Arduino motor drive shield allows driving two channel DC motors. It uses a L298N chip which deliveries output current up to 2A each channel. The speed control is achieved through conventional PWM which can be obtained from Arduino's PWM output Pin 5 and 6. The enable/disable function of the motor control is signalled by Arduino Digital Pin 4 and 7. The Motor shield can be powered directly from Arduino or from external power source. It is strongly encouraged to use external power supply to power the motor shield.

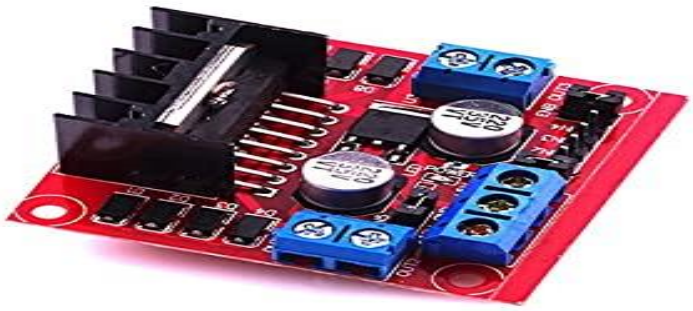

Figure 1.5. 3: *L298N MOTOR DRIVER SHIELD.*

- **DATA CABLE**: A **data cable** is any media that allows baseband transmissions (binary 1s and 0s) from a transmitter to a receiver.

Examples Are:

- Networking Media
- Ethernet Cables (Cat5, Cat5e, Cat6, Cat6a)
- Token Ring Cables (Cat4)
- Coaxial cable is sometimes used as a baseband digital data cable, such as in serial digital interface and thicknet and thinnet.
- Optical fibre cable; see fibre-optic communication
- Serial cable
- Telecommunications Cable
- Cat2 or telephone cord
- submarine communications cable
- Media device
- USB cable.

Figure 1.5. 4: Data Cable.

- **JUMPER WIRE**: Jumper wires are electrical wires used to connect different components of an electronic circuit board. They are usually made of insulated copper wire and have connectors or pins on both ends, which allow them to be easily plugged into a breadboard, a microcontroller board, or other electronic devices. Jumper wires are essential components in prototyping and testing circuits, as they allow quick and easy connections between different parts of a circuit without the need for soldering. They come in various lengths and colours, and are often used in combination with other electronic components such as resistors, capacitors, and sensors.

- **IR SENSOR**: An IR sensor, or an infrared sensor, is a device that detects and measures infrared radiation in the surrounding environment. Infrared radiation is a type of electromagnetic radiation that has a longer wavelength than visible light and is emitted by all objects with a temperature above absolute zero.IR sensors can detect the presence of objects or people based on the amount of infrared radiation they emit. They work by using a specialized detector, such as a thermopile or a pyroelectric sensor, which converts the incoming infrared radiation into an electrical signal that can be processed by a microcontroller or other electronic devi

Figure 1.5. 5: IR Sensor.

- **ULTRA SONIC SENSOR**: An ultrasonic sensor is a device that uses high-frequency sound waves to detect the presence of objects and measure their distance. The sensor emits a burst of high-frequency sound waves, which then bounce back from nearby objects and return to the sensor. By measuring the time it takes for the sound waves to return, the sensor can calculate the distance to the object. Ultrasonic sensors can be used in a variety of applications, such as obstacle detection, distance measurement, and level sensing. They are commonly used in automotive parking assist systems, industrial automation, robotics, and even in medical imaging.

Figure 1.5. 6: Ultra Sonic Sensor.

- **50W BATTERY**: A 50 watts battery is a type of rechargeable battery that has a power capacity of 50 watts. Wattage is a measure of the rate at which energy is transferred or consumed, and a 50 watts battery is capable of delivering 50 watts of power over a certain period of time, depending on the load and other factors. The actual capacity of a 50 watts battery will depend on its chemistry, size, and other factors. For example, a 12-volt lead-acid battery with a capacity of 4.2 amp-hours (Ah) can deliver a maximum power of around 50 watts for up to an hour, assuming no losses or other inefficiencies.

Figure 1.5. 7: 50W Battery.

- **POWER SWITCH**: A power switch is an electrical component used to turn an electrical circuit on or off. It is a simple and fundamental component that can be found in a wide range of electronic devices and appliances. Power switches can come in different types and configurations, such as rocker switches, toggle switches, push button switches, and rotary switches. They can also have different ratings and specifications, such as current and voltage ratings, and may be designed for different applications and environments.

Figure 1.5. 8: Power Switch.

- **9V BATTERY**: A 9V battery is a type of battery that provides 9 volts of direct current (DC) power. It is a compact and commonly used battery that is widely available

and used in various applications. 9V batteries are commonly used in small electronic devices, such as remote controls, smoke detectors, and other portable devices. They can also be used in larger devices, such as radios, amplifiers, and other audio equipment, where they are often used to power preamp circuits or other low-power components.

Figure 1.5. 9: 9V Battery.

- **DC MOTORS**: DC motors are electrical machines that convert direct current (DC) electrical energy into mechanical energy. They work based on the principle of Lorentz force, where a magnetic field interacts with an electric current to generate rotational motion. DC motors are widely used in various applications, from small hobbyist projects to industrial machines and equipment. They come in different sizes and configurations, from small toy motors to large industrial motors that can produce thousands of horsepower.

Figure 1.5. 10: DC Motors.

- **WHEELS:** Wheels are circular objects that rotate around an axle, and are commonly used to move vehicles and other machines. There are many different types of wheels, each with its own specific characteristics and uses.

Figure 1.5. 11: Wheels.

Software Requirements:

1. **Arduino IDE**: The Arduino Integrated Development Environment (IDE) is an open-source software application used for programming and developing Arduino microcontroller-based projects. It is a cross-platform application that works on Windows, Mac, and Linux operating systems. The Arduino IDE provides a simplified and user-friendly interface for writing, compiling, and uploading code to Arduino boards. It includes a code editor with syntax highlighting and auto-completion, a serial monitors for debugging and communication with the board, and a library manager for easy integration of third-party libraries.

2. **Thing Speak Cloud**: Thingspeak Cloud is an Internet of Things (IoT) platform that allows users to collect, store, and analyses data from connected devices. It is a cloud-based service that provides a simple and easy-to-use interface for managing IoT applications and devices. With Thingspeak Cloud, users can create channels to collect

data from various sources, such as sensors and other IoT devices. The data can be analysed, visualized, and shared with others through customizable charts, graphs, and maps. Users can also set up alerts and notifications based on predefined thresholds or conditions. Thingspeak Cloud offers a variety of features, including support for various communication protocols and APIs, integration with third-party services like MATLAB and IFTTT, and support for multiple user accounts with different levels of access.

2.LITERATURE SURVEY

2.1 Existing System

There are several existing systems for Line Following Robots with Obstacle Avoidance using Arduino. One popular example is the Pololu Zumo Robot, which is a small, low-cost robot designed for line following and maze solving. The Zumo Robot includes five infrared reflectance sensors for line following and two ultrasonic sensors for obstacle avoidance. It is controlled by an Arduino-compatible microcontroller and includes a pre-programmed library for easy programming.

Another example is the SparkFun RedBot Kit, which is a more customizable and expandable robot platform for robotics enthusiasts. The RedBot Kit includes a motor driver board, line following sensors, and a range of optional sensors for obstacle avoidance, such as ultrasonic and infrared sensors. It is also controlled by an Arduino-compatible microcontroller and can be programmed using the Arduino IDE.

The Cytron Maker Line is another existing system for Line Following Robots with Obstacle Avoidance using Arduino. This robot kit includes two infrared line following sensors and two ultrasonic sensors for obstacle avoidance, as well as a motor driver board and an Arduino-compatible microcontroller. It also includes a pre-programmed library for easy programming and customization.

Overall, these existing systems demonstrate the potential of Line Following Robots with Obstacle Avoidance using Arduino to provide efficient and effective solutions for navigating a path while avoiding obstacles. They provide a range of options for robotics enthusiasts and hobbyists to build and customize their own robots, and offer pre-programmed libraries for easy programming and customization.

2.2 Proposed System

Proposed systems for Line Following Robots with Obstacle Avoidance using Arduino include innovative designs and approaches that aim to improve the functionality and performance of the robot. For example, some proposed systems use machine learning algorithms to improve

the accuracy of the line following and obstacle avoidance capabilities of the robot. Others incorporate advanced sensors, such as LIDAR or stereo cameras, to provide more precise data and improve the robot's ability to navigate complex environments.

In terms of the overall system design, both existing and proposed systems typically involve a similar architecture, with a microcontroller, motor driver, sensors, and other components working together to control the movement of the robot. However, proposed systems often include more advanced components and more complex algorithms to improve the robot's performance and functionality.

In terms of the programming and code used in existing and proposed systems, there is often a range of different approaches and languages used. Some systems use C++ or Python to program the Arduino board, while others use more specialized languages or libraries designed specifically for robotics applications.

Overall, both existing and proposed systems for Line Following Robots with Obstacle Avoidance using Arduino demonstrate the potential of this technology to provide efficient and effective solutions for navigating a path while avoiding obstacles. As the technology continues to evolve and new approaches are developed, it is likely that we will see even more innovative and sophisticated systems in the future.

2.3 Future Enhancement:

The future scope for lane following robots and obstacle avoidance systems is vast, as these technologies continue to evolve and find new applications. Some possible areas of future development and research include:

1. **Improving accuracy and efficiency**: There is still room for improvement in the accuracy and efficiency of lane following robots and obstacle avoidance systems. Future developments may focus on enhancing the sensors and algorithms used in these technologies to achieve better performance.

2. **Integration with AI and machine learning**: The integration of artificial intelligence (AI) and machine learning (ML) could improve the performance and adaptability of lane following robots and obstacle avoidance systems. This could allow for more advanced decision-making and behaviour prediction capabilities.

3. **Use in autonomous vehicles**: Lane following robots and obstacle avoidance systems are critical components of autonomous vehicles. As the development of self-driving

cars continues to progress, the use of these technologies is expected to become more widespread.

4. **Applications in agriculture**: Lane following robots and obstacle avoidance systems have the potential to be used in agriculture for tasks such as crop monitoring, irrigation, and harvesting.
5. **Use in hazardous environments**: Lane following robots and obstacle avoidance systems could be used in hazardous environments such as nuclear plants, where human workers may be at risk.

2.4 Feasibility Study

A feasibility study for a line following robot with obstacle avoidance would typically be conducted in the early stages of the project to determine whether the project is viable from both a technical and economic standpoint. Here are some of the key factors that would need to be considered in such a study:

Technical feasibility:

- Line following mechanism: This would involve researching and evaluating different line following mechanisms, such as infrared or optical sensors, to determine which one would be most suitable for the specific application.
- Obstacle detection and avoidance: The feasibility study would need to determine how obstacles in the robot's path could be detected and avoided. This could involve evaluating different methods, such as ultrasonic or lidar sensors, to determine which one would work best.
- Power source: The study would need to evaluate different power sources, such as batteries or a power supply, to determine which one would be most appropriate for the robot.

Economic feasibility:

- Budget: The study would need to determine the budget for the project, including the cost of components, tools, and other resources that would be required to build the robot.

- Cost of materials: The feasibility study would need to research the cost of the materials required to build the robot, such as sensors, microcontrollers, and motors, to ensure that the project is financially viable.
- Labour cost: The study would need to estimate the labor cost involved in building and testing the robot to ensure that the project is economically feasible.
- Potential revenue: The feasibility study would need to identify potential revenue sources for the robot, such as selling it to customers or using it for internal purposes, to determine whether the project is economically viable.

Market feasibility:

- Market demand: The study would need to research the market demand for line following robots with obstacle avoidance in the target industry or application to determine whether there is sufficient demand for the product.
- Competitors: The study would need to identify existing competitors and evaluate their products and pricing to determine whether there is room in the market for a new product.
- Unique selling proposition: The study would need to determine the unique selling proposition of the robot and how it differentiates from existing products in the market to determine whether it is viable from a market standpoint.

Based on the findings of the feasibility study, the project team can make an informed decision about whether to move forward with building a line following robot with obstacle avoidance. If the project is deemed feasible, the team can develop a detailed project plan, including a timeline, budget, and resource allocation.

3. SYSTEM ANALYSIS AND DESIGN

3.1 Requirement Specification

1. System Overview

- The robot must be able to follow a designated lane while avoiding obstacles that may appear in its path.
- The robot must be autonomous, meaning that it must be able to navigate and make decisions without human intervention.
- The robot must have a camera and sensors to detect lanes and obstacles, respectively.
- The robot must be able to move at a steady speed, with the ability to slow down or stop when necessary.

2. Lane Following

- The robot must be able to detect the lane markings on the road and adjust its position accordingly to stay within the lane.
- The robot should be able to handle different types of lane markings, including solid lines, dashed lines, and arrows.
- The robot should be able to handle different types of road conditions, including curves, inclines, and declines.

3. Obstacle Avoidance

- The robot must be able to detect obstacles in its path and avoid them without leaving the designated lane.
- The robot should be able to handle different types of obstacles, including stationary objects, moving objects, and pedestrians.
- The robot must be able to make decisions in real-time to avoid collisions with obstacles.

4. **Control System**
    - The robot's control system should be robust and reliable, able to handle unexpected situations.
    - The robot's control system should be optimized for power efficiency to maximize battery life.
    - The robot's control system should be modular and extensible, allowing for easy updates and improvements.
5. **User Interface**
    - The robot should have a user interface that allows users to configure and monitor its behaviour.
    - The user interface should provide information on the robot's current status, including its speed, position, and obstacle detection.
    - The user interface should be intuitive and easy to use.
6. **Safety**
    - The robot should be designed with safety in mind, minimizing the risk of harm to people and property.
    - The robot should have a fail-safe mechanism that automatically stops the robot in case of a malfunction or unexpected behaviour.
    - The robot should be designed to operate within legal and ethical boundaries, following applicable laws and regulations.

3.2 Flowchart

Flowchart:

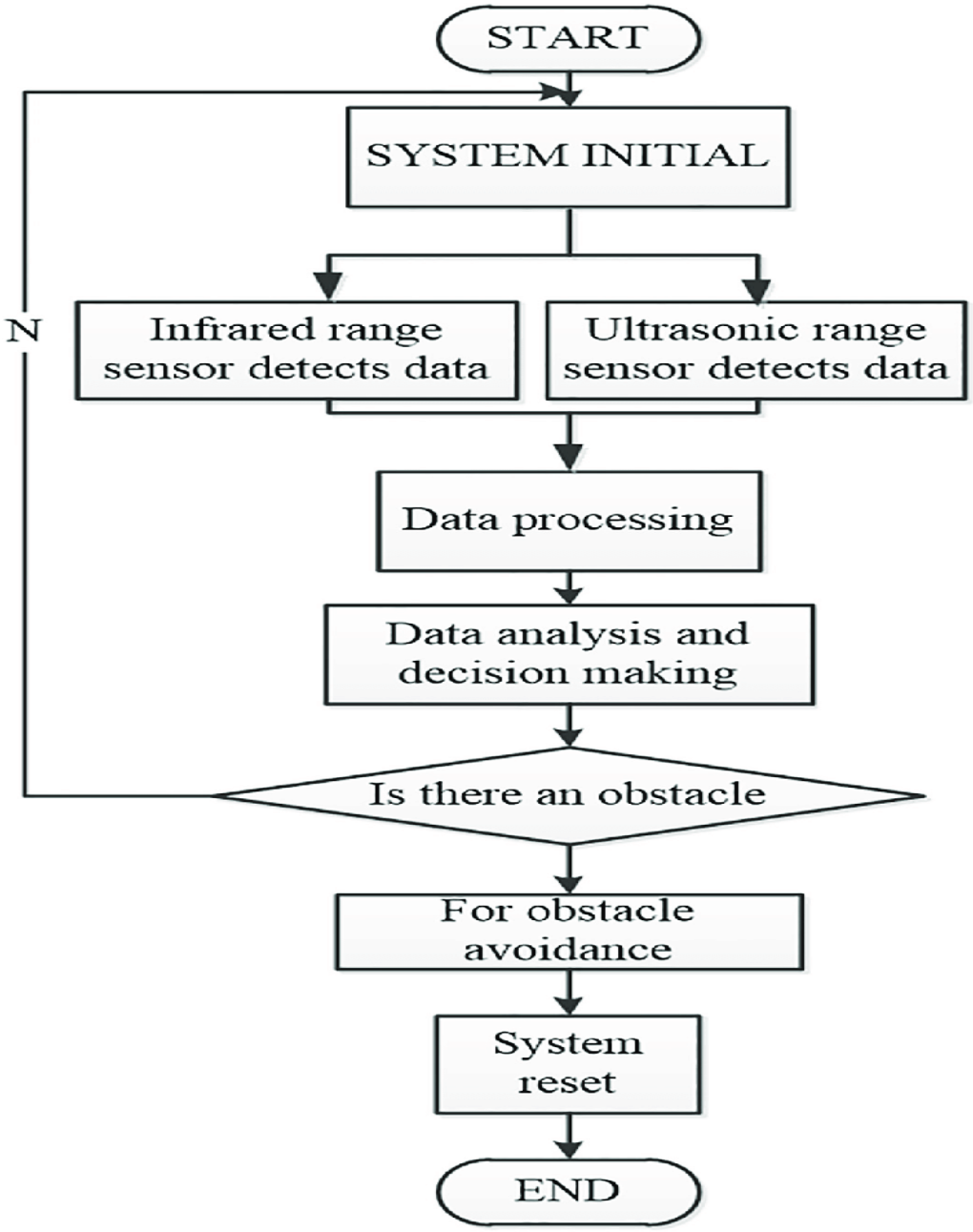

Figure 3.2.1: Flowchart of Line Following Robot with Obstacle Avoidance

3.3 Design and Criteria

Working:

These robots are pretty cheap and easy to design. Infrared Sensor is used to detect the black line on the path and Ultrasonic Sensor is used to detect obstructions on the path. The robot then responds to the sensor's reading and does something. This robot can follow a thick line of at least of 1 inch width perfectly and even follow the most complex path consisting of obtuse/acute angled turns and intersection of those black lines. When the two Infrared sensors connected at both sides of robot senses white path then the two motors rotate clockwise and the robot moves forward. Similarly, when both the Infrared sensors senses black path (signifying intersection of black lines) then also the two motors rotate clockwise and the robot moves forward. When one of the Infrared sensors (say the one located at the right side) senses a black path while the other one (left one) sense a white path, then the path is turning towards right, hence the robot turns right. To make the robot move right, the right motor stays stationary and left motor rotates clockwise, hence robot takes a right turn. When one of the Infrared sensors (say the one located at the left side) senses a black path while the other one (right one) sense a white path, then the path is turning towards left, hence the robot moves left. To make the robot move left, the left motor stays stationary and right motor rotates clockwise, hence robot takes a left turn. To take a sharp right turn, make the left motor rotate clockwise while making the right motor rotate anticlockwise. To take a sharp left turn, make the right motor rotate clockwise while making the left motor rotate anticlockwise. When the ultrasonic sensor in front of the robot senses any obstruction (in programmed range) while moving forward then the motors stops rotating and the robot stops. The robot starts moving as soon as the obstruction is removed.

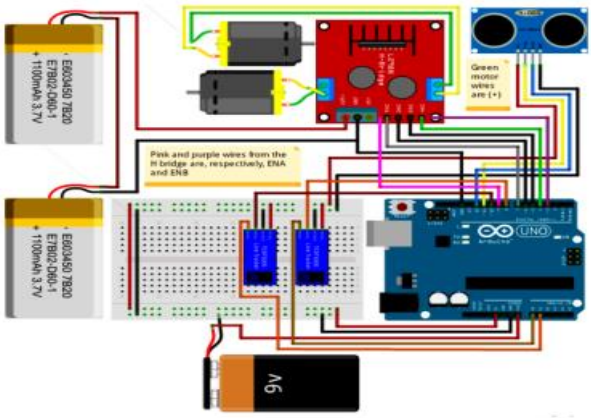

Figure 3.3. 1: Assembly

DESIGN:

1. **Chassis and Wheels:**
    - Choose a suitable chassis and wheels to support the weight of the robot and ensure stable movement.
    - Consider using wheels with good traction to ensure the robot can move on different types of road surfaces.
2. **Motor and Motor Controller:**
    - Choose a suitable motor and motor controller to control the speed and direction of the robot's movement.
    - Consider using a motor with high torque and speed control for accurate and smooth movement.
3. **Sensors:**
    - Choose a camera and sensors to detect the lane and obstacles.
    - Consider using a combination of sensors, such as ultrasonic sensors, LiDAR, or radar, to detect obstacles at different distances and heights.
4. **Microcontroller:**
    - Choose a microcontroller to process the data from the sensors and control the robot's movement.
    - Consider using a microcontroller with multiple input/output pins and a high clock speed for fast and accurate processing.
5. **Power System:**
    - Choose a suitable battery and power management system to provide power to the motor, sensors, and microcontroller.
    - Consider using a rechargeable battery with a high capacity and voltage, and a power management system with overvoltage and overcurrent protection.

6. **Obstacle Avoidance Algorithm:**
    - Implement an algorithm to detect obstacles and calculate an appropriate action to avoid them.
    - Consider using a combination of image processing and machine learning techniques to detect obstacles and make decisions in real-time.
7. **Lane Following Algorithm:**
    - Implement an algorithm to detect the lane and adjust the robot's steering to stay within the lane.
    - Consider using a combination of image processing and control theory techniques to accurately detect the lane and adjust the robot's steering.
8. **User Interface:**
    - Implement a user interface to monitor and configure the robot's behavior.
    - Consider using a display and buttons to show the robot's status and allow users to adjust its behaviour.
9. **Safety:**
    - Implement a fail-safe mechanism to stop the robot in case of a malfunction or unexpected behaviour.
    - Consider using physical or software-based mechanisms to stop the robot, such as an emergency stop button or a watchdog timer.

TEST STEPS:

1. **Lane following test:**
    - Place the robot on a straight lane with no obstacles.
    - Start the robot and verify that it can detect and follow the lane accurately.
    - Test the robot on different types of lanes, such as straight, curved, or dashed.

2. **Obstacle detection and avoidance test:**
    - Place an obstacle, such as a box or a wall, in front of the robot.
    - Start the robot and verify that it can detect the obstacle and calculate an appropriate action to avoid it.
    - Test the robot on different types of obstacles, such as stationary or moving objects, at different distances and heights.
3. **Integration test:**
    - Place the robot on a lane with obstacles.
    - Start the robot and verify that it can detect and follow the lane while avoiding obstacles.
    - Test the robot on different types of lanes and obstacles to ensure reliable performance.
4. **Speed and performance test:**
    - Measure the speed of the robot and verify that it can move at a consistent and stable speed.
    - Test the robot on different types of road surfaces, such as concrete, asphalt, or gravel, to ensure stable movement.
5. **User interface test:**
    - Verify that the user interface displays the robot's status and allows users to adjust its behaviour, such as speed or obstacle detection threshold.
    - Test the user interface on different types of devices, such as a display or a mobile app.
6. **Safety test:**
    - Verify that the fail-safe mechanism can stop the robot in case of a malfunction or unexpected behaviour.
    - Test the fail-safe mechanism in different scenarios, such as loss of communication or sensor malfunction.

CRITERIA:

1. **Accuracy:**
    - The robot should be able to detect and follow the lane accurately, with minimal deviation or overshooting.
    - The robot should be able to detect obstacles and calculate an appropriate action to avoid them with high accuracy.
2. **Speed:**
    - The robot should be able to move at a consistent and stable speed, with good acceleration and deceleration control.
    - The robot should be able to adjust its speed to follow the lane and avoid obstacles at different speeds.
3. **Robustness:**
    - The robot should be able to operate reliably in different types of environments, such as different road surfaces, lighting conditions, and weather conditions.
    - The robot should be able to handle different types of obstacles and avoid collisions in real-time.
4. **Scalability:**
    - The robot should be able to handle different types of lanes and obstacles, with the ability to adapt to new situations and road conditions.
    - The robot should be able to work with different types of sensors and microcontrollers, with the ability to integrate new components and algorithms as needed.
5. **User interface:**
    - The robot should have a user-friendly interface that displays the robot's status and allows users to adjust its behaviour.

- The user interface should be accessible on different types of devices, such as a display or a mobile app.

6. **Safety:**

- The robot should have a fail-safe mechanism that can stop the robot in case of a malfunction or unexpected behaviour.
- The robot should have physical or software-based mechanisms to prevent collisions with objects or people, such as emergency stop buttons or sensors.

3.4 Algorithms and Pseudocode

Algorithm

1. Import required libraries: SoftwareSerial, ESP8266WiFi, WiFiClient, and ThingSpeak.
2. Define pins for the ultrasonic sensor (pingPin and echoPin), infrared sensors (IR_SENSOR_RIGHT and IR_SENSOR_LEFT), and motor control pins (enableRightMotor, rightMotorPin1, rightMotorPin2, enableLeftMotor, leftMotorPin1, and leftMotorPin2).
3. Define a constant for the motor speed (MOTOR_SPEED).
4. Define a function named rotateMotor that takes two integer parameters: rightMotorSpeed and leftMotorSpeed. This function is responsible for rotating the motors based on the provided speed values.
5. In the setup function, set the pins as input/output, initialize the serial communication with a baud rate of 9600, and connect to the WiFi network.
6. In the loop function, calculate the distance using the ultrasonic sensor and store it in the variable cm.
7. Send the distance value to the ESP8266 module over SoftwareSerial.
8. If the distance is less than 25 cm, stop the motors by calling the rotateMotor function with speed 0.
9. Read the values of the two infrared sensors and store them in variables rightIRSensorValue and leftIRSensorValue.

10. If both sensors detect white, move forward by calling the rotateMotor function with speed MOTOR_SPEED.
11. If the right sensor detects black, turn right by calling the rotateMotor function with rightMotorSpeed = MOTOR_SPEED and leftMotorSpeed = -MOTOR_SPEED.
12. If the left sensor detects black, turn left by calling the rotateMotor function with rightMotorSpeed = -MOTOR_SPEED and leftMotorSpeed = MOTOR_SPEED.
13. If both sensors detect black, stop the motors by calling the rotateMotor function with speed 0.
14. Read data from the serial port and store it in the variable alldata1.
15. If the value of alldata1 is less than 25, set the data variable to "Stop".
16. Send the distance value and the data variable to ThingSpeak using the ThingSpeak library.
17. Repeat the loop.

Pseudocode

```
#include <SoftwareSerial.h>
SoftwareSerial espSerial(11, 12);
const int pingPin = 2;
const int echoPin = 3;
long duration;
int distance;


#define IR_SENSOR_RIGHT 4
#define IR_SENSOR_LEFT 13
#define MOTOR_SPEED 200

//Right motor
```

```
int enableRightMotor=10;
int rightMotorPin1=9;
int rightMotorPin2=8;
//Left motor
int enableLeftMotor=5;
int leftMotorPin1=7;
int leftMotorPin2=6;
void setup()
{
  pinMode(pingPin, OUTPUT);
  pinMode(echoPin, INPUT);
  TCCR0B = TCCR0B & B11111000 | B00000010 ;
  // put your setup code here, to run once:
  pinMode(enableRightMotor, OUTPUT);
  pinMode(rightMotorPin1, OUTPUT);
  pinMode(rightMotorPin2, OUTPUT);
  pinMode(enableLeftMotor, OUTPUT);
  pinMode(leftMotorPin1, OUTPUT);
  pinMode(leftMotorPin2, OUTPUT);
  pinMode(IR_SENSOR_RIGHT, INPUT);
  pinMode(IR_SENSOR_LEFT, INPUT);
  rotateMotor(0,0);
  Serial.begin(9600);
  espSerial.begin(9600);
}
void loop()
{
  long duration, inches, cm;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  cm = microsecondsToCentimeters(duration);
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  String str1=String("distance:= ")+String(cm);
espSerial.println(str1);

if(cm < 25)
{
 rotateMotor(0, 0);
  }
  int rightIRSensorValue = digitalRead(IR_SENSOR_RIGHT);
  int leftIRSensorValue = digitalRead(IR_SENSOR_LEFT);

  //If none of the sensors detects black line, then go straight
  if (rightIRSensorValue == LOW && leftIRSensorValue == LOW)
  {
    if(cm > 25)
    {
   rotateMotor(MOTOR_SPEED, MOTOR_SPEED);
    }
  }
  //If right sensor detects black line, then turn right
```

```
  else if (rightIRSensorValue == HIGH && leftIRSensorValue == LOW )
  {
     rotateMotor(MOTOR_SPEED, -MOTOR_SPEED);
  }
  //If left sensor detects black line, then turn left
  else if (rightIRSensorValue == LOW && leftIRSensorValue == HIGH )
  {
     rotateMotor(-MOTOR_SPEED, MOTOR_SPEED);
  }
  //If both the sensors detect black line, then stop
  else
  {
    rotateMotor(0, 0);
  }
 }
void rotateMotor(int rightMotorSpeed, int leftMotorSpeed)
{
  if (rightMotorSpeed < 0)
  {
    digitalWrite(rightMotorPin1,LOW);
    digitalWrite(rightMotorPin2,HIGH);
  }
  else if (rightMotorSpeed > 0)
  {
    digitalWrite(rightMotorPin1,HIGH);
    digitalWrite(rightMotorPin2,LOW);
  }
  else
  {
    digitalWrite(rightMotorPin1,LOW);
```

```
  digitalWrite(rightMotorPin2,LOW);
 }
 if (leftMotorSpeed < 0)
 {
  digitalWrite(leftMotorPin1,HIGH);
  digitalWrite(leftMotorPin2,LOW);
 }
 else if (leftMotorSpeed > 0)
 {
  digitalWrite(leftMotorPin1,LOW);
  digitalWrite(leftMotorPin2,HIGH);
 }
 else
 {
  digitalWrite(leftMotorPin1,LOW);
  digitalWrite(leftMotorPin2,LOW);
 }
 analogWrite(enableRightMotor, abs(rightMotorSpeed));
 analogWrite(enableLeftMotor, abs(leftMotorSpeed));
}
long microsecondsToCentimeters(long microseconds)
{
 return microseconds / 29 / 2;
}
```

***Obstacle avoidance**:*

```
#include <ESP8266WiFi.h>;
#include <WiFiClient.h>;
#include <ThingSpeak.h>;
const char* ssid = "Nish"; //Your Network SSID
const char* password = "12345678"; //Your Network Password
```

```
WiFiClient client;
unsigned long myChannelNumber = 2038673; //Your Channel Number (Without Brackets)
const char * myWriteAPIKey = "P9YIGFAU48HHO59D"; //Your Write API Key
int alldata1;
String data;
void setup() {
WiFi.begin(ssid, password);
ThingSpeak.begin(client);
// Open serial communications and wait for port to open:
Serial.begin(9600);
while (!Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
}
void loop() { // run over and over
if (Serial.available()) {
Serial.write(Serial.read());
 alldata1= Serial.read();
}
if(alldata1 < 25)
{
  data = "Stop";
}
ThingSpeak.writeField(myChannelNumber, 1, alldata1, myWriteAPIKey); //Update in
ThingSpeak
ThingSpeak.writeField(myChannelNumber, 2, data, myWriteAPIKey); //Update in
ThingSpeak
}
```

4.RESULT AND OUTPUT

4.1 Result:

The performance and results of a lane following robot with obstacle avoidance will depend on various factors, such as the quality of the hardware and software components, the accuracy of the camera input, and the effectiveness of the machine learning algorithms used.

Here are some of the expected results that we aim for when building a lane following robot with obstacle avoidance:

1. **Accurate lane detection**: The robot should be able to accurately detect the lanes and follow them without deviating from the path.
2. **Obstacle avoidance**: The robot should be able to detect obstacles in its path and navigate around them, without colliding or getting stuck.
3. **Smooth and steady movement**: The robot should be able to move smoothly and steadily, without jerky or erratic movements that could cause it to lose balance or deviate from the path.
4. **Real-time performance**: The robot should be able to process the camera input and respond to changes in the environment in real-time, without significant delays or lag.
5. **Robustness and reliability**: The robot should be able to operate in different environments and conditions, and withstand minor shocks or collisions without malfunctioning.

4.2 Output:

The robot moves forward until it detects an obstacle within 25cm using an ultrasonic sensor. Then it stops the motors, waits for a while and resumes moving forward. Additionally, the robot follows a black line using the IR sensors. If none of the sensors detects a black line, then it moves straight. If the right sensor detects a black line, it turns right, and if the left sensor detects a black line, it turns left. If both sensors detect a black line, then it stops.

The code starts with the inclusion of the SoftwareSerial library and the declaration of the pins used for the ultrasonic sensor, IR sensors, and motor driver. The setup function initializes the pin modes and sets the motor speed to 0. The loop function runs continuously and executes the following steps:

1. Measure the distance using the ultrasonic sensor
2. Send the distance value to the ESP8266 module using the SoftwareSerial library
3. If the distance is less than 25cm, stop the motors
4. Read the IR sensor values and execute the appropriate motor commands to follow the black line
5. If no black line is detected, and the distance is greater than 25cm, move forward at a constant speed

The rotateMotor function takes two arguments, rightMotorSpeed and leftMotorSpeed, and sets the motor direction and speed based on the sign of the input values. If the speed is negative, the motor rotates in reverse. If it is positive, the motor rotates forward. If it is zero, the motor stops.

The microsecondsToCentimeters function converts the duration of the pulse received from the ultrasonic sensor to centimeters using the formula distance = duration / 29 / 2. This formula is based on the speed of sound in air (approximately 340 m/s) and the fact that the pulse travels twice the distance between the sensor and the obstacle.

Overall, the code controls a simple robot that can detect obstacles and follow a black line. It demonstrates how to use the ultrasonic sensor, IR sensors, and DC motor driver with the Arduino. However, the code could be improved by adding more sophisticated obstacle avoidance and line

Figure 4.2. 1: Robot Detecting and moving towards the black line.

For Obstacle Avoidance we have used libraries like ESP8266WiFi.h, WiFiClient.h, and ThingSpeak.h to enable the microcontroller to connect to a Wi-Fi network and send data to a ThingSpeak channel.

The code starts with the declaration of the Wi-Fi network SSID and password that the microcontroller will connect to. These are defined as constants using the 'const' keyword. The 'WiFiClient' object is then created which will be used to establish a connection to the Wi-Fi network.

Next, the ThingSpeak channel number and write API key are defined. These are used by the ThingSpeak library to identify the specific channel where the data will be sent.

In the 'setup()' function, the Wi-Fi network is connected to using the 'WiFi.begin()' function, and the ThingSpeak library is initialized using the 'ThingSpeak.begin()' function. Serial communication is also initialized using the 'Serial.begin()' function at a baud rate of 9600.

In the 'loop()' function, the microcontroller constantly checks if there is any data available on the serial port using the 'Serial.available()' function. If data is available, it is read using the 'Serial.read()' function and stored in the 'alldata1' variable. The 'Serial.write()' function is also used to echo back the received data to the serial monitor.

If the value of 'alldata1' is less than 25, the 'data' string is set to "Stop". This is done to indicate that the data is outside the expected range and should be ignored.

The 'ThingSpeak.writeField()' function is used to send the data to ThingSpeak. It takes four arguments: the channel number, the field number (in this case, 1 or 2), the value to be sent, and the write API key. The first call to this function sends the 'alldata1' value to the first field of the ThingSpeak channel. The second call sends the 'data' string to the second field of the channel.

This process repeats indefinitely in the 'loop()' function, allowing the microcontroller to constantly receive and send data as long as it remains powered on and connected to the Wi-Fi network

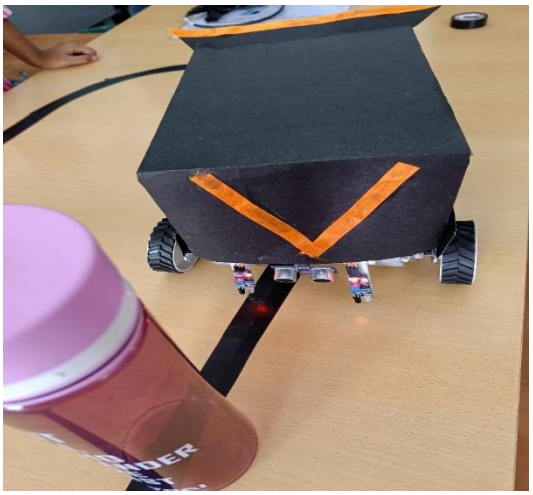

Figure 4.2. 2:Robot is stopped when it detected obstacle

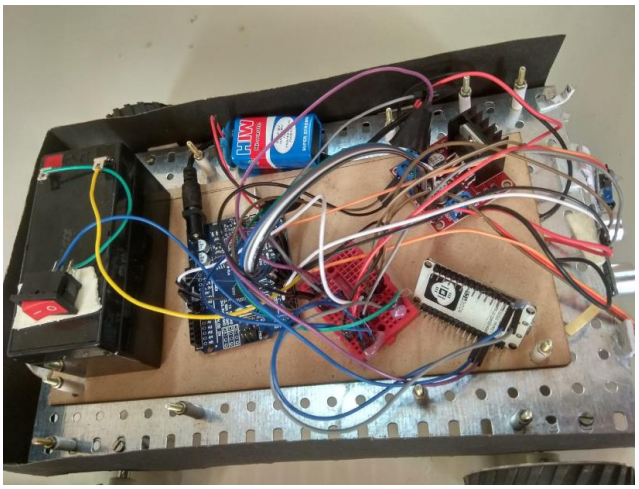

Figure 4.2. 3:Top View of Line Following Robot With Obstacle Avoidance

5.CONCLUSION

In conclusion, a lane following robot and obstacle avoidance system are both important applications of robotics that have numerous real-world applications.

A lane following robot uses sensors to detect and follow a line or path, which can be useful in industrial settings for automated transportation or in the development of self-driving cars. It can also be used in educational settings to teach basic robotics concepts to students.

Obstacle avoidance systems use sensors to detect objects in the path of a robot or vehicle and take action to avoid collisions. This technology is essential for autonomous vehicles, industrial robots, and other applications where safety is a concern.

When combined, a lane following robot with an obstacle avoidance system can be used to create a more advanced and efficient robotics system. For example, a warehouse robot could use a lane following system to navigate through aisles while an obstacle avoidance system detects and avoids any obstacles in its path.

Overall, the development and integration of these technologies are important for advancing the field of robotics and creating more advanced and capable robotic systems.

6.REFERENCES

1. S. Sarker, S. B. M. Ali, and T. Islam, "Line Following Robot: A Survey of Techniques," International Journal of Computer Applications, vol. 107, no. 10, pp. 32-38, December 2014.
2. L. Zhao, H. Wang, Y. Jiang, and Y. Zhang, "A Novel Approach for Line Following and Obstacle Avoidance of Mobile Robot," International Journal of Control and Automation, vol. 9, no. 4, pp. 45-54, 2016.
3. S. Bora, S. S. Sahu, and S. S. Mahapatra, "Design and Development of Obstacle Avoidance Robot Using Ultrasonic Sensor," International Journal of Advanced Research in Computer Science and Electronics Engineering, vol. 6, no. 2, pp. 244-247, 2017.
4. T. H. Nguyen, H. D. Nguyen, and H. H. Nguyen, "An Obstacle Avoidance System for Mobile Robots Based on Kinect Sensor," in Proceedings of the International Conference on Control, Automation and Information Sciences (ICCAIS), pp. 129-134, 2018.
5. Y. Zou, J. Liu, Y. Zhang, and Y. He, "Obstacle Avoidance Control for Autonomous Mobile Robots Based on Real-Time Path Planning," in Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA), pp. 1902-1907, 2019.
6. "Line Following Robot with PID Control" by Rui Santos: https://randomnerdtutorials.com/line-following-robot-with-pid-control/
7. "Autonomous Robots: From Biological Inspiration to Implementation and Control" by George A. Bekey: https://www.amazon.com/Autonomous-Robots-Biological-Inspiration-Implementation/dp/0262025788

8. "Obstacle Avoidance in Autonomous Vehicles: A Survey" by Qian Wang et al.: https://ieeexplore.ieee.org/abstract/document/8007479
9. "Obstacle Avoidance System for Mobile Robots" by M. Saritha and K. R. N. Reddy: https://www.researchgate.net/publication/322739987_Obstacle_Avoidance_System_for_Mobile_Robots
10. "A Comparative Study on Different Approaches of Obstacle Avoidance in Mobile Robots" by Anwarul Hasan et al.: https://www.researchgate.net/publication/341714618_A_Comparative_Study_on_Different_Approaches_of_Obstacle_Avoidance_in_Mobile_Robots
11. "Design and Implementation of a Line Follower Robot" by M. A. Islam et al.: https://ieeexplore.ieee.org/document/6753083
12. "Autonomous Obstacle Avoidance Robot" by T. N. Nasser and M. M. Hassan: https://ieeexplore.ieee.org/document/7016972.