KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

COLLEGE OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

PROJECT TOPIC:

Obstacle Avoidance and Line Follower Robot using Arduino

PROJECT SUPERVISOR

Prof. James Ben Hayfron-Acquah

STUDENTS

Solomon Nortey - 9408419

Kwaku Bandoh Amankwah – 9394919

August 2023

DECLARATION

We, Kwaku Bandoh Amankwah and Solomon Nortey confirm that this documentation and the project “Obstacle Avoidance and Line Follower Robot using Arduino” presented in it are our own achievement under the supervision of Prof. James Ben Hayfron-Acquah.

……..... ……..…….

Date Signature

(Kwaku Bandoh Amankwah)

……..….. ……..…….

Date Signature

(Solomon Nortey)

……..... ……..…….

Date Signature

(Prof. James Ben Hayfron-Acquah)

ACKNOWLEDGEMENT

We are most grateful to God Almighty for favour, grace and protection. Our deepest appreciation to our project supervisor, Prof. James Ben Hayfron-Acquah, the HOD of Computer Science Department, KNUST for his guidance, unwavering support and encouragement. Their insights and advice played a pivotal role in shaping the direction of this project.

A heartfelt thank you to families and friends for their patience, understanding, and constant encouragement throughout this endeavor.

We extend our appreciation to our anonymous peer reviewers whose constructive feedback and insights greatly improved the quality of this project.

This project would not have been possible without the collective effort and support of these individuals and entities. Their contributions have been instrumental in its successful completion.

Thank you all for your unwavering support and belief in this project.

ABSTRACT

In an era marked by the rapid advancement of automation and robotics, the development of intelligent robotic systems capable of navigating complex environments has become a pressing necessity. This project presents the design, implementation, and performance evaluation of an Obstacle Avoidance and Line Follower Robot using Arduino, which combines the capabilities of both obstacle avoidance and line-following in a single autonomous robotic platform.

The goal of this project is to build a robot car that can identify obstructions in its path while it is travelling and adjust path when necessary without the assistance of an outside force. The path that has the greatest separation between the obstacle and the sensor is the new direction that needs to be traveled in order to prevent collision, and the robot decides this based on sensor inputs.

The robot will be equipped with an infrared sensor array to detect the black line on the ground, and an ultrasonic sensor to detect obstacles. A microcontroller will be used to process the sensor data and generate control signals for the robot's motors. . Additionally, the motor driver shield regulates the motion of the servo motor (for sensor movement) and the DC motors (for wheel movement) to enable the obstacle avoidance feature. The Arduino microcontroller chip receives the instructions, which are used to direct the sensor and movement of the robot car.

Table of Contents

DECLARATION ..... **I**

ACKNOWLEDGEMENT ..... **II**

ABSTRACT ..... **III**

CHAPTER 1: INTRODUCTION **1**

1.1 BACKGROUND 1

1.2 PROBLEM STATEMENT 1

1.3 MOTIVATION 2

1.4 AIM AND OBJECTIVES 2

1.5 METHODOLOGY 3

1.6 PROJECT SCOPE 3

1.7 PROJECT LIMITATIONS 3

1.8 ACADEMIC AND PRACTICAL RELEVANCE OF THE PROJECT 5

1.9 PROJECT ACTIVITY PLANNING AND SCHEDULES 5

CHAPTER 2: LITERATURE REVIEW **6**

2.1 ARTIFICIAL INTELLIGENCE 6

2.2 ROBOTICS AND ROBOTS 6

2.3 ROBOT LEARNING 7

2.4 AUTONOMOUS ROBOT 7

2.5 ARDUINO 8

2.6 PROCESSES OF EXISTING SYSTEMS 8

CHAPTER 3: METHODOLOGY **10**

3.2 ROBOT WORKING PRINCIPLE 10

3.3 DEVELOPMENT TOOLS 10

3.3.1 HARDWARE COMPONENTS 10

3.3.2 SOFTWARES NEEDED 31

3.3.3 PROGRAMMING LANGUAGE 31

3.3.4 OTHER NECESSARY TOOLS AND MACHINES 31

3.4 HARDWARE SETUP 31

3.5 PROGRAMMING THE ROBOT 33

3.6 ROBOT MOVEMENT ON PATH 33

3.7 UNIFIED MODELING LANGUAGE (UML) DIAGRAMS 36

3.7.1 Sequence diagram of the program 36

3.7.2 Use case Diagram of the program 36

3.7.3 Flowchart of the program 37

3.7.4. Activity diagram of the program 38

3.7.5 DBML diagram of the program 39
3.8 SOFTWARE IMPLEMENTATION 39
3.8.1 ARDUINO PROGRAMMING 39
3.8.2 PROJECT METHOD: AGILE APPROACH 40
3.8.3 FUNCTIONAL REQUIREMENTS 41
3.8.4 NON-FUNCTIONAL REQUIREMENTS 41
3.9 CODE 42
CHAPTER 4: RESULTS AND RECOMMENDATIONS 52
4.1 RESULTS 52
4.2 RECOMMENDATION 53
CHAPTER 5: CONCLUSION 54
5.1 CONCLUSION 54
REFERENCES 55

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

In our world today, robotics is a very interesting research area, which is fast growing as it is the simplest way for modifying modern day technology. Since robotics is crucial to the development of technology, we chose to work in this area and create intelligent robots that would make life easier for people.

An autonomous robot is able to navigate independently in an unstructured environment that it is unfamiliar with. The robot's ability to assess its surroundings, recognize any impediments in its route, and navigate its environment by dodging them is due to the software intelligence that has been included into it. There are numerous robotic designs that can be used when creating an autonomous robot. The physical environment in which the robot will be operated must be taken into account while deciding on the design to be used. Robotic automobiles, drones, and walking robots are a few examples of autonomous robots.

The obstacle avoiding robot is intelligent enough to cover as much of the available space as possible. It also contains an ultrasonic sensor that it uses to identify any obstructions in its path before moving in a different direction to get around them.

The primary goal of such technology is that it can play a significant part in modern transportation by being utilized to prevent accidents, which typically occur on congested routes, by deploying emergency brake. If this technology is implemented in a car, it will automatically detect any objects (living or inanimate) in its route and, if necessary, apply breaks or veer to the side into a vacant area.

1.2 PROBLEM STATEMENT

Obstacle avoidance is a prerequisite for any autonomous mobile robot. This obstacle avoidance capability is critical in a robot's navigation system in an unfamiliar region to avoid collisions during operation. An autonomous robot must avoid collisions in order to avoid damaging the item or the robot itself. Obstacle avoidance is required in applications such as automated vacuum cleaners and aircraft. Even in robots that work in a familiar environment with a well-defined path, some environmental changes may occur that because the robot to collide with an object in its path, so the robot must be able to adapt to the change by avoiding any objects in

its path. The necessity for a robot that can identify and avoid items on a pre-computed route, as well as things that come unexpectedly, stems from the difficulty of effective trajectory planning. The solution to this trajectory problem involves the robot using sensors to identify and avoid obstructions, making the robot more autonomous because it does not require external intervention.

1.3 MOTIVATION

The concept of an autonomous robot is not novel. Every organization that utilizes mobile robots for duties wants the robot to be able to fulfill its activities successfully without the need for external control. With advancements in GPS technology, autonomous robot mobility is becoming a possibility. However, before using the robot, the issue of how the robot perceives and interacts with its surroundings must be solved. To solve this worry, sensors are utilized to collect enough environmental data for the robot to comprehend in order for the robot to navigate smoothly.

1.4 AIM AND OBJECTIVES

The Aim of this project is to design and implement a robot car that is able to move round an unknown environment without running into obstacles in its path. The Objectives of the project are as follows:

- The robot car should have the capacity to detect obstacles in its path based on a predetermined threshold distance.
- After detection of an obstacle, the robot should be able to change its direction to a relatively open path by making an autonomous decision.
- The robot car should not require any external control during its operation.
- The robot car should be able to measure distance between itself and an obstacle in real time.
- The robot car should be able to operate effectively in an environment which is unknown to it.

1.5 METHODOLOGY

- Attach four motors on acrylic sheet and sold the wires.
- On the other side of the acrylic sheet mount Arduino.
- Mount motor driver on the top of Arduino then connect wires of motor driver.
- Attach servo motor at front of the car and placed sensor holder on it.
- Mount ultrasonic sensor on holder then connect servo motor to the Motor Shield.
- The servo 1 input is connected to the Arduino Digital10 input.
- The servo 2 input is connected to the Arduino Digital9 input.
- Power for the Servos comes from the Arduino's on board 5V regulator, powered directly from the USB or DC power jack on the Arduino.
- Then the battery is connected to a leg of the button. Connect a cable to the other legs of the button. This cable is for Motor Shield (+)
- Attach the wheels.

1.6 PROJECT SCOPE

This project will involve designing a chassis for the robot, programming the Arduino Uno to control its movements, integrating sensors for obstacle detection and line following, testing its ability to navigate through obstacles and follow a line.

1.7 PROJECT LIMITATIONS

1. **Limited Sensor Accuracy:** IR sensors and ultrasonic sensors have limitations in terms of accuracy and precision. They might not provide highly accurate distance measurements or reliable obstacle detection in all scenarios, leading to occasional false positives or negatives.
2. **Limited Terrain Adaptability:** The robot's obstacle avoidance and line following capabilities might not work optimally on all types of terrains. Uneven surfaces, complex patterns, textures or reflective surfaces can pose challenges for both line following and obstacle detection.

3. **Motor Control Complexity:** Coordinating four TT DC motors and stepper motors simultaneously requires careful control and synchronization. Complex maneuvers might lead to difficulties in maintaining precise movement control.
4. **Speed Limitations:** The motors used might have speed limitations, affecting the robot's ability to respond quickly in certain situations, especially during obstacle avoidance. Drop in voltage can cause change in speed and give inconsistent results.
5. **Power Consumption:** Operating multiple motors and sensors, along with wireless communication through the Bluetooth module, can lead to higher power consumption. This might impact the robot's overall runtime and the need for frequent battery recharges.
6. **Noise and Interference:** The sensor readings might be affected by environmental factors such as noise, interference, or varying lighting conditions, leading to inconsistent behavior in different settings.
7. **Bluetooth Range:** The HC-05 Bluetooth module's communication range is limited, potentially limiting the range at which the robot can be monitored remotely.
8. **Data Transfer Speed:** While Bluetooth provides wireless communication, the data transfer speed might not be sufficient for real-time monitoring or transferring large datasets quickly.
9. **Maintenance and Durability**: Over time, the robot's components might undergo wear and tear, affecting its reliability and requiring periodic maintenance especially the TT DC gearbox motors.
10. **Cost:** Depending on the quality of components used, the cost of building such a robot with multiple sensors, motors, and communication modules can be relatively high.
11. **Limited Processing Power:** The processing power of the Arduino Uno might impose limitations on the complexity of algorithms that can be implemented.
12. **Conflicting timer values:** Some of the libraries have conflicting timer values during compilation so errors are generated during.

1.8 ACADEMIC AND PRACTICAL RELEVANCE OF THE PROJECT

Academic relevance

This project is a wonderful instructional resource for students studying robotics, mechatronics, or electronics. It gives them the opportunity to apply their academic knowledge to a real-world project. Sensor integration, control systems, programming, and signal processing are some of the principles it may teach. It also allows you to learn about microcontroller programming with the Arduino platform.

Practical relevance

This project may be used as a prototype platform for constructing more complex robotics applications, making it valuable for engineers and innovators. The skills learned in building an obstacle avoidance and line follower robot are directly applicable in the field of industrial automation, such as home automation, where robots can be programmed to perform tasks like vacuuming or lawn mowing while avoiding obstacles, and robots that can follow crop rows and avoid obstacles can be used for tasks like automated weeding or harvesting.

1.9 PROJECT ACTIVITY PLANNING AND SCHEDULES

Below is a generalized project plan with activities and estimated timeframes.

Activity	Duration (week(s))
Project Initiation and Planning	1
Electronics and Sensor Integration	1
Programming and Testing	2
Integration and Final Testing	2
Documentation	1
Enhancement and Advanced Features	2
Total Project Duration	**10**

CHAPTER 2: LITERATURE REVIEW

2.1 ARTIFICIAL INTELLIGENCE

Artificial Intelligence (AI) is the ability of a computer or a computer controlled-robot to think and perform tasks like humans. An artificially intelligent machine is one which is capable of learning, reasoning, planning, perception, problem solving. This field was discovered on the claim that human intelligence can be sufficiently described to the point that a machine can simulate it.

2.2 ROBOTICS AND ROBOTS

Robotics is an interdisciplinary research area at the interface of computer science and engineering. Robotics involves the design, construction, operation, and use of robots. The goal of robotics is to design intelligent machines in order to help human beings in their daily activities. This technology has resulted in automated machines that can replace humans in manufacturing processes or dangerous environments. These robots have numerous structures depending on their functions. Generally, robots are grouped into:

- Manipulator robots (industrial robots)
- Mobile robots (autonomous vehicles)
- Self-reconfigurable robots, which are robots that can adjust themselves based on the task to be performed.

Robots may be designed to act on their decision-making ability or to be controlled by humans. Robots work by trying to mimic/replicate the human behaviour as they are made to possess the same components of human beings. These components include:

- A muscle system in order to move the body structure
- A body structure.
- A power source used to activate sensors and muscles
- A sense system used to obtain environmental information
- A brain system which processes the sensed information and gives the muscles information on how to respond

2.3 ROBOT LEARNING

Robot learning involves the combination of robotics and machine learning by the learning of different algorithms by the robot in order to apply techniques which will enable it to obtain skills and adjust to its surrounding. The robot learning can take place by imitation of humans or by self-learning guided by a human.

2.4 AUTONOMOUS ROBOT

An Autonomous robot is one that is capable of performing activities with a high level of self-control and without any form of externa control. This kind of robot is achieved by integrating artificial intelligence, robotics, and information engineering.

Autonomous robots, just like human beings, additionally can settle on their own choices and afterward respond accordingly. A genuinely autonomous robot is one that can sense and recognize its environment, settle on choices dependent on what it sees and afterwards, actuate a movement in response. Regarding mobility, for instance, the decision-based activities incorporate but however are not restricted to the following: starting, stopping, and manoeuvring around obstructions that are in their path.

Some mobile robots utilize ultrasound sensors to detect obstacles or infrared. These sensors work in similar fashion to animal echolation. The robot sends a beam of light or a sound signal out and observes the reflection distance by identifying how long it takes the signal to bounce back.

In some advanced robots, stereo vision is used. This method uses two cameras for depth perception and image recognition in order to detect and categorize different objects.

In more advanced robots, they can be able to study new environments and adjust to them. They even work on areas with uneven land as they can link land patterns with certain actions. For example, a rover robot constructs a land map by utilizing its visual sensors and if the map depicts uneven land patterns, the robot can adapt to the environment and choose to take a different path. Such kind of systems can be highly beneficial for exploration and can be used for operation on other planets.

2.5 ARDUINO

Arduino is a widely used open-source prototyping platform which is based on software and hardware that are easy to use. It is made up of a programmable microcontroller unit and the Arduino Integrated Development Environment (IDE) which is used to write and upload codes to the Arduino board. The Arduino board is the physical microcontroller boards used for making electronic projects. Arduino boards are capable of reading inputs and converting them into outputs. You can send instructions to the microcontroller on the Arduino board and tell it what to do using the Arduino programming language (based on Wiring). Due to the open-source nature of the Arduino environment, codes are easily written and uploaded to the Arduino board. Arduino is written in Java so it can be run on Linux, Mac OSX and windows.

The key features of Arduino include:

- The Microcontroller boards are capable of detecting signals (analog or digital) from different sensors and convert the input into an output by triggering a response such as activating a motor, turning a LED on or off, and many other actions.
- The microcontroller board functions can be controlled by sending instructions to the microcontroller using the Arduino Integrated Development Environment.
- An external piece of hardware for loading code into the board is not requires unlike in some other circuit boards. Just a USB cable is required.
- It is simple to learn the Arduino programming language as it is a simplified version of C++.
- Arduino allows for a standard form factor, which breaks the functions of the board to a more accessible package.

2.6 PROCESSES OF EXISTING SYSTEMS

The following steps are commonly included in the present system of an Obstacle Avoidance and Line Follower Robot utilizing Arduino:

- The robot detects and follows a line on the ground using sensors. It moves along a predetermined course, modifying its motions based on sensor data.

- The robot has sensors that identify obstructions in its route. When an impediment is recognized, the robot alters course to avoid it and continue going forward. Obstacle is detected, the robot changes its direction to avoid the obstacle and continue moving forward.
- Based on specified algorithms, the Arduino board evaluates sensor input and makes decisions. It calculates whether the robot should follow the line or avoid obstacles based on sensor input.
- The Arduino board regulates the speed and direction of the robot's motors. It delivers the orders required to drive the robot forward, backward, or turn.

To successfully explore its surroundings, the system integrates line following and obstacle avoidance skills. It achieves its functionality through the use of sensor data, decision-making algorithms, and motor control.

2.7 BENEFITS OF IMPLEMENTATION OF THE PROJECT

- The robot can explore its surroundings automatically, following lines and avoiding obstacles without continual human interaction.
- The robot's capacity to properly follow lines and avoid obstacles aids in the productivity of jobs such as warehouse automation, assembly line procedures, and even house cleaning.
- The robot can travel securely in dynamic situations by including obstacle avoidance skills, lowering the danger of mishaps or accidents.
- The Arduino platform enables for programming and customization flexibility, allowing the robot to adapt to different line-following and obstacle avoidance conditions.
- Building and programming the robot using Arduino allows for hands-on learning in robotics, electronics, and coding, while also encouraging creativity and problem-solving abilities.
- Arduino-based projects are frequently less expensive than more complicated robotic systems, making them suitable for educational or small-scale applications.

CHAPTER 3: METHODOLOGY

OVERVIEW

In this chapter, we will discuss the methodology employed to design, build, and implement an obstacle avoidance and line following robot using the specified components. The methodology outlines the step-by-step process followed to achieve the objectives of the project.

3.2 ROBOT WORKING PRINCIPLE

The robot uses the Ultrasonic sensor to measure the distance in front of it then it moves. As the distance reduces, the robot interprets it as the presence of an obstacle. As soon as the robot detects the obstacle, it stops and moves back a few cm then looks left and right before moving to a free path.

3.3 DEVELOPMENT TOOLS

3.3.1 HARDWARE COMPONENTS

- Arduino Uno
- L239D motor shield driver
- HC-SR04 Ultrasonic sensor
- HC-05 Bluetooth Module
- 2 IR sensor
- Servo motor
- Passive buzzer
- Rechargeable Lithium battery
- Battery management system (BMS)
- 5v Power bank
- 4 Wheel Robot Car Chassis (Includes Motors)
- Jumper wires

ARDUINO UNO

An Arduino is a microcontroller development board that is open source. The Arduino Uno is the most popular version. When compared to other development boards, it is very inexpensive, connects directly into a computer's USB connection, and is simple to set up and operate.

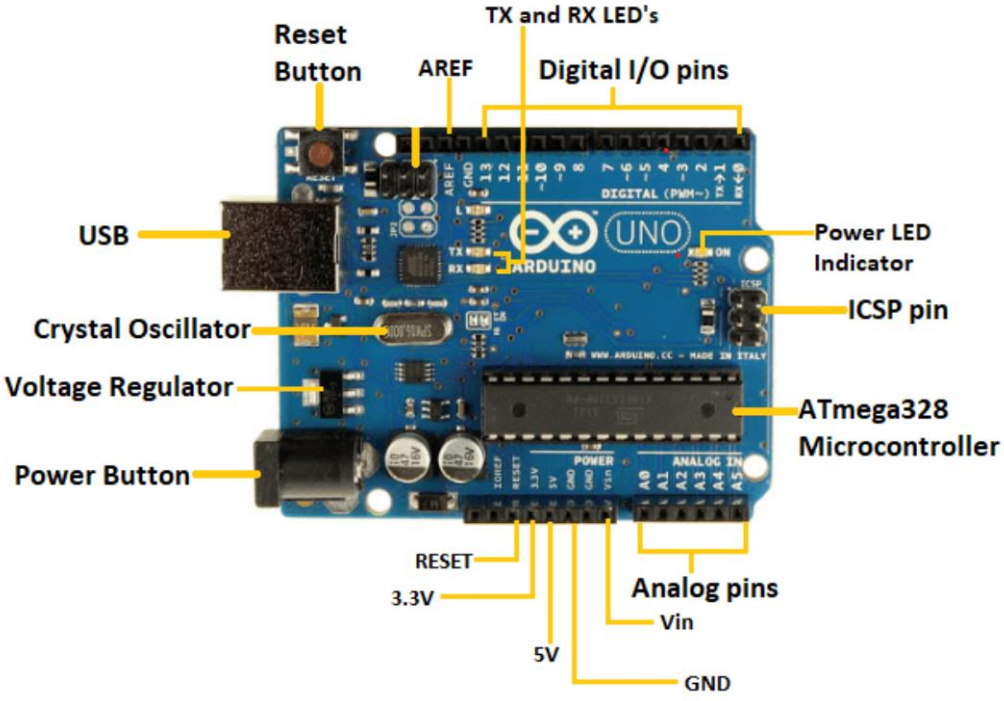


Fig. 1 Labeled parts of an Arduino Uno

Some of the key features of the Arduino Uno include:

- Open source design. Large community at arduino.cc/forum/ of people using and troubleshooting it.
- Easy USB interface. The chip on the board plugs straight into your USB port and registers on your computer as a virtual serial port. This allows us to serially communicate which is an extremely easy protocol
- Convenient power management and built-in voltage regulation. 12v can easily be regulated to both 5v and 3.3v
- Easy-to-find and cheap, microcontroller
- Countless number of hardware features like timers, PWM pins, external and internal interrupts, and multiple sleep modes.

Components on the Board

Power (USB / Barrel Jack): Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack.

GND: Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.

The pins on the Arduino are the places where you connect wires to construct a circuit. They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

5V & 3.3V: the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.

Analog: The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.

Digital: Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

PWM: You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM).

AREF: Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button: Just like the original Nintendo, the Arduino has a reset button. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times.

Power LED Indicator: Just beneath and to the right of the word "UNO" on the circuit board, there's a tiny LED next to the word 'ON'. This LED should light up whenever you plug your

Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

TX RX LEDs: TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data.

Main IC: The black thing with all the metal legs is an IC, or Integrated Circuit. Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's. This can be important, as you may need to know the IC before loading up a new program from the Arduino software.

Voltage Regulator: The voltage regulator is not actually something you can interact with on the Arduino. The voltage regulator does exactly what it says, it controls the amount of voltage that is let into the Arduino board.

L293D MOTOR SHIELD DRIVER

The 4-channel L293D Motor Driver Shield is a hardware module commonly used to control multiple DC motors and servo motors in robotics and automation projects. This shield is particularly popular in the Arduino ecosystem and can be interfaced with the Adafruit Motor Shield library for convenient control. The Arduino's L293D motor driver shield is compatible with Arduino Mega and UNO. The shield can be fitted on top of the Arduino board. The shield is built around the **L293D chip**, which is a **dual H-bridge motor driver IC**. Each H-bridge can control the direction and speed of a DC motor by applying varying voltage levels to its inputs. The shield has four of these H-bridges, allowing it to control up to four DC motors independently.

For each of the four DC motors, the shield provides two control pins: one for setting the motor's direction (typically labeled as 'MOTORx_A' and 'MOTORx_B') and another for controlling the motor's speed through PWM (Pulse Width Modulation) signal. The Adafruit Motor Shield library abstracts the control of these pins, allowing you to command the motors to move forward, backward, or stop, as well as control their speed.

In addition to the four DC motors, the shield can also control a servo motor. The servo motor is a different type of motor that is often used for precise angular control, such as in robotic arms or for controlling steering mechanisms. The shield provides a dedicated servo header for connecting the servo motor.

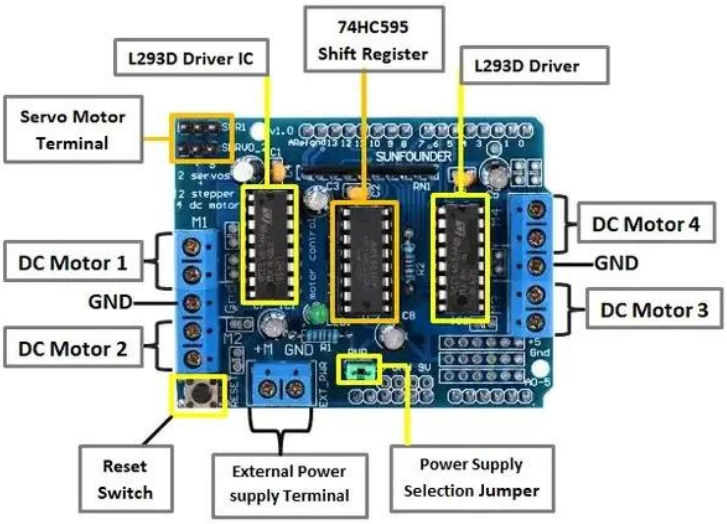


Fig. 2 Labeled parts of a motor shield driver

The **Adafruit Motor Shield library** offers easy-to-use functions for controlling both the DC motors and the servo motor. You can set the direction and speed of each DC motor using methods like setSpeed() and run() with arguments like FORWARD, BACKWARD, and RELEASE. For the servo motor, the library provides functions to set the angle of rotation. Under the hood, the library generates the appropriate control signals for each motor based on the commands you give it. It handles the PWM signals for speed control and the logic signals for direction control, allowing you to focus on the higher-level aspects of your project without needing to worry about the low-level hardware details.

When the shield is embedded on the Arduino Uno,

- Six analog pins including pins 2 and 13 remain free for use.
- To control the servo motor, use:

  Pins 2, 9, and 10.

- To control the DC motors, use:

  Pin 11 for the motor port M1

Pin 3 for the motor port M2

Pin 5 for the motor port M3

Pin 6 for the motor port M4

Pins 4, 7, 8, and 12 may also all be used

- To control the stepper motors, use:

  Pins 11 and 3 for the motor port M1-M2

  Pins 5 and 6 for the motor port M2-M3

  Pins 4, 7, 8, and 12 may also all be used

L293D CHIP

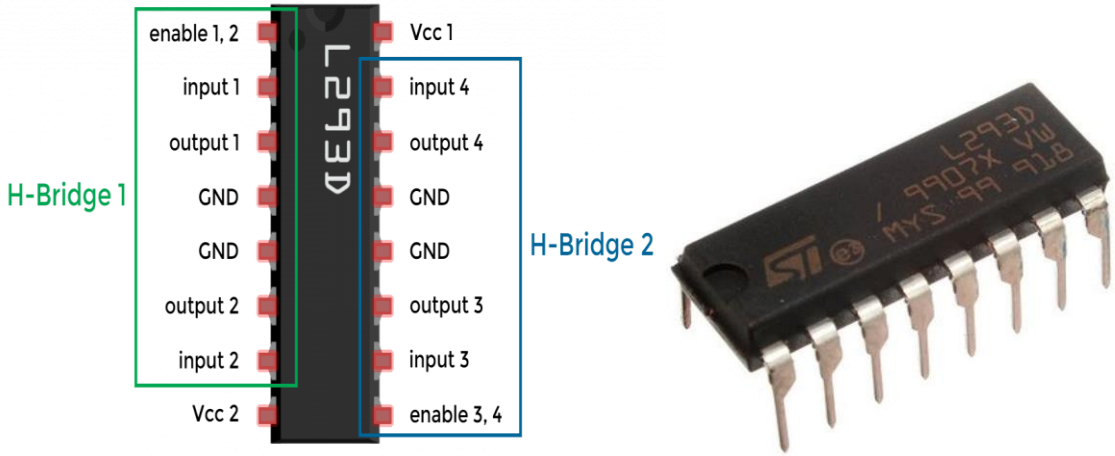


Fig. 3 pictures of L293D Chip

The L293D Motor Driver IC is a versatile integrated circuit commonly used to control the movement of DC motors in both directions. With 16 pins, this IC facilitates the control of a pair of DC motors simultaneously, allowing them to rotate in either direction. This makes it an excellent choice for various applications where precise control over motor movement is required. At its core, the L293D operates based on the H-bridge configuration, a fundamental motor control circuit. The H-bridge allows the flow of voltage to be manipulated in different directions, enabling the DC motor to rotate clockwise or counterclockwise. This bidirectional

control is crucial for achieving full motor functionality, as it lets us change the voltage direction to determine the motor's rotation.

The L293D IC contains two distinct H-bridge circuits within a single package, making it capable of independently controlling two separate DC motors. This feature makes it highly suitable for robotics and other projects involving multiple motors, as you can effectively control the movement of each motor independently. This is particularly valuable in scenarios where precise coordination or differential motor control is necessary. The primary application of the L293D IC is in robotics and automation, where its compact size and integrated design prove advantageous for managing DC motors efficiently. By sending appropriate signals to the input pins of the L293D, you can regulate the direction and speed of the connected motors. Additionally, the L293D is compatible with both small and larger motors, broadening its range of use cases.

H-Bridge Circuit

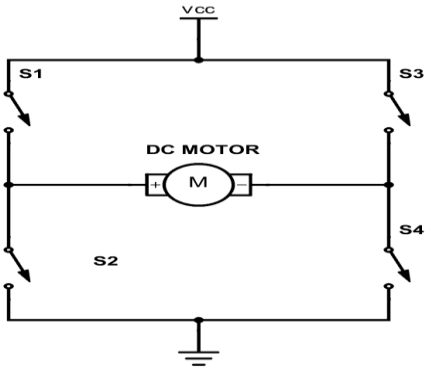


Fig.4 H-Bridge Circuit

An H-bridge circuit is an essential electronic configuration used to manipulate the direction of voltage applied across a load, such as a DC motor. This circuit is extensively employed in various fields, including robotics and power electronics, to enable the controlled forward and backward movement of DC motors and other devices. In the context of robotics and motor control, H-bridge circuits are of paramount importance. They serve as the foundation for allowing DC motors to operate in both forward and reverse directions, thus enabling the precise control of motor-driven mechanisms. Beyond robotics, H-bridges find application in a wide array of power electronic converters, such as DC-DC, DC-AC, and AC-AC converters.

In particular, H-bridges are especially relevant when working with bipolar stepper motors. These types of motors require a motor controller equipped with two H-bridges to control their operation effectively. An H-bridge consists of four switches denoted as S1, S2, S3, and S4. When switches S1 and S4 are closed, a positive voltage is applied across the load, typically a DC motor. This configuration results in the motor turning in one direction. By opening switches S1 and S4 and simultaneously closing switches S2 and S3, the voltage polarity is reversed, effectively causing the motor to rotate in the opposite direction. This mechanism allows for precise control over the motor's movement, making it possible to move it forward and backward as desired.

The flexibility of the H-bridge design is evident in its capacity to not only control motor direction but also to manage motor braking. For instance, in scenarios where the motor needs to come to an abrupt halt, the terminals of the motor can be shorted using appropriate switch configurations. Similarly, the motor can be allowed to coast to a stop by detaching it from the circuit entirely

ULTRASONIC SENSOR

Ultrasonic sensors are electronic devices that detect the distance of a target by emitting ultrasonic sound waves and then translating those waves into electrical signals. The speed of ultrasonic waves traveling is faster than the speed of audible sound. The concept behind ultrasonic sensors is that they generate waves with a higher frequency than humans can hear. After waiting for the wave to be reflected, the sensor determines how far away the object is when it is in front of it.

Fig. 5 Ultrasonic Sensor

How It Works

Ultrasonic sensors measure distance or detect the presence of objects by emitting and receiving ultrasonic sound waves. A transducer in the ultrasonic sensor creates high-frequency sound waves, generally in the ultrasonic range, which is above the range of human hearing. The most common frequencies utilized are in the 20 kHz to 65 kHz range, however this might vary depending on the sensor. It sends out a brief burst of ultrasonic sound waves in one direction. At room temperature, these sound waves move through the air at a measured speed of 343 meters per second (m/s). When the produced sound waves collide with an item, part of the waves are reflected back towards the sensor. This occurs when sound waves bounce off things in the same way as light reflects off a mirror. The identical transducer that generated the sound waves is now used as a receiver. It detects echoes or reflected waves that have returned after bouncing off an object.

The sensor measures the time it takes for the emitted sound waves to travel to the object and return as echoes. This time interval is often referred to as "time of flight."

Distance Calculation

Using the known speed of sound in the air, which is approximately 343 m/s at room temperature, the sensor calculates the distance to the object using the formula:

$$\text{Distance} = \frac{\text{Speed of Sound} \times \text{Time of Flight}}{2}$$

The division by 2 is necessary because the sound wave has to travel to the object and then back to the sensor.

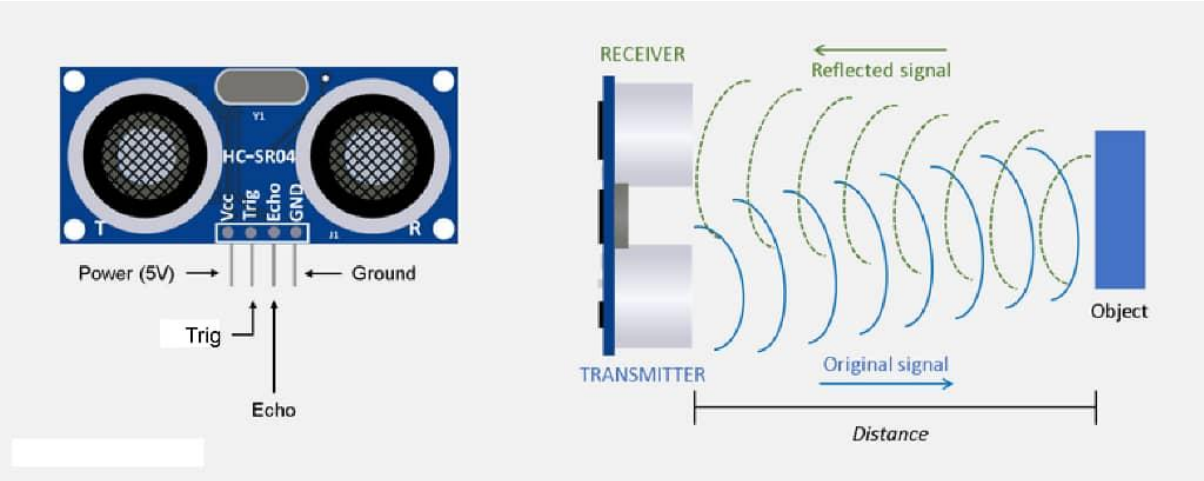


Fig. 6 parts of an ultrasonic sensor

HC-05 BLUETOOTH MODULE

The HC-05 Bluetooth module is a Bluetooth SPP (Serial Port Protocol) module commonly used electronic component that enables wireless communication between devices over short distances using Bluetooth technology. It's widely utilized in various projects and applications, especially in the field of electronics, robotics, and IoT (Internet of Things). The HC-05 module acts as a bridge, allowing data to be transmitted wirelessly between devices (in this case our robot and laptop) that are equipped with Bluetooth capabilities.

Bluetooth is a wireless communication technology that uses short-range radio waves to exchange data between devices. It operates on the 2.4 GHz frequency band and is widely used for creating wireless connections between devices like smartphones, laptops, printers, and more.

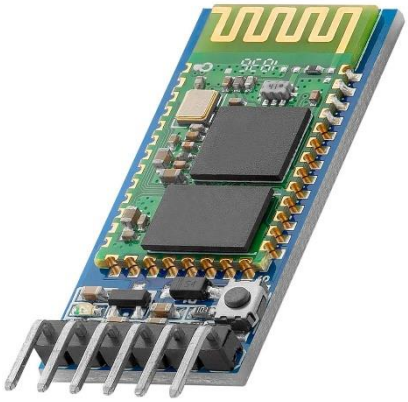


Fig. 7 Bluetooth Module

The Bluetooth module is fully certified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation and has a serial interface with a comprehensive 2.4GHz radio transceiver and baseband. It employs a CSR Blue core 04-External single chip Bluetooth system with AFH (Adaptive Frequency Hopping Feature).

The HC-05 module can operate in two primary modes:

- **Master Mode:** In this mode, the module acts as a master device and can connect to multiple slave devices simultaneously. This mode is often used in applications where one device (master) communicates with several other devices (slaves).
- **Slave Mode:** In slave mode, the module can connect to a single master device. This configuration is useful when you want to establish a connection between one central device (master) and another peripheral device (slave).

To establish a connection between devices, the HC-05 module needs to be paired with the target device. During pairing, a secure Bluetooth link is established between the devices, allowing them to exchange data. Once paired, the module can send and receive data using the serial communication interface (UART), making it relatively easy to integrate into microcontroller-based projects.

The HC-05 module supports a set of AT commands that allow you to configure various settings, such as changing the device name, setting the communication parameters (baud rate), enabling or disabling certain features, and more. These commands are sent over the serial interface, enabling you to customize the module's behavior.

The HC-05 Bluetooth module is versatile and can be used in a wide range of applications, including: Wireless data transmission between microcontrollers or embedded systems, remote control of electronic devices using a smartphone or other Bluetooth-enabled devices, creating wireless sensor networks, Bluetooth-enabled home automation systems and Bluetooth-based robotics and automation projects.

IR SENSOR

IR Sensor also known as **infrared proximity sensor** is an electronic device that emits infrared light to detect objects in the environment is a type of electrical equipment. The location of a line follower in relation to the robot position is determined using infrared ray sensors. The most common type of sensor utilized in the creation of a line follower robot is an infrared sensor (IR) for line sensing operations.

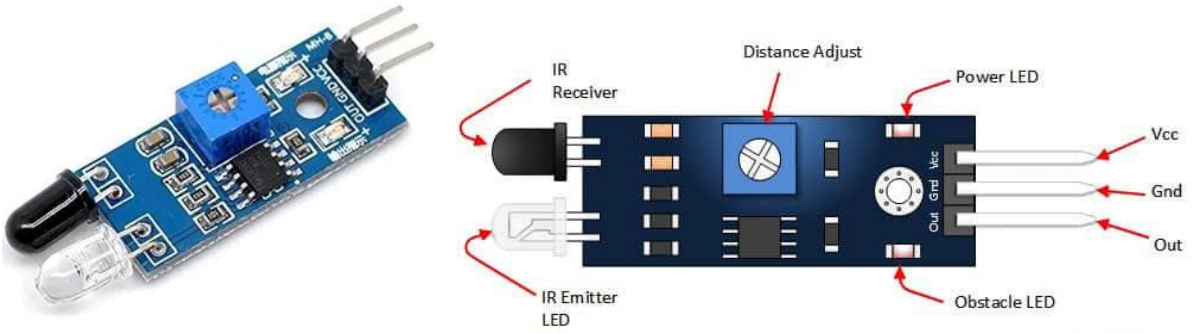


Fig. 8 Labeled parts of IR Sensor

Infrared (IR) light is a type of electromagnetic radiation that falls within the wavelength range between visible light and microwaves. It is not visible to the human eye because its wavelength

is longer than that of visible light. Despite being invisible, IR light is widely used in various applications, ranging from communication and remote sensing to heating and imaging.

Many IR sensors consist of both an emitter and a receiver. The emitter typically contains an infrared LED (*Light Emitting Diode*) that emits infrared light when powered and an infrared light receiver (*photodiode or phototransistor*). These components are positioned close to each other, usually facing in the same direction but separated by a gap. Some IR sensors are passive and do not emit IR radiation themselves. Instead, they rely on detecting the IR radiation naturally emitted or reflected by objects in their field of view. This is often the case with IR temperature sensors or passive infrared (PIR) motion sensors.

Reflection or Absorption in IR sensors

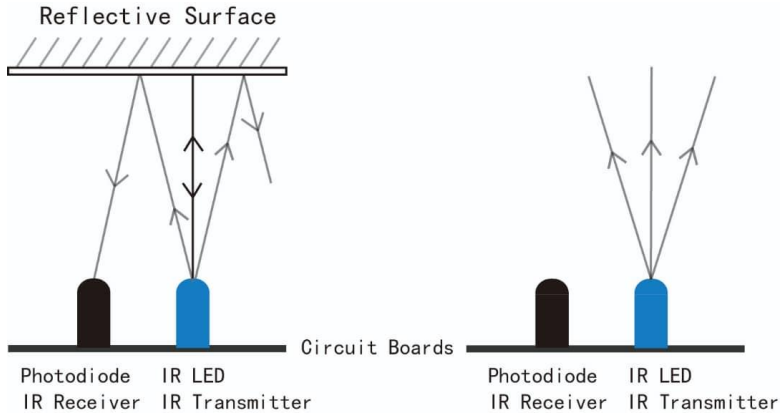


Fig. 9 image for reflection and absorption in IR sensor

In some applications, IR sensors emit infrared light, and the receiver measures the amount of light that is reflected off an object. The presence or absence of an object can be determined based on the reflected light. Other IR sensors, such as temperature sensors, detect the amount of IR radiation absorbed by an object. The amount of absorption can be related to the objects

These sensors are commonly used in a wide range of applications, such as robotics, automation, security systems, and consumer electronics.

How It Works

Reflective Mode: In this mode, the IR emitter and receiver are positioned side by side, facing the same direction. The emitted light bounces off nearby objects and is detected by the receiver. The presence of an object is detected when the intensity of the reflected light changes from the baseline level.

Transmissive Mode: Here, the emitter and receiver are positioned facing each other but separated by a gap. The emitted light travels across the gap and is received by the receiver. When an object enters the gap, it interrupts the light path, causing a decrease in the received light intensity.

Signal Processing: The intensity of the received light depends on the distance between the sensor and the object. Generally, closer objects reflect more light, resulting in a higher intensity at the receiver. The sensor measures this intensity and converts it into an electrical signal.

Threshold Comparison: IR proximity sensors often include signal conditioning and threshold comparison circuitry. The output signal is compared against a predefined threshold. If the measured signal exceeds this threshold, the sensor interprets it as the presence of an object.

IR proximity sensors are utilized in various applications, including:

- **Object Detection:** Used to detect the presence or absence of objects on conveyor belts, assembly lines, or automated machinery.
- **Obstacle Avoidance:** Commonly used in robots and drones to detect obstacles in their path.
- **Proximity Detection:** Used in touchscreen devices to turn off the display when the user's ear is close to the screen.
- **Security Systems:** Used in motion detection systems and burglar alarms.

SERVO MOTORS

A servo motor is a small type of rotary actuator that is widely used for precise control of angular position. It's commonly employed in a variety of applications ranging from robotics, automation, remote control systems, to camera gimbals and more. Servo motors offer accurate and controlled movement, making them an essential component in many projects. Servos are frequently used for specific velocity, acceleration, and angular or linear position.

Fig. 10 Servo motors

Components Of A Servo Motor

- **DC Motor:** At the core of a servo motor is a DC motor that provides the mechanical rotation. The DC motor usually has low inertia, allowing for quick and accurate response to control signals.
- **Feedback Device (Potentiometer or Encoder):** This is a sensor that provides feedback about the current position of the motor shaft. It allows the control system to determine the motor's real-time position and make necessary adjustments.
- **Control Circuitry:** The control circuitry interprets the input control signal and compares it with the feedback signal from the feedback device. Based on this comparison, it generates the appropriate control signals for the motor.
- **Gear box:** The DC motor's output shaft is connected to a gear box, which increases the torque output and reduces the speed. This combination of gears provides the motor with higher torque and finer control over the rotation.

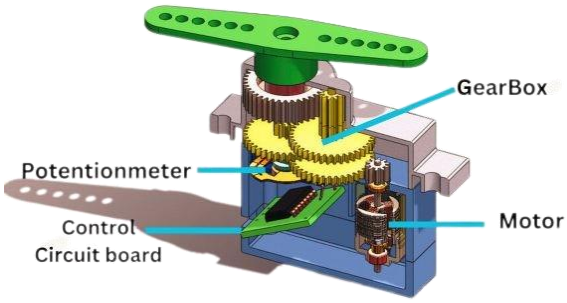


Fig. 11 Labeled parts of a servo motor

The servo motor works based on a closed-loop control system. Here's how it operates:

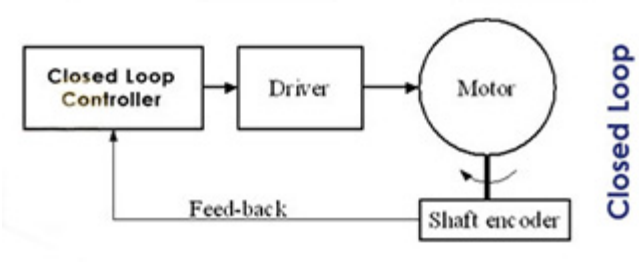


Fig. 12 Image of how servo motor works

- **Input Control Signal:** A servo motor is typically controlled using a control signal, often in the form of PWM (Pulse Width Modulation). The control signal represents the desired angle or position that the motor shaft should move to.
- **Comparing Control Signal and Feedback:** The control circuitry compares the control signal with the feedback signal from the feedback device. The feedback signal provides information about the current position of the motor shaft.
- **Error Calculation:** The control circuit calculates the error, which is the difference between the desired position (control signal) and the actual position (feedback signal). This error signal drives the control loop.
- **Generating Control Output:** Based on the calculated error, the control circuit generates control output to the DC motor. This output adjusts the voltage applied to the motor to drive it towards the desired position.
- **Motor Movement:** The DC motor receives the control output and begins to rotate the shaft. As the shaft moves, the feedback device continuously updates the feedback signal.
- **Closed-Loop Control:** The control circuit constantly adjusts the control output based on the feedback signal. As the error decreases, the motor's rotation slows down until it reaches the desired position.

PASSIVE BUZZER

An electronic component that makes sound when an electrical signal is supplied to it. A passive buzzer, as opposed to an active buzzer, which has its own built-in oscillator to generate a tone, relies on an external waveform generator, such as a microcontroller or a function generator, to make sound.

Fig. 13 Passive Buzzer

How To Use A Passive Buzzer On The Arduino

One advantage of passive buzzers over active buzzers is that you can control the tone or pitch of the sound produced by the buzzer. With active buzzers only one tone is possible, but with passive buzzers any tone within the dynamic range of the buzzer is possible.

Passive buzzers need a square wave signal to produce sound. By changing the frequency of the square wave you can change the pitch of the sound. The Arduino has a built in function called tone() that generates square waves at a range of frequencies:

tone(pin, frequency, duration);

The tone() function takes three parameters – the pin number you want to send the square wave to, the frequency of the tone in hertz, and optionally the duration of the tone in milliseconds.

The noTone() function can be used to turn off the tone() function:

noTone();

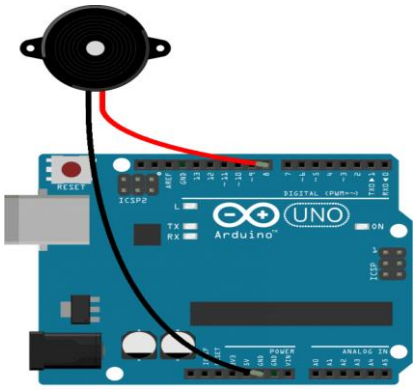


Fig. 14 a passive Buzzer connected to an Arduino

POWER SUPPLY

A power supply is a device or system that supplies electrical energy to other devices or circuits. It transfers electrical energy from a source, such as an outlet or a battery, into a useable form, often voltage and current, for powering electronic devices and equipment.

Power Supplies used;

- **Lithium batteries** are rechargeable batteries that employ lithium as a primary component in their electrochemical processes. A lithium battery is made up of a lithium anode (positive electrode), a cathode (negative electrode), and an electrolyte that allows lithium ions to flow between the anode and cathode. Various types of lithium batteries use various materials and chemistries.

Fig.15 Lithium battery

- **Power bank,** also known as a portable charger or external battery pack, is a portable device designed to store electrical energy and provide it to other devices, such as smartphones, tablets, laptops, or any other electronic gadgets, for charging or powering when no traditional electrical outlet is available. Power banks have grown in popularity due to their ease, particularly for consumers on the go.

Fig.16 power bank

BATTERY MANAGEMENT SYSTEM

BMS is an abbreviation for "**Battery Management System"** is technology dedicated to the oversight of a battery pack, which is an assembly of battery cells, electrically organized in a row x column matrix configuration to enable delivery of targeted range of voltage and current for a duration of time against expected load scenarios. It is a critical component used to monitor and control the operation of batteries especially Lithium-ion rechargeable cells in this case, ensuring their optimal performance, safety, and longevity. The term "3s 2p" refers to a specific configuration of batteries within the BMS.

3s: The "3s" stands for "3 cells in series." In the context of battery management, it means that three individual battery cells are connected in a series configuration. When batteries are connected in series, their voltages add up while their capacity remains the same. This results in

an increased total voltage output, making it suitable for applications that require higher voltage levels.

2p: The "2p" stands for "2 cells in parallel." This configuration involves connecting two sets of three battery cells each in parallel. Parallel connection maintains the total voltage of the individual cells while increasing the overall capacity of the battery pack. This is useful for applications that require higher energy storage and longer runtime.

Therefore, "**3s 2p**" describes a battery configuration where three individual battery cells are connected in series to create a higher voltage, and two sets of these series-connected cells are connected in parallel to increase the overall capacity. This configuration strikes a balance between higher voltage output and increased energy storage capacity, making it suitable for various applications, such as electric vehicles, power tools, renewable energy systems, and more

Key function of the BMS

BMS monitors the voltage and temperature of each individual battery cell to ensure they are operating within safe limits.

- **Cell Balancing:** In a parallel configuration, battery cells may discharge and charge at slightly different rates, leading to imbalances. The BMS ensures that cells are balanced by redistributing charge between them.
- **Over/Under Voltage Protection:** The BMS prevents overcharging or over-discharging of cells by disconnecting them from the circuit if their voltage goes beyond safe limits.
- **Temperature Monitoring:** The BMS monitors cell temperature and triggers safety measures if cells get too hot, preventing thermal runaway.
- **State of Charge Estimation:** The BMS estimates the state of charge of the entire battery pack, providing accurate information about the remaining energy.

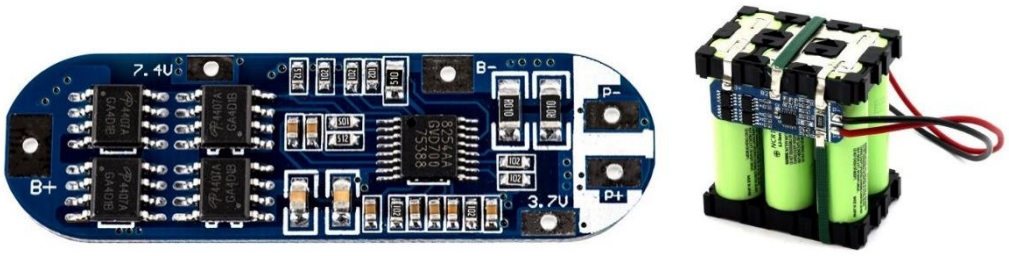


Fig. 17 3s 2p battery connection with a BMS

TT DC GEAR BOX MOTORS

DC gear box motors are a type of electric motor that combines a DC (direct current) motor with a gearbox, resulting in a motor unit capable of delivering both torque and controlled speed. These motors are widely used in various applications where precise control over speed, torque, and direction is essential.

Fig.18 Tt Dc Gear Box Motors

The basic components of a DC gear box motor include:

- **DC Motor:** The heart of the system, the DC motor converts electrical energy from the power source (typically a battery or power supply) into mechanical rotational energy. DC motors are known for their simple control and are available in various sizes and power ratings.
- **Gearbox:** The gearbox is a mechanical component that consists of a series of gears with varying sizes. By interconnecting these gears in specific configurations, the gearbox can increase or decrease the speed of the motor's output shaft while increasing the torque. This trade-off between speed and torque is achieved through the gear ratio, which is the ratio of the number of teeth on the input gear to the number of teeth on the output gear.

DC gear box motors are designed to provide different gear ratios, allowing them to serve a wide range of applications. High gear ratios result in higher torque output and lower speed, making them suitable for scenarios where the motor needs to move heavy loads or overcome resistance. Conversely, low gear ratios provide higher speed and lower torque, which is useful for applications requiring precise motion control.

Advantages

1. **Torque Amplification:** The gearbox helps amplify the torque produced by the motor, enabling the motor to handle higher loads.
2. **Speed Control:** By changing the gear ratio, you can control the motor's speed, making it suitable for applications requiring varying speeds.
3. **Precision:** DC gear box motors offer precise control over motion due to the ability to adjust the gear ratio.
4. **Efficiency:** In applications where high torque and low speed are required, using a gearbox can make the system more efficient by allowing the motor to operate at its optimal speed range.

Limitations

1. Reduced maximum speed compared to non-geared DC motors.
2. Gear wear and backlash can affect precision.

DC gear box motors find application in various industries, including robotics, industrial automation, automotive systems, conveyor systems, medical equipment, and more. They are versatile tools that provide the necessary combination of torque and speed control required for a wide range of mechanical tasks.

4 WHEEL ROBOT CAR CHASSIS

A 4-wheel robot car chassis is a mechanical framework or structure that serves as the foundation for building a mobile robot with four wheels. These chassis are commonly used in robotics projects, especially for hobbyists, students, and enthusiasts who want to create their own customized mobile robots.

Fig.19 4-wheel robot car chassis

JUMPER WIRES

Jumper wires are simple cables with connector pins at either end that may be used to connect two places without soldering. Jumper wires are commonly used with breadboards and other prototyping tools to allow for quick circuit changes as needed. It's quite straightforward. Indeed, it doesn't get simpler than jumper wires.

Fig.20 Jumper wires

Although jumper wires come in a variety of colours, they do not actually mean anything. The wire colour is just an aid to help you keep track of what is connected to which. It does not affect the operation of the circuit. This means that a red jumper wire is technically the same as the black one. The colours can be used to your advantage to differentiate the types of connections.

Types of Jumper Wires

Jumper wires come in three versions:

- Male-to-male jumper
- Male-to-female jumper
- Female-to-female jumper

And two types of head shapes: square head and round head.

The difference between each is in the *endpoint* of the wire. Male ends have a pin protruding and can plug into things, while female ends do not but are also used for plugging.

Moreover, a male connector is referred to as a plug and has a solid pin for centre conduction. Meanwhile, a female connector is referred to as a jack and has a centre conductor with a hole in it to accept the male pin. Male-to-male jumper wires are the most common and what you will likely use most often. For instance, when connecting two ports on a breadboard, a male-to-male wire is what you will need.

3.3.2 SOFTWARES NEEDED

- Arduino IDE
- Microsoft EXCEL

3.3.3 PROGRAMMING LANGUAGE

- C++

3.3.4 OTHER NECESSARY TOOLS AND MACHINES

- Soldering Iron
- Solder
- Glue gun
- Glue sticks
- Screw drivers
- Voltmeter
- Cardboard
- Black tape

3.4 HARDWARE SETUP

Steps:

1. Solder two jumper wires to each terminal of the TT DC gearbox motors.
2. Mount 4 TT DC gearbox motors onto the chassis. Position two motors on each side to ensure balanced movement.
3. Attach the L239D motor driver shield onto the Arduino Uno board.
4. Connect both positive and negative terminal of each motor to the L293D motor driver for precise speed and direction control.
    - Motor 1 to motor driver M1
    - Motor 2 to motor driver M2
    - Motor 3 to motor driver M3
    - Motor 4 to motor driver M4
5. Install the 2 IR sensors at the front of the robot to detect line deviations.
    - IR sensor OUT pin is connected to motor driver A0 and A5 pin for left and right respectively.

- IR sensor GND pin is connected to motor driver GND pin.
- IR sensor VCC pin is connected to motor driver 5v pin.

6. Attach the servo motor to motor driver servo1 slot on the robot to enable the ultrasonic sensor's rotational movement.
    - Position the ultrasonic sensor on the front for obstacle detection and avoidance and connect it to the motor driver shield.
    - Hc-sr04 TRIG pin to motor driver A3.
    - Hc-sr04 ECHO pin to motor driver A2.
    - Hc-sr04 5v pin to motor driver 5v.
    - Hc-sr04 GND pin to motor driver GND.
7. Connect the HC-05 Bluetooth module to the Arduino Uno for wireless communication. Ensure proper wiring and communication settings for data exchange
    - HC-05 RX pin to the TX slot of the L293d motor shield.
    - HC-05 TX pin to the RX slot of the L293d motor shield.
    - HC-05 5v pin to motor driver 5v.
    - HC-05 GND pin to motor driver GND.

    NB: Make sure you disconnect the RX and TX shield every time you are uploading a code or an error would be generated. After uploading the code, you can connect the RX and TX pins back
8. Connect the power source (battery) to the robot's components, ensuring proper voltage levels and current capacity. Since we are using a high voltage to control the DC motors, remove the power supply selection jumper from the L293D shield to avoid the high voltage from passing through the Arduino board which can damage the circuit board.
    - Connect the 5v power bank to the Arduino board itself.
    - And connect the 3s2p lithium battery to the L293D motor driver shield.
9. Connect the Switch to the battery source and the motor shield to control the circuit.
10. Connect the passive buzzer to produce a specific frequency of sound when a certain condition is met.
    - passive buzzer Signal pin to A1
    - Passive buzzer 5v pin to motor driver 5v.
    - passive buzzer pin to motor driver GND

3.5 PROGRAMMING THE ROBOT

The Arduino microcontroller communicates with the PC via the Bluetooth connection. Data is transferred between the board and the PC bit by bit. An adaptor is used for power supply to the board and a USB programmer is used to burn the hardware program (written in Arduino IDE) into the board.

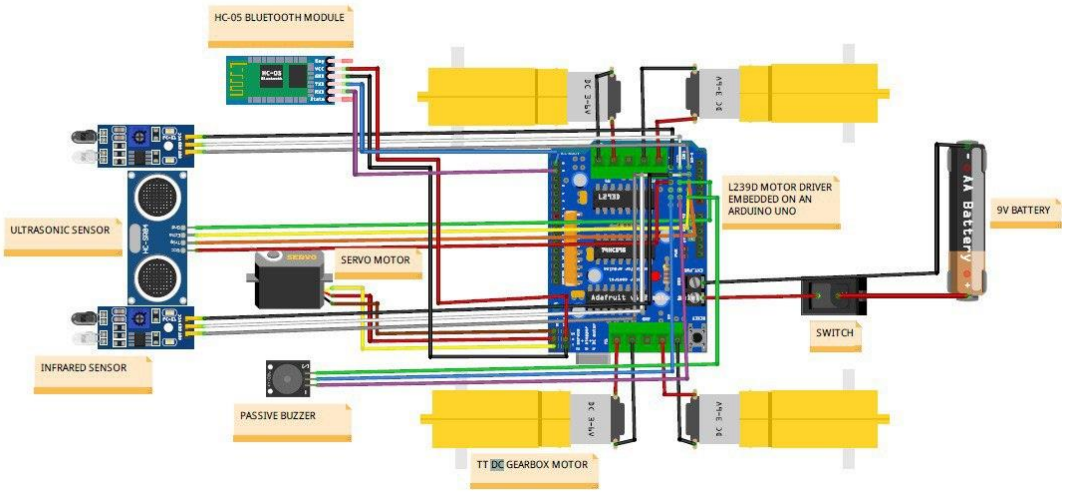


Fig.21 Architectural diagram

3.6 ROBOT MOVEMENT ON PATH

The Robot proceeds along its route locating the black line. When the distance between the object and the robot is closer than the threshold distance, which is set in the programmer, the obstacle-avoiding module is activated.

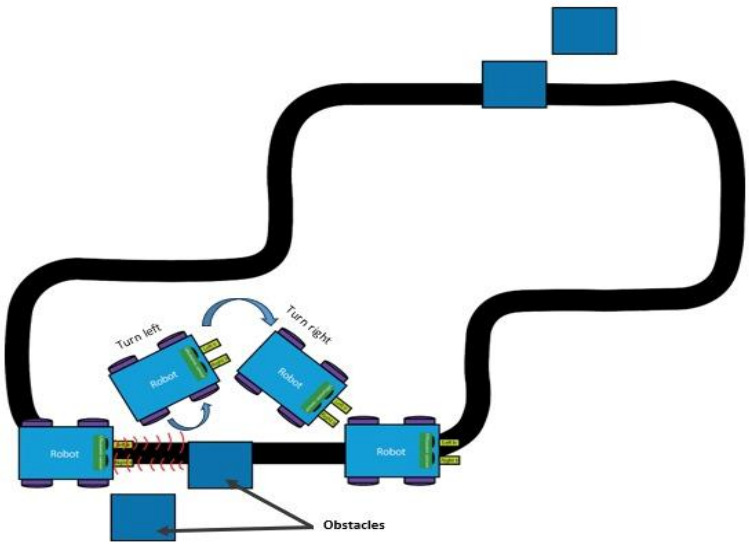


Fig.22 Movement of motor on a path

How the robot run on black line and avoid the obstacle in front of it.

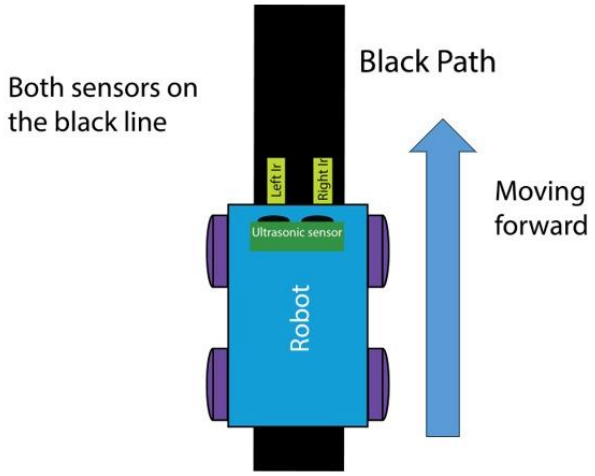


Fig.23 Straight movement of motor on a path

Once the robot detects an obstacle, it uses the information from the sensors to calculate a new path. The algorithms analyse the sensor data and determine the best course of action to avoid the obstacle. This could involve adjusting the robot's speed, changing its direction, or even stopping completely to prevent a collision.

When an obstruction is detected, the robot rotates left or right to locate a space. When the robot has to turn left, it slows or stops the motors on the right side while maintaining a constant speed on the left side.

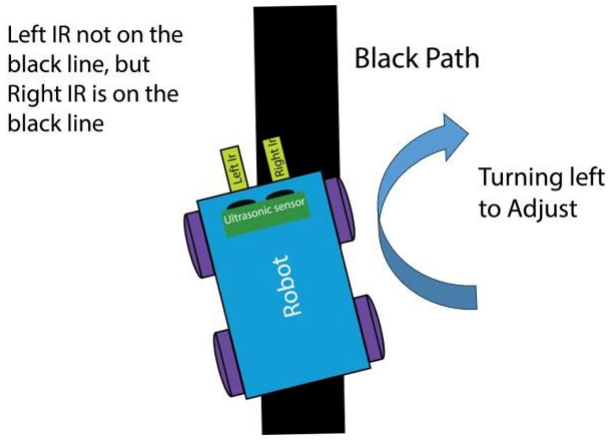


Fig.24 Left movement of motor on a path

Similarly, when the robot has to turn right, the speed of the motors on the left side is adjusted. The robot can make accurate rotations and maneuver around obstacles by adjusting the various motor speeds.

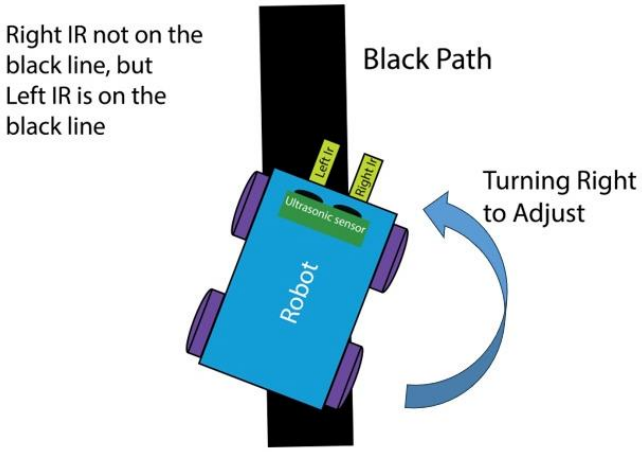
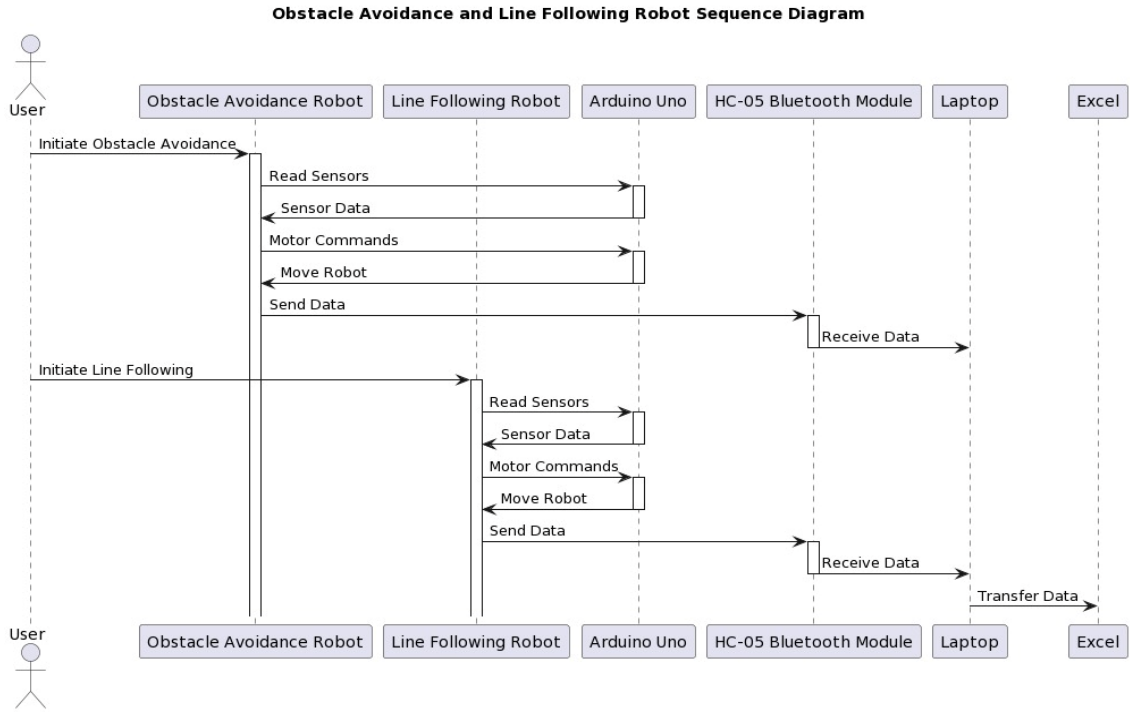


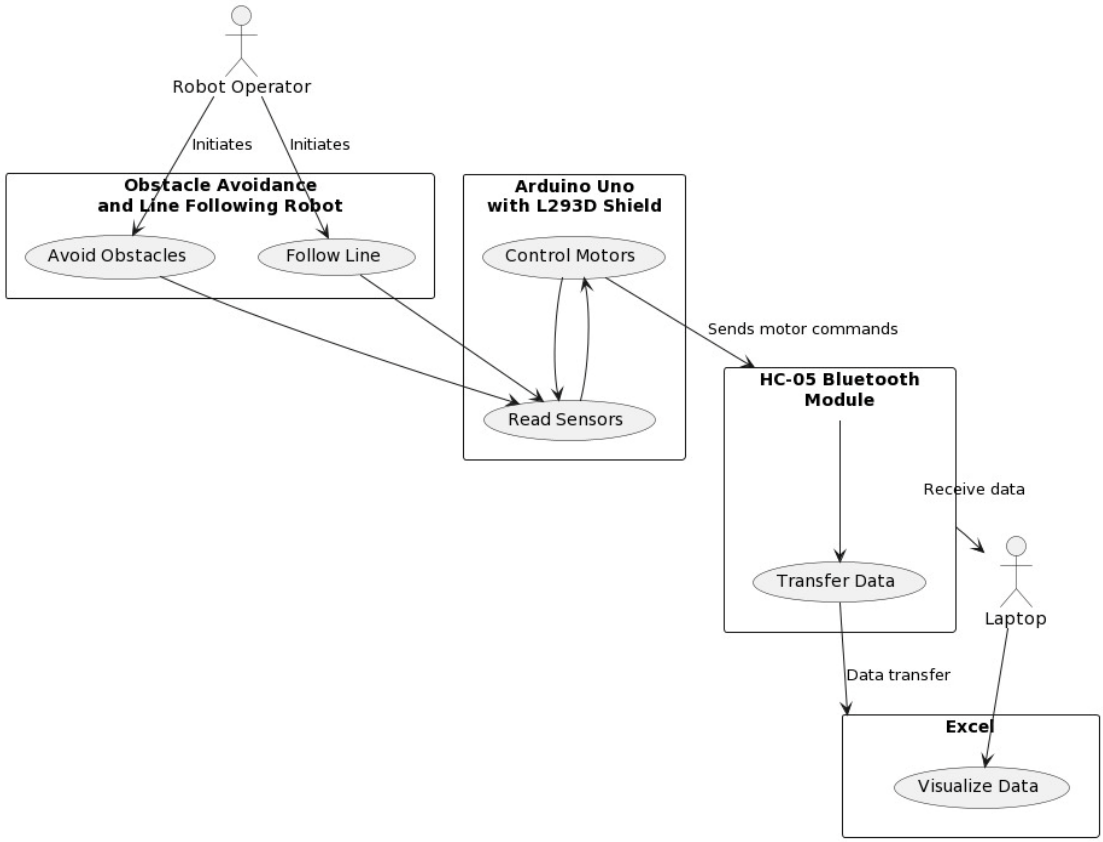
Fig.25 Right movement of motor on a path

3.7 UNIFIED MODELING LANGUAGE (UML) DIAGRAMS

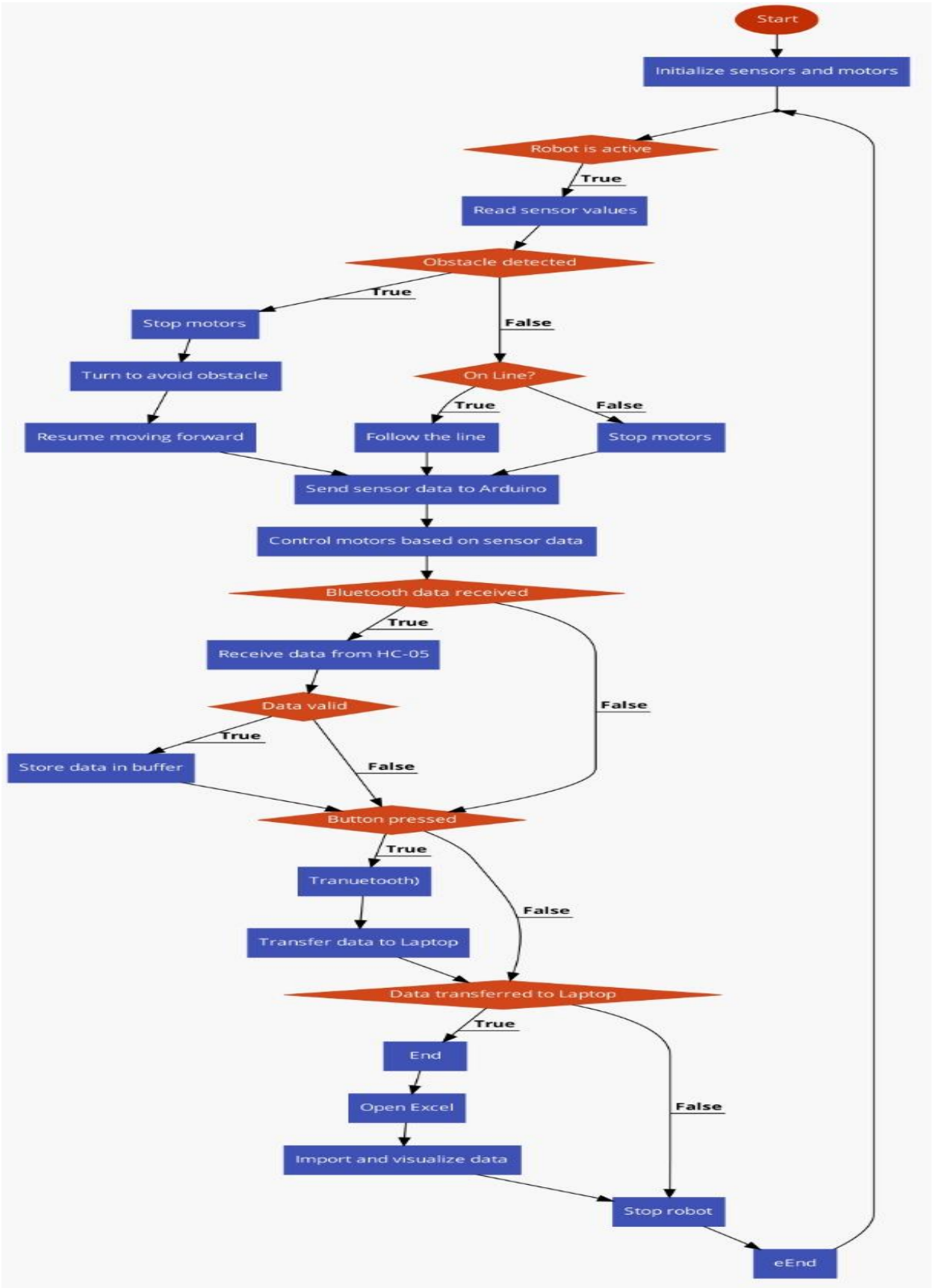
3.7.1 Sequence diagram of the program



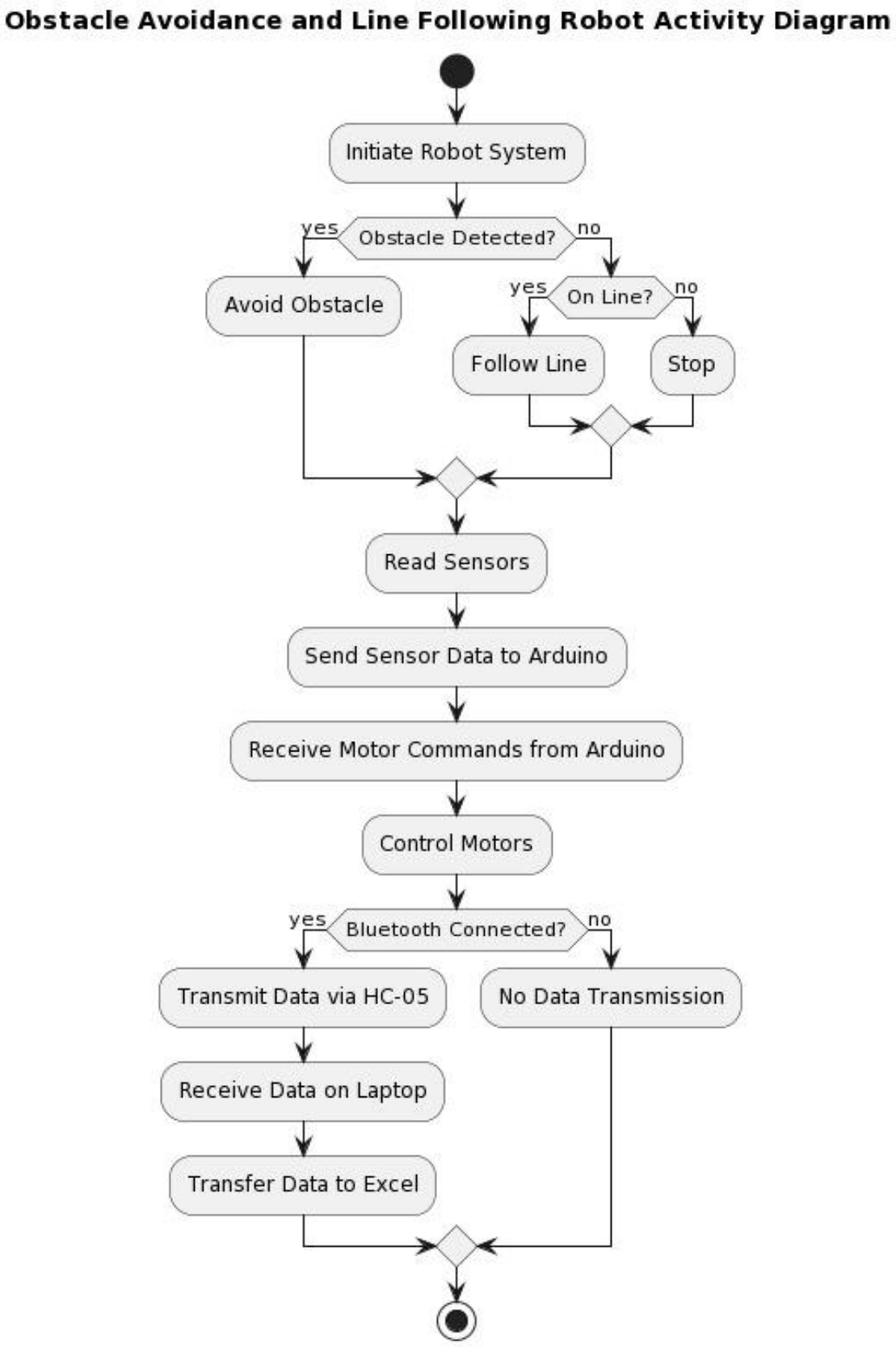
3.7.2 Use case Diagram of the program



3.7.3 Flowchart of the program



3.7.4. Activity diagram of the program



3.7.5 DBML diagram of the program

3.8 SOFTWARE IMPLEMENTATION

3.8.1 ARDUINO PROGRAMMING

Develop the Arduino code to read data from IR sensors for line following. Implement logic to control the DC motors using the L293D motor driver for smooth movement. Integrate the ultrasonic sensor to detect obstacles and implement avoidance maneuvers. Enable servo motor control for rotating the ultrasonic sensor when an object is detected. Also set up Bluetooth communication to send sensor data to the laptop for data visualization in excel.

- **Obstacle Avoidance Algorithm:** Design an algorithm that triggers obstacle avoidance when the ultrasonic sensor detects an obstacle. Implement a sequence of movements to navigate around the detected obstacle.
- **Line Following Algorithm:** Develop a line following algorithm using data from IR sensors. Define motor control logic to keep the robot centered on the line.
- **Bluetooth Data Transmission**: Configure the HC-05 Bluetooth module to establish a connection with a laptop. Implement data transmission routines to send sensor readings to the laptop.
- **Data Visualization:** Set up excel to receive data via excel Data Streamer from COM4 O COM5 and interpret the robot's sensor readings and movements. Import and analyze

the transmitted data to understand the robot's behavior during operation. Visualize the data received in a form of line chart.

SYSTEM INTEGRATION

- Combine the hardware and software components to create a functional obstacle avoidance and line following robot.
- Calibrate sensors and motors to ensure accurate readings and smooth movements.
- Conduct thorough testing to identify and resolve any issues.

PERFORMANCE EVALUATION

- Test the robot's obstacle avoidance capabilities in various environments with different obstacles.
- Evaluate the line following accuracy on various types of tracks and line widths.
- Assess the Bluetooth communication's reliability and range.

3.8.2 PROJECT METHOD: AGILE APPROACH

The Agile project strategy was chosen for the development of an Obstacle Avoidance and Line Follower Robot using Arduino. Agile technique complements the project's complexity and iterative nature, allowing for gradual development, ongoing feedback, and adaptability to changing needs.

Agile Methodology Justification:

Given the dynamic nature of an Obstacle Avoidance and Line Follower Robot using Arduino, the agile approach offers a pragmatic solution. The incremental development process permits early testing and validation of system components, leading to quicker identification and resolution of any issues. By embracing the Agile approach, we effectively manage evolving requirements, foster collaboration among team members, and deliver a more responsive and relevant Obstacle Avoidance and Line Follower Robot using Arduino

3.8.3 FUNCTIONAL REQUIREMENTS

1. **Line Following**: The robot should be able to follow a black line on a white surface using infrared sensors.
2. **Obstacle Detection:** The robot should be able to detect obstacles in its path using ultrasonic sensors.
3. **Obstacle Avoidance:** The robot should be able to avoid obstacles by changing its direction or stopping until the obstacle is removed.
4. **Speed Control:** The robot should be able to adjust its speed based on the distance between it and the obstacle.
5. **Manual Control:** The robot should have manual control options for testing and debugging purposes.
6. **Battery Management:** The robot should be able to operate for at least 2 hours on a single battery charge.
7. **Robustness:** The robot should be designed to withstand minor collisions without causing damage or malfunctioning.
8. **Customizability**: The robot should be customizable to add new sensors or functionality, depending on the user's requirements.

3.8.4 NON-FUNCTIONAL REQUIREMENTS

1. **Performance:** The robot should perform accurately and consistently under different lighting conditions, surface types, and obstacle shapes/sizes.
2. **Reliability**: The robot should operate reliably without frequent breakdowns or malfunctions.
3. **Maintainability:** The design of the robot should allow for easy maintenance, repair, and replacement of components if necessary.
4. **Usability:** The user interface of the robot should be intuitive and easy to use for users with varying levels of technical expertise.
5. **Safety:** The design of the robot should prioritize safety by avoiding sharp edges or hazardous materials that could harm users or damage property.

3.9 CODE

Complete codes will be found at the public GitHub repository link below:

https://github.com/Snortey/Obstacle-avoidance-and-path-following-robot

```
#include <TimerFreeTone.h>// Library for the Buzzer
#include <AFMotor.h> // Library for the motors
#include <Servo.h>     // Library for the servo motor
#include <NewPing.h> // Library for the ultrasonic sensor
#include <IRremote.h> //Library for the IR reciever


//Ultrasonic sensor
#define TriggerPin A3
#define EchoPin A2
#define max_distance 50 //50 at first
//Infrared Sensor
#define IrLeft A0 //Left line sensor
#define IrRight A5 //Right line sensor
//Pezzo Buzzer
#define buzzer A1
//motor
#define MaxSpeed 200
#define MaxSpeed_Offset 20
//speed of the motoes
#define SPEED1 58 //54
#define SPEED2 97
#define SPEED3 60

//creates an instance of the Servo class for controlling the servo motor.
Servo servo;

const int Motor_1 = 1;
```

```
const int Motor_2 = 2;
const int Motor_3 = 3;
const int Motor_4 = 4;

//creates an instance of the NewPing class for reading from the HC-SR04 ultrasonic sensor
NewPing sonar(TriggerPin, EchoPin, max_distance);

//creates instances of the AF_DCMotor class for controlling the four motors of the robot.
//Pulse Width Modulation and is a technique used to control the speed of DC motors. By varying the pulse width of the signal sent to the motor, the effective voltage supplied to the motor is varied, which in turn varies the speed of the motor.
//AF_DCMotor(motor#, frequency)
AF_DCMotor motor1(Motor_1, MOTOR12_1KHZ);// create motor object, 1KHz pwm....1,2 4,8,16,32,64
AF_DCMotor motor2(Motor_2, MOTOR12_1KHZ);
AF_DCMotor motor3(Motor_3, MOTOR34_1KHZ);
AF_DCMotor motor4(Motor_4, MOTOR34_1KHZ);

//declares variables for storing the distances detected by the sensors.
int distance =0;
int LeftDistance = 0;
int RightDistance = 0;
int IRLEFT=0;
int IRRIGHT =0;
int set = 6;
String STATE = " ";
char val;
//boolean variable for determining the direction of the turn
boolean Object;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);//initializes the serial communication
  Serial.println("Start");
```

```
  servo.attach(10);//attaches the Servo object to pin 10
  servo.write(95); //sets the initial position of the servo to 90 degrees

  pinMode(IrLeft, INPUT); // sets the pins for the infrared sensors as inputs
  pinMode(IrRight, INPUT);

  pinMode(buzzer, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  //If both sensors detect a line, it calls the objectAvoid() function and moves forward

  // if(Serial.available() > 0){
  //   val = Serial.read();
  //   Stop();
  //   if (val == 'F'){
  //     moveForward();
  //   }
  //   if(val == 'B'){
  //     moveBackward();
  //   }
  //   if(val == 'L'){
  //     moveLeft();
  //   }
  //   if(val == 'R'){
  //     moveRight();
  //   }
  //   if(val == 'T'){
  //     Stop();
  //   }
  // }

  printValues();
```

```
  // Reset the printed values to zero
  distance = 0;
  LeftDistance = 0;
  RightDistance = 0;


  IRLEFT = digitalRead(IrLeft);
  IRRIGHT= digitalRead(IrRight);

  if (!digitalRead(IrLeft) == 0 && !digitalRead(IrRight) == 0 ) {
    objectAvoid();
  }
  //If only the right sensor detects a line, it calls the objectAvoid() function, movesright , and prints "TR" to the serial monitor.
  else if (digitalRead(IrLeft) == 0 && !digitalRead(IrRight) == 0 ) {
    objectAvoid();
    STATE = "Turn Right";
    moveBackward();
    delay(1);
    moveRight();
  }
  //If only the left sensor detects a line, it calls the objectAvoid() function, moves left, and prints "TL" to the serial monitor.
  else if (!digitalRead(IrLeft) == 0 && digitalRead(IrRight) == 0 ) {
    objectAvoid();
    STATE = "Turn Left";
    moveBackward();
    delay(1);
    moveLeft();
  }
  //If neithber sensors detect line, the motors stops.
  else if (digitalRead(IrLeft) == 0 && digitalRead(IrRight) == 0 ) {
    Stop();
    TimerFreeTone(buzzer,10,100);
```

```
    STATE= "Stop";
  }
}
//function for detecting obstacles and avoiding them
void objectAvoid() {
  distance = getDistance();
  //calls the getDistance() function and assigns its return value to the variable distance
  //checks whether the distance variable is less than or equal to 15 centimeters, which is the threshold distance for detecting obstacles
  if (distance <= set) {
    Stop();
    STATE = "Stop";
    moveBackward();
    delay(300);
    Stop();
    // delay(100);
    TimerFreeTone(buzzer,700,500);
    TimerFreeTone(buzzer,261,500);

    //These lines call the lookLeft() and lookRight() functions to measure the distance to the left and right of the robot using a servo-mounted ultrasonic sensor.
    LeftDistance = lookLeft();
    RightDistance = lookRight();

    // if right distance is less than or equal to the left distance, the robot will turn left to avoid the obstacle.
    if (RightDistance <= LeftDistance) {
      Object = true;//Set the object variable to true to indicate that an obstacle has been detected on the right
      TimerFreeTone(buzzer,392,1000);
      turn(); // call the turn() function to turn left
      STATE = "Move Left";
    //If the right distance is greater than the left distance, the robot will turn right to avoid the obstacle
```

```
    } else {
      Object = false;
      TimerFreeTone(buzzer,440,1000);
      turn();
      STATE = "Move Right";
    }
    delay(100);
  }
    //If no obstacle is detected within the threshold distance, the robot will move forward.
  else {
    STATE = "Move Forward";
    moveForward();//moveForward() function to move the robot forward.
  }
}
// function to turn the robot when an obstacle is detected
void turn(){
  if (Object == false){//obstacle is detected on the left side so you move right
    moveRight();
    delay(620);//700
    moveForward();
    delay(2000);//800
    moveLeft();
    delay(700);//720
    moveForward();
    delay (400);
     //check if the IR sensor on the left side detects the line.If the line is detected, it calls the loop() function. If not, it calls the moveForward() function again
    if (digitalRead(IrLeft) == 1) {
      loop();
    } else {
      moveForward();
    }
  }
  else if (Object == true) {
```

```
    moveLeft();
    delay(620);//700
    moveForward();
    delay(2000);//800
    moveRight();
    delay(700);//720
    moveForward();
    delay (400);
    if (digitalRead(IrRight) == 1) {
      loop();
    } else {
      moveForward();
    }
  }
}


// function responsible for measuring the distance to an obstacle using a servo-mounted ultrasonic sensor.
int getDistance(){
  delay(50);
  int cm = sonar.ping_cm();//use the ping_cm() function of the NewPing library to measure the distance to an obstacle in centimeters and store it.
  //If the measured distance is zero or -ve, the distance is set to 100 centimeters.
  if (cm == 0) {
    cm = 100;
  }
  return cm;//distance is returned
}

//detect object at the left side
int lookLeft(){
  servo.write(180);// set the servo position to 170 degrees which corresponds to the leftmost position.
```

```
  delay(500);//wait 500 milliseconds to allow time for the ultrasonic sensor to measure the distance.
  int dist = getDistance();
  delay(200); //waits for 100 milliseconds.
  servo.write(95);//sets the servo motor back to the center position (90 degrees)
  return dist;//returns the distance value.
  delay(100);
}

//detect object at the Right side
int lookRight(){
  servo.write(10);// set the servo position to 10 degrees which corresponds to the right position.
  delay(500);//wait 300 milliseconds to allow time for the ultrasonic sensor to measure the distance.
  int dist = getDistance();
  delay(200);
  servo.write(95);
  return dist;
  delay(100);
}
//MOTOR SPEED
void set_Motorspeed(char g) {
  if (g == 'L' || g == 'R' ) {
    motor1.setSpeed(SPEED2);//130
    motor2.setSpeed(SPEED2);
    motor3.setSpeed(SPEED2);
    motor4.setSpeed(SPEED2);
  }else if(g == 'B'){
    motor1.setSpeed(SPEED3);//100
    motor2.setSpeed(SPEED3);
    motor3.setSpeed(SPEED3);
    motor4.setSpeed(SPEED3);
  }
  else {
```

```
    motor1.setSpeed(SPEED1);//110
    motor2.setSpeed(SPEED1);
    motor3.setSpeed(SPEED1);
    motor4.setSpeed(SPEED1);
  }
}

//This function sets all the motors to run in the forward direction by calling the FORWARD parameter of the run() method of the AF_DCMotor library.
void moveForward(){
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
  set_Motorspeed('F');
}
//This function sets all the motors to run in the backward direction by calling the BACKWARD parameter of the run() method of the AF_DCMotor library
void moveBackward(){
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
  set_Motorspeed('B');
}
//function is used to make the robot turn right. It makes the left motors turn backward and the right motors turn forward by calling the BACKWARD and FORWARD parameters of the run() method of the AF_DCMotor.
void moveRight(){
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
  set_Motorspeed('R');
```

```
}
```
//function is used to make the robot turn left. It makes the left motors turn forward and the right motors turn backward by calling the FORWARD and BACKWARD parameters of the run() method of the AF_DCMotor.
```
void moveLeft(){
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
  set_Motorspeed('L');
}
```
//This function stops all the motors by calling the RELEASE parameter of the run() method of the AF_DCMotor library.
```
void Stop(){
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}

void printValues(){
    Serial.print(STATE);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(",");
    Serial.print(IRLEFT);
    Serial.print(",");
    Serial.print(IRRIGHT);
    Serial.print(",");
    Serial.print(LeftDistance);
    Serial.print(",");
    Serial.println(RightDistance);
    // delay(2000);
  }
```

CHAPTER 4: RESULTS AND RECOMMENDATIONS

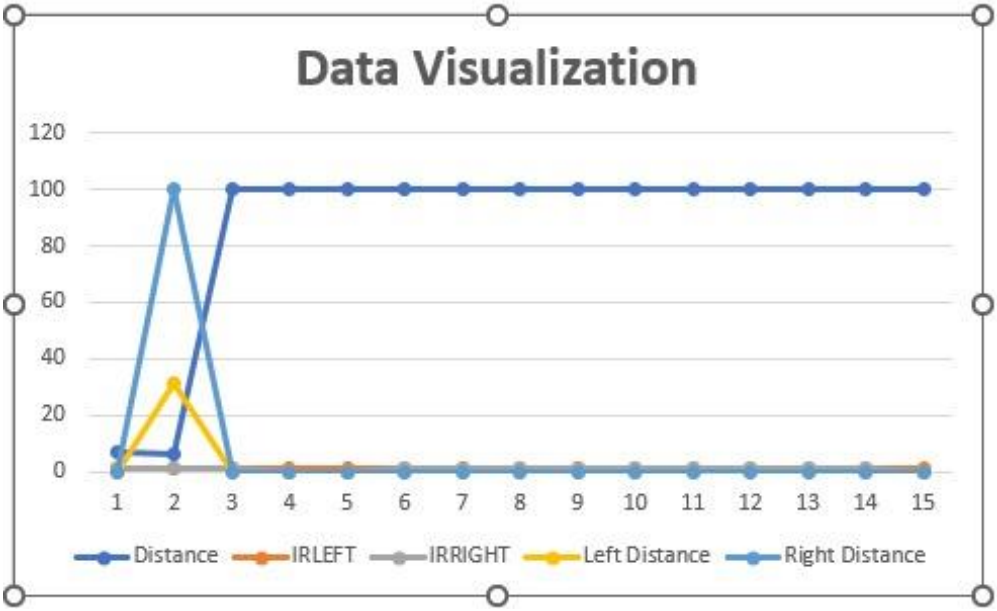
4.1 RESULTS

The end result of this project is a simple, Arduino-controlled robot car that moves while avoiding obstacles by sensing them. The ultrasonic sensor emits an ultrasound wave to the front position, right position, and left position while the robot is in motion. The wave bounces back after hitting an obstruction, and the distance for the front, right, and left positions is recorded. The microcontroller then compares the values using its algorithm to decide whether to go or change course.

Fig.26 Robot model

Table no. 1: Show reading of robot

	A	B	C	D	E	F	G
6	Data From Robot						
7	TIME	STATE	Distance	IRLEFT	IRRIGHT	Left Distance	Right Distance
8	15:21:54.76	Move Forward	7	1	1	0	0
9	15:22:02.40	Stop	6	1	1	31	100
10	15:22:02.49	Move Right	100	1	1	0	0
11	15:22:02.55	Turn Left	100	1	0	0	0
12	15:22:02.59	Turn Left	100	1	TBL_HST[CH3]	0	0
13	15:22:02.69	Move Forward	100	1	1	0	0
14	15:22:02.73	Move Forward	100	1	1	0	0
15	15:22:02.78	Move Forward	100	1	1	0	0
16	15:22:02.84	Move Forward	100	1	1	0	0
17	15:22:02.90	Move Forward	100	1	1	0	0
18	15:22:02.93	Move Forward	100	1	1	0	0
19	15:22:03.02	Move Forward	100	1	1	0	0
20	15:22:03.06	Move Forward	100	1	1	0	0



4.2 RECOMMENDATION

A completely autonomous robot should be able to adapt to its surroundings (surface, underwater, underground, and space), as this is critical in robotics. It should also be able to function for long periods of time without being interrupted, avoid dangerous circumstances, and move all or part of itself across its operational area. Better sensors are necessary for this robot to get greater outcomes and precision. Furthermore, improved actuators will result in a speedier and more efficient robot. Other suggestions include:

1. **Addition of a Camera:** If a camera is added into this project, the robot can be driven beyond line of sight. Also, the camera could be used for image processing purposes.
2. **Addition of wireless technology:** If wireless technology is added to the current project, this could allow for external communication and control of the robot from a remote computer.
3. Depending on the application, the robot can be given a task to identify a place on a map, go there, and perform a certain function

CHAPTER 5: CONCLUSION

5.1 CONCLUSION

We live in a robotics-based world today and employ many robot types on a regular basis. This "Obstacle Avoidance Robot Car" project is demonstrated utilizing an ultrasonic sensor for object detection, a motor driver shield for DC motor driving, and DC motors for the robot's wheel movement with the aid of an Arduino microcontroller. The environment in which the robot was tested and the quantity of impediments in the testing area are two aspects that have an impact on the designed robot's accuracy. The sensor was primarily impacted by these elements, hence the sensor itself determines how accurate the robot will be.

REFERENCES

[1] Crisp, J. (2004). Introduction to Microprocessors and Microcontrollers. Oxford: Newnes.

[2] Hall, D. (2005). Microprocessors and Interfacing. New York: Tata McGraw Hill Education Private Limited.

[3] M. Saravanan, N. S. (2010). Microprocessors and Microcontrollers. Oxford: Oxford University Press.

[4] Monk, S. (2017). Electronics Cookbook. California: O'Reilly Media, Inc.

[5] Murphy, R. R. (2000). Introduction to AI Robotics. Massachusetts: MIT Press.

[6] Paul, O. (2019). Robotics. Retrieved from https://www.sciencedirect.com/topics/engineering/robotics.

[7] What is robotics? (2019). Retrieved from https://builtin.com/robotic

[9] Artificial Intelligence. (2018). Retrieved from https://builtin.com/artificial-intelligence.

[8] Banzi, M. (2008). Getting started with Arduino. California: O'Reilly Media, Inc.

[9] HC-SR04 Ultrasonic sensor. (2017). Retrieved from https://components101.com/ultrasonic sensor-working-pinout-datasheet

[10] Ultrasonic Sensor HC-SR04 and Arduino Tutorial. (2015). Retrieved from https://howtomechatronics.com /tutorials/Arduino/ultrasonic-sensor-hc-sr04/

[11] Jagruti Chaudhari et. al, Department of Electronics & Telecommunication MGM's Jawaharlal Nehru Engineering College Aurangabad (MS)India 2019.

[12] Sandeep Polina1, Pavan Kumar Barathula2and K P Prasad Rao3 Department of Electrical and Electronics Engineering K L University, Vaddeswaram - 522 502, Andhra Pradesh, India. Lawanya Shri M," A line following robot for Hospital management", Research gate publications, February 2019.

[13] Obstacle Avoiding Robot By Faiza Tabassum, Susmita Lopa, Muhammad Masud Tarek & Dr. Bilkis Jamal Ferdosi, Global Journal of Researches in Engineering.

[14] https://www.tutorialspoint.com/arduino/arduino_servo_motor.htm#:~:text=A%20Servo%20Motor%20is%20a,angular%20position%20of%20the%20shaft

[15] https://circuitdigest.com/microcontroller-projects/interfacing-ir-sensor-module-with-arduino

[16] https://www.electronicwings.com/sensors-modules/bluetooth-module-hc-05-

[17] https://www.synopsys.com/glossary/what-is-a-battery-management-system.html

[18] https://en.wikipedia.org/wiki/Battery_management_system

[19] https://www.adafruit.com/product/3777

[20] https://www.elprocus.com/h-bridge-motor-control-circuit-using-l293d-ic/