ACKNOWLEGEMENT

We would like to convey our heartfelt gratitude to Anil Thapa for his tremendous support which kept us motivated to continue our project each time we faced any difficulty. We would also like to thank our HOD, Nischal Regmi, Principal, Birod Rijal for providing us the hardware components used in our project. Special thanks to Supriya Pandey who shared us reference to project related to Arduino, Bishnu Hari KC sir for providing other useful components for circuit testing.

ABSTRACT

Shortest-path bot is a bot that can follow track on the basis of the rule that defines a path. It is made in order to solve the shortest path problem. Even though many similar robots are made but they are mostly limited to just follow a line or path. In this research, we examine different such scenarios and address them by also including shortest path algorithm. We gathered the similar topics from different papers and used their findings as a reference to our work. Moreover, we build a bot that can follow shortest path from a single source to different destination having different costs. We found that there are many constraints that defines the cost of the path and ways to uniquely identify a node/vertex. This research proposes a skeleton that addresses the communication between micro-controller and various sensors. Moreover, this work explains the sensing the physical parameters that a robot takes as its input and acts accordingly.

Keywords: Single Source Shortest Path, Sensors, Micro-Controller.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

IR	-	Sensor- Infrared Sensor
I/O	-	Input Output
USB	-	Universal Serial Bus
IDE	-	Integrated Development Environment
DC	-	Direct Current Motors
RPM	-	Rotation Per Minute
LED	-	Light Emitting Diode
PCB	-	Printed Circuit Board
AI	-	Artificial Intelligence
PWM	-	Pulse Width Modulation
SRAM	-	Static Random-Access Memory
KB	-	Kilo Bytes
EEPROM	-	Electronic Erasable Programmable Read Only Memory
MHz	-	Mega Hertz
PC	-	Personal Computer
IC	-	Integrated Circuit
IDE	-	Integrated Development Environment
PID	-	Proportional-Integral-Derivative Controller

Chapter 1: INTRODUCTION

1.1Background:

In robotics and automation, there is increasing demand of intelligent systems that is capable of determining and navigating the environment autonomously. Shortest-Path Bot represent advancement in this system, offering the potential to transform industries by providing efficient and precise navigation along predefined paths.

Despite significant development in robotics, the development of Shortest-Path Bot remains a complex challenge. The combination of sensors, control systems and adaptable algorithms requires a balance to ensure the bot's effectiveness in different scenarios. The successful completion of this project suggests potential practical uses ranging from warehouse automation to smart homes and precision agriculture. Although several path-following or line following bots are already made but bot following the complex joints such as sharp (90^{o}) turns, T- joints, etc. still remains a quite hard situation to overcome by a bot. In this project, the bot will be designed and built so that it can easily tackle the complex turns like sharp turn and T-joints.

1.2 Statement of Problem:

The current problem we want to solve by our project is to use sensor technologies to navigate through predefined environment to find shortest distance between two points (Source and Destination) and display it through physical representation of Shortest-Path Bot.

Problems associated with Project that can be solved with this Bot**:** -

1.2.1. Navigation

Vehicles and Robots can move autonomously without colliding with obstacles.

1.2.2. Safety

Increases safety in environments by avoiding collision and navigating around obstacles.

1.2.3. Search and Rescue

Helps in Search and Rescue by navigating through challenging terrains.

1.2.4. Monitoring

Assist in monitoring tasks navigating predefined paths systematically.

1.2.5. Exploration

Allows robots to explore on their own in places like jungles, oceans or even outer space.

1.2.6. Human Support

Supports robots designed to assist humans, like robotic vacuum cleaners in avoiding obstacles.

1.3 Objective:

The primary objective of a Shortest-Path Bot is to navigate autonomously in a predefined environment to find Shortest Distance between two points by detecting and following a designated path.

1.4 Scope:

Some Scopes of Shortest-Path Bot are: -

1.4.1. Automation and Robotics

Bot can be employed for tasks like warehouse navigation, material handling and automation in industries.

1.4.2. Home automation

Bot can be used in smart home devices, such as robotic vacuum cleaners, to navigate and clean floors efficiently.

1.4.3. Education

Bot can be used as educational tools, helping students understand robotics, programming and sensor integration.

1.4.4. Security and Surveillance

Bot can be equipped with camera for patrolling and monitoring areas, enhancing security.

1.4.5. Agriculture

In precision agriculture, bot can be used to navigate fields for tasks like planting, monitoring crops, or applying fertilizers.

1.4.6. Healthcare

Bot can be used in tasks like transporting medical supplies within a facility.

1.4.7. Research and Exploration

Bot can navigate and explore environments that may be challenging or hazardous for humans, such as disaster-stricken areas or other planets.

1.5 Applications:

Bot find diverse applications across various industries, contributing to automation, efficiency, and safety.

Some notable applications include: -

1.5.1. Warehouse Automation

Bot navigate through warehouses, optimizing the movement of goods, and automating tasks such as inventory management and order fulfillment.

1.5.2. Smart Cleaning Devices

Robotic vacuum cleaners use path detection to navigate and clean floors systematically, ensuring thorough coverage.

1.5.3. Agricultural Robotics

Bots equipped with path-detection capabilities are employed in precision agriculture for tasks like planting, monitoring crops, and harvesting.

1.5.4. Manufacturing

Bots navigate assembly lines, delivering components and ensuring the smooth flow of materials in manufacturing processes.

1.5.5. Security and Surveillance

Shortest-Path Bot enhance security by patrolling predefined areas, monitoring for unusual activities, and providing real-time surveillance.

1.5.6. Healthcare Assistance

Bots can transport medical supplies within healthcare facilities, reducing the need for human involvement.

1.5.7. Educational Robotics

Bots can serve as educational tools, helping students learn about robotics, programming and sensor integration.

1.5.8. Research and Exploration

Bots explore environments that may be hazardous or challenging for humans, aiding in research and exploration tasks.

1.5.9. Delivery Services

Autonomous delivery bots use path detection to navigate sidewalks and streets, delivering packages to specified locations.

The versatility of Shortest-Path Bots makes them valuable across a wide range of industries, enhancing efficiency, reducing labor requirements and opening up new possibilities for automation and innovation.

1.6 Hardware and Software Requirements:

Hardware Components required for Shortest-Path Bot are:

1.6.1. Arduino Mega (ATmega2560)

The Arduino Mega is a microcontroller board based on the ATmega2560 chip. It has 54 digital I/O pins and 16 analog Input pins. It has 256 KB of flash memory for storing program, 8 KB of SRAM for variables and 4 KB of EEPROM for non-volatile storage.

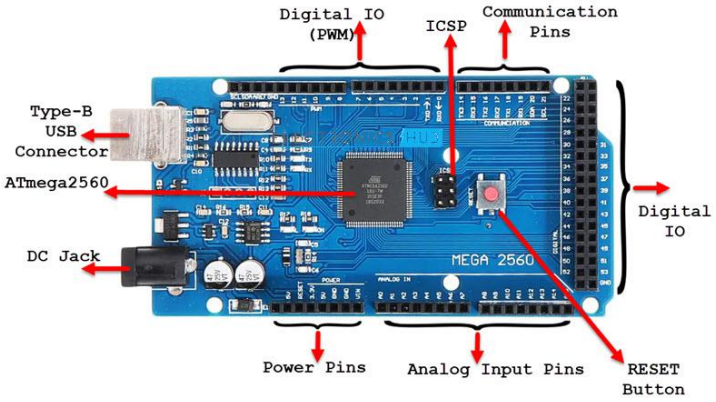
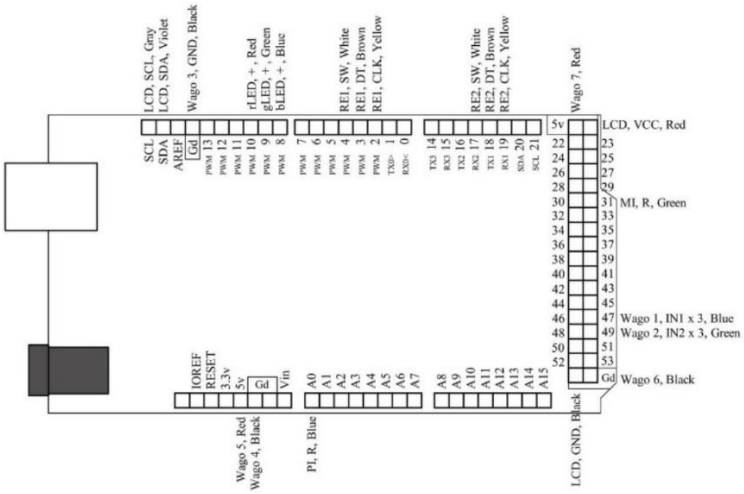


Fig-1.6.1: Arduino Mega

We use Arduino Mega due to its I/O pins, Powerful microprocessor, larger memory capacity compared to Arduino Uno making it suitable for Further implementation in Future Project.

1.6.1.1 Arduino Mega Pin Diagram



IOREF: This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

RESET (RST Pin): The RST pin of this board can be used for rearranging the board. The board can be rearranged by setting this pin to low.

3.3V, 5V: These pins are used for providing o/p regulated voltage approximately 5V. This RPS (regulated power supply) provides the power to the microcontroller as well as other components which are used over the Arduino mega board.

GND: The Arduino mega board includes 5-GND pins which are used to ground the circuit.

Vin: The range of supplied input voltage to the board ranges from 7volts to 20volts. The voltage provided by the power jack can be accessed through this pin. However, the output voltage through this pin to the board will be automatically set up to 5V.

A0~A15 (Analog Pins): There are 16-analog pins included on the board which is marked as A0-A15. These analog pins are used to read analog sensors. All the analog pins on this board can be utilized like digital I/O pins. Every analog pin is accessible with the 10-bit resolution which can gauge from GND to 5 volts.

SCL and SDA: The I2C (Inter-Integrated Circuit) protocol involves using two lines to send and receive data: a Serial Clock Pin (SCL) that the Arduino Controller board pulses at a regular interval, and a Serial Data Pin (SDA) over which data is sent between the two devices.

AREF: The term AREF stands for Analog REFerence. It allows us to feed the Arduino a reference voltage from an external power supply.

PWM (Pin-2~13): PWM (Pulse Width Modulation) is a technique that alters a continuous digital signal into a series of pulses. PWM Pin is commonly used as a way to generate an analog signal that correlates to a digital value. Common use of PWM pins includes controlling LEDs and DC Motors.

TXD and RXD (Serial Pins): The serial pins are used for communication between the Arduino board and a computer or other devices. The TXD and RXD are used to transmit & receive the serial data respectively. Tx indicates the transmission of information whereas the RX indicates receive data. The serial pins of this board have four combinations. For serial 0, it includes Tx (1) and Rx (0), for serial 1, it includes Tx(18) & Rx(19), for serial 2 it includes Tx(16) & Rx(17), and finally for serial 3, it includes Tx(14) & Rx(15).

Digital I/O Pin-(0~53): Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50 k ohm.

1.6.2. Motor Driver L298N

L298N is a Dual H-bridge motor driver IC with two independent channels, enabling bidirectional control of DC motors. Operating with a motor voltage of 4.8V to 46V and a current handling capacity of 2A per channel. The IC uses digital inputs for direction and PWM for speed control. Built-in diodes protect against back EMF.

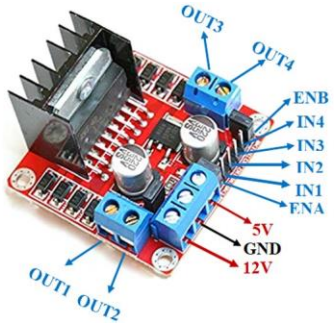
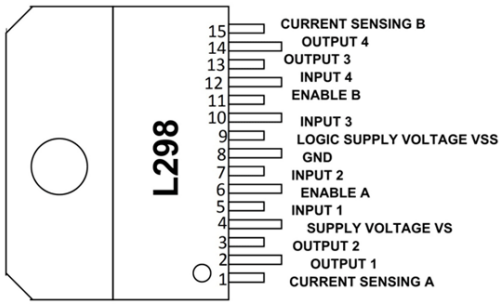


Fig-1.6.2: Motor Driver L298N

We use L298N Motor Driver because it has built-in diodes to protect from EMF which is suitable for BO motors. And this Motor Driver uses digital inputs for direction and PWM signals for speed control which is very helpful in our bot's movement.

1.6.2.1 Pin-Diagram of Motor Driver L298N:



Current Sensing A, B: Between this pin and the ground, a sense resistor is connected to control the current of the load.
Enable A, B: TTL Compatible Enable Input: The L state disables the bridge A (enable A) and/or the bridge B (enable B).
Supply Voltage VS: Supply Voltage for the Logic Blocks. (A100nF capacitor must be connected between this pin and the ground.)
Input 1,2,3,4: TTL Compatible Inputs of the Bridge A and B.
Output 1,2,3,4: Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and the ground.
Logic Supply Voltage VSS: Supply Voltage for the Logic Blocks. (A100nF capacitor must be connected between this pin and ground.)
GND: Ground

1.6.3. DC Motors

A motor with built -in gears for controlled movement in small devices. It is used to spin the wheels of the device. The DC motors are an advanced variation of the brush DC motors. They have a gear assembly attached to the motor. The speed of the motor is measured in Rotation per Minute (RPM). The speed of the
motor is reduced with an increase in torque with the help of motor driver. It's used to adjust the whole bot movement. In this project we have used two dc motor for driving two bot wheels.

Fig-1.6.3: DC Bo Motors

We use DC Motors to spin the wheels of the Bot.

Weight:	**20 g**
RPM:	**60 RPM, 100 RPM, 300 RPM**
Operating Voltage (VDC):	**3 ~ 12V**
Shaft Length (mm):	**8.5**
No Load Current (mA):	**40-180mA**

1.6.4. Wheels / Caster Wheel

Wheels are connected to the chassis and DC motor to move Bot from one place to another.

Fig-1.6.4: Wheels / Caster Wheel

Wheels are important for movement of the bot.

1.6.5. Infrared Sensors

Infrared sensors detect infrared radiation emitted or reflected by objects. Commonly used for proximity sensing, object detection and temperature measurement. They work by converting infrared radiation into an electrical signal. Widely applied in security systems, consumer electricity and industrial automation.

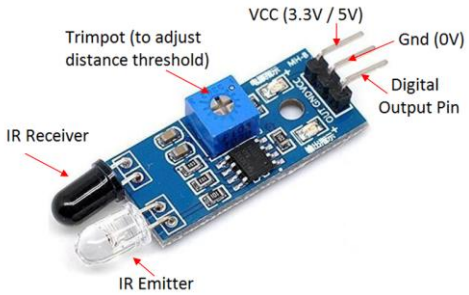
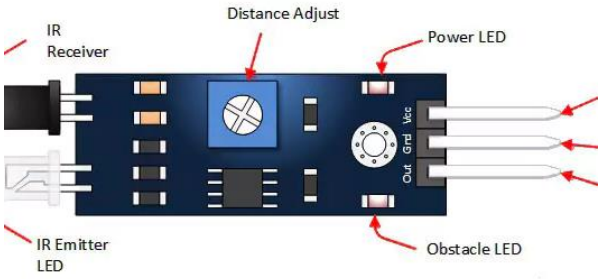


Fig-1.6.5: IR Sensor

We use IR Sensor to detect the black line on the floor which sends signal to the Arduino which stops bots from going out of the line. It is one of the most important components in our project.

1.6.5.1 Pin-Diagram of Infrared Sensor



VCC: 3.3 to 5 Vdc Supply Input
GND: Ground Input
Digital Output Pin: The output that goes low when an obstacle is in range.
Power LED: Illuminates when power is applied
Obstacle LED: Illuminates when an obstacle is detected
IR Emitter: Infrared emitter LED
IR Receiver: The infrared receiver that receives signal transmitted by Infrared emitter.

1.6.6. Ultra-Sonic Sensor (HC- SR04)

Ultrasonic sensors use sound waves to measure distance or detect objects. They work by emitting ultrasonic pulses and measuring the time it takes for the waves to travel to an object and back. They are non-contact, suitable for short to medium-range applications, but can be affected by environmental conditions.

Fig-1.6.6: US Sensor

We use US sensor to detect any objects in the path and we also use US sensor to detect source and destination.

1.6.6.1 Pin-Diagram of Ultra-Sonic Sensor

VCC: The Vcc pin powers the sensor, typically with +5V
Trigger Pin: Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
Echo Pin: Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
GND: This pin is connected to the Ground of the system.

1.6.7. Servo Motor (SG90)

A servo motor is like a robot muscle that can move to specific position and hold it there, thanks to built-in control and feedback systems. It's widely used in things like robots and remote-controlled devices for precise movements.

Fig-1.6.7: Servo Motor

We use Servo Motor to control the Ultrasonic Sensor left and right to measure and detect objects.

1.6.7.1. Pin-Diagram of Servo Motor:

VCC (Red): Powers the motor typically +5V is used
GND (Brown): Ground wire connected to the ground of system
Signal (Orange): PWM signal is given in through this wire to drive the motor.

1.6.8. Bluetooth Module (HC-05)

A Bluetooth module is a wireless communication device that enables the exchange of data between electronic devices like mobile phones laptops over short distances. It uses Bluetooth technology to establish a connection allowing devices like microcontroller and laptop to communicate with each other. It is used for wireless data transfer between devices and remote-control.

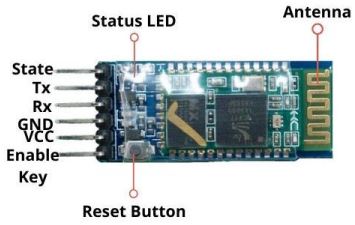
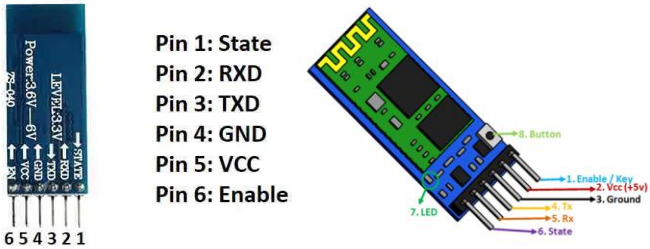


Fig-1.6.8: Bluetooth Module

It is very important component for our project it lets us to transfer input and instruction to the Arduino wirelessly.

1.6.8.1. Pin-Diagram of Bluetooth Module:



State: The state pin is connected to on board LED, it can be used as feedback to check if Bluetooth is working properly.

RXD: Receive Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth.

TXD: Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.

GND: Ground pin of module, connect to system ground.

VCC: Powers the module. Connect to +5V Supply voltage

Enable: This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default, it is in Data mode.

LED: Indicates the status of Module

- Blink once in 2 sec: Module has entered Command Mode
- Repeated Blinking: Waiting for connection in Data Mode
- Blink twice in 1 sec: Connection successful in Data Mode

Button: Used to control the Key/Enable pin to toggle between Data and command Mode.

1.6.9. DC-DC Buck Boost Converter (CA-6009)

The buck–boost converter is a type of DC-to-DC converter that has an output voltage magnitude that is either greater than or less than the input voltage magnitude. This converter is placed between the battery source and the motor driver. It regulates the voltage to ensure that the motor driver receives a constant and appropriate voltage, regardless of fluctuations in the battery voltage.

Fig-1.6.9: DC-DC Buck Boost Converter

We use this converter to control the voltage fluctuation from the current source. And to provide motor driver with stable voltage to operate efficiently.

1.6.10. Power Supply (Rechargeable Battery) [Lithium-Ion]

A high-energy, reusable power source found in devices like smartphones and laptops.

Fig-1.6.10: Rechargeable Battery with Battery Charger

We use this battery because it is compact in size and very efficient. It can also be recharged so we can reuse it as many times which is suitable for testing phase of bot.

1.6.11. Chassis

A chassis is a framework or structure that provide support for various components of a machine or electronic devices. It's the skeleton that holds wheels, Arduino, battery, sensors and other parts together. It provides physical framework for the device.

Fig-1.6.11: Chassis

We use chassis to have structure in our bot to hold all the components.

1.6.12. Breadboard

Breadboard is the workbench for electronics. It is a flat board with lots of tiny holes where you can stick jumper wires and electric components. It lets you build and test circuits without soldering, making it easy to experiment and prototype different electronic projects.

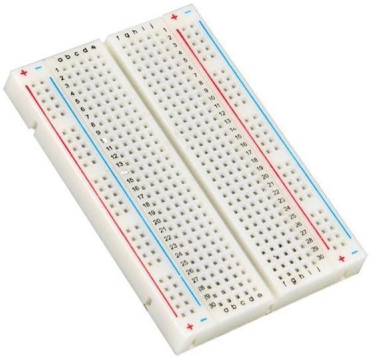


Fig-1.6.12: Breadboard

We use breadboard to test the circuit for the bot before soldering it.

1.6.13. Wires and Connectors (Jumper Wire)

A Jumper wire is a short electrical wire with connectors at each end, used to create connections on a breadboard or between electronic components. It's a simple and flexible tool for temporary linking different points in a circuit during prototyping or testing phases, making it easier to experiment without soldering.

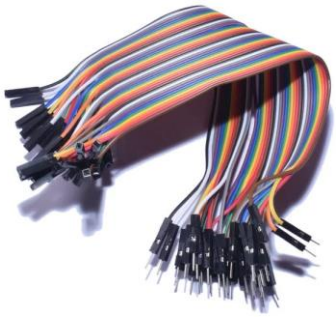


Fig-1.6.13: Jumper Wire

We use Jumper wire to make the connection to the motor driver, sensors, Arduino and all the components.

1.6.14. PC/Laptop

Minimum Requirement:

Processor	**:** Pentium 4 or above
OS	**:** 32-bit and 64-bit Operating System
RAM	**:** 256 MB excluding RAM required for OS
Memory	**:** 20 GB or higher
Window	**:** Windows 7 or later

Software Components required for Path-Detecting Bot: -

1.6.15. C++

C++ is a general-purpose programming language designed as an extension of the C programming language. It supports object-oriented programming features and provides low-level memory manipulation, making it versatile for various applications, from system programming to developing software applications and games.

We use C++ programming language to program our Bots movements and Sensors Operation to make our bot follow the path and detect the shortest path between source and destination.

1.6.16. Arduino IDE

The Arduino IDE (Integrated Development Environment) is a software application that provides a platform for writing, compiling and uploading code to Arduino microcontrollers. It offers a user-friendly interface for programming Arduino boards and is equipped with features like code highlighting, serial monitoring and library manager, making easier for developers to create and upload code to their Arduino Projects**.**

We use Arduino IDE to connect our program written in computer with Arduino. All the program and algorithms are written in this IDE which is then implemented in Arduino.

Chapter 2: LITERATURE REVIEW

Shortest-Path bots are robots designed to travel along specific routes without human intervention. Many research has explored different ways to make these robots move accurately and safely.

2.1. In the research performed by Mustafa Engin and Dilsad Engin proposed a Path-Planning of Line Follower Robot where they developed a line follower wheeled mobile robot. It used LM3S811 as the microcontroller as the main controller to react towards the data received from IR sensos to give fast, smooth, accurate and safe movement in particular structured environment. Their experimental results showed that the dynamic PID (Proportional-Integral-Derivative Controller) algorithm can be performed under the system real-time requirements [1].

2.2. In the project made by Khin Saw and Lae Yin Mon proposed a Design and Construction of Line Following Robot using Arduino where they developed a Line Following Robot where they used Arduino Nano as microcontroller. Similar to our project, the motors are controlled by L293D motor driver. Direction of the robot car is controlled by signal from the IR sensor. It sensed black line through three IR sensors used in the robot [2].

2.3. In the research of Abhijit Pathak, Refat Khan Pathan and their team proposed a Line Follower Robot for Industrial Manufacturing Process where they developed Line Follower Robot. They have used Arduino Uno R3 as a microcontroller. They have used Adafruit motor shield as a motor driver which can control up to 4 dc motor or 2 stepper motors. They have used Digital IR Sensor array which detect lines or patterns. They have just made a line following robot using IR sensor array and have not used any algorithm [3].

2.4. In the project made by Tanisha Chawada, Ayush Sharma and Aditya Medatwal proposed a Shortest Path Finder Robot Using Arduino Uno. They used Arduino Uno as microcontroller. Their project was very similar to our project but they used Uno instead of Mega. They also used Dijkstra Algorithm to find the shortest distance and implemented these using C++ programming Language, Arduino Library and Arduino IDE. The project was aimed at developing a low-cost and efficient solution for autonomous navigation of robots [5].

These are some of the research and projects done by other people they used very similar technology to our project. Some used Arduino Uno or Nano and some used Dijkstra Algorithm whereas some used IR sensor array which did not require any algorithm.

Chapter 3: METHODOLOGY

3.1. Introduction

When the word robot comes to our mind, we often tend to think it is some kind of a very sophisticated machine that has its own capability to think and act as close as to human. In our research we find it a very heavy to use the word robot. This is the reason we choose to give our idea as Shortest-Path Bot. Our bot utilizes the data from the sensor and acts accordingly. For this we use sensors such as Ultrasonic sensor that senses the obstacle from the sound it produces. Another important sensor in our work is IR sensor which detects the path.

The track of the bot is modelled on the white cloth surface where the roads will be of the white color and the lane of the road is of black color (Black Tape). The four IR sensors mounted on the bottom of the bot transmits and receives the IR light waves. The bot utilizes these reflected waves in order to follow its path. The bot sends the received signal to the micro-controller which will then decide what to do and it sends the appropriate signals to the motor driver. The motor driver will transmit the signals to the corresponding motors. To move the bot to left direction, the right motor rotates forward while the left motor rotates backward which finally cause the robot to move left.

As we intend to find the shortest path between the Source and Destination. In our project we use Ultrasonic Sensor to identify the destination. This sensor works on the basis of ultrasonic sound. When the obstacle is seen then the destination is known and stops the bot. The shortest path is identified with the implementation of the shortest path left hand algorithm.

3.2. Block Diagram

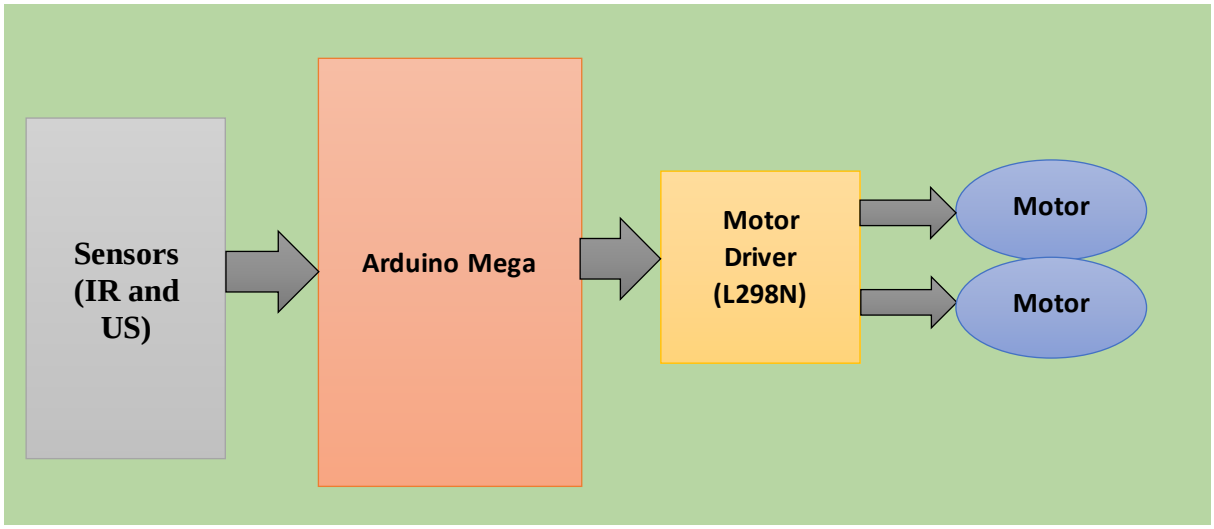
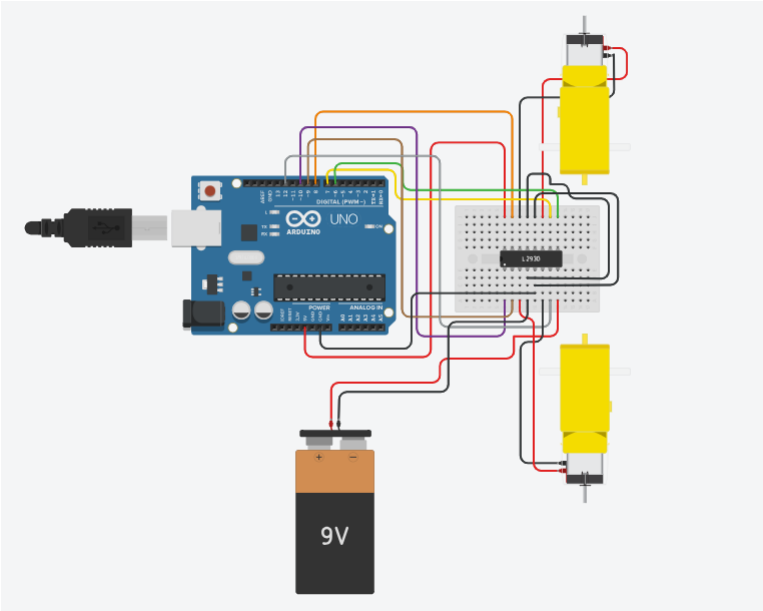


Fig-3.2 Block Diagram of Shortest-Path Bot

3.3. Methodology

Circuit Connection:

Fig-3.3.1 Simulation Circuit Connection between Motor Driver and Arduino:



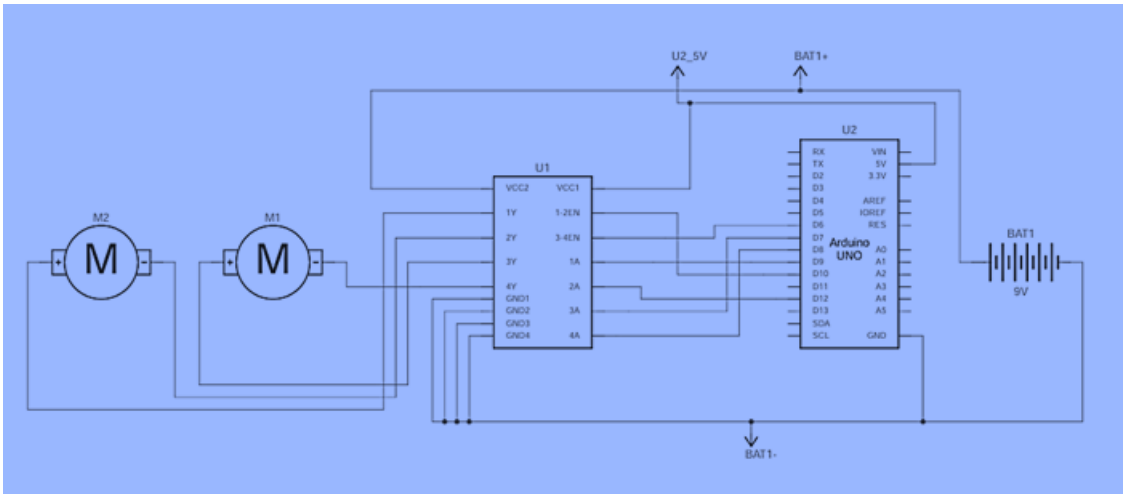
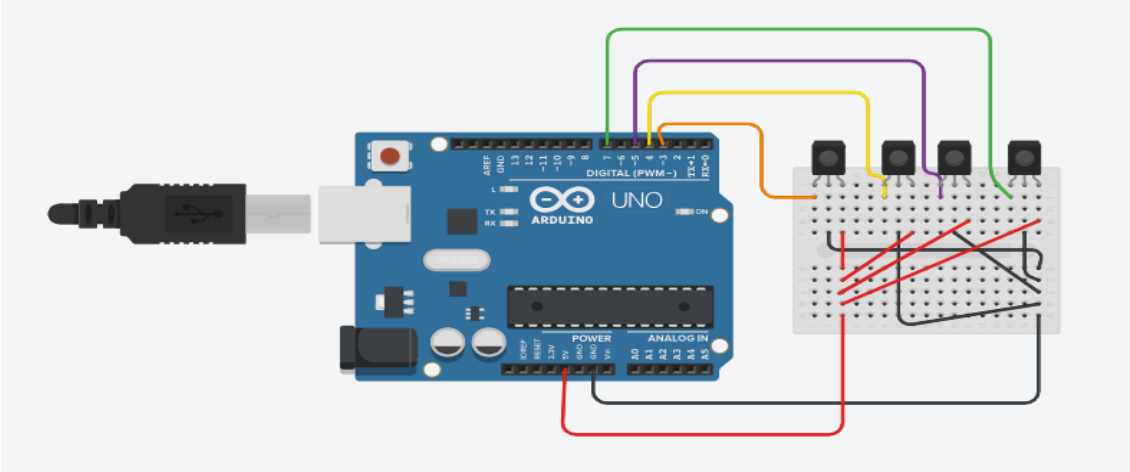


Fig-3.3.2 Simulation Circuit Connection between 4 IR Sensor and Arduino:



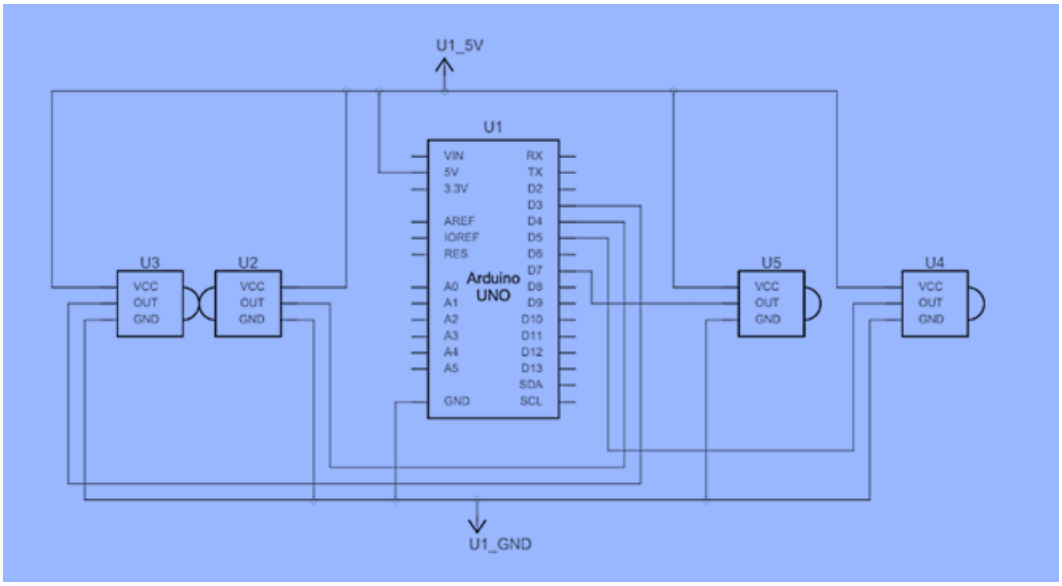
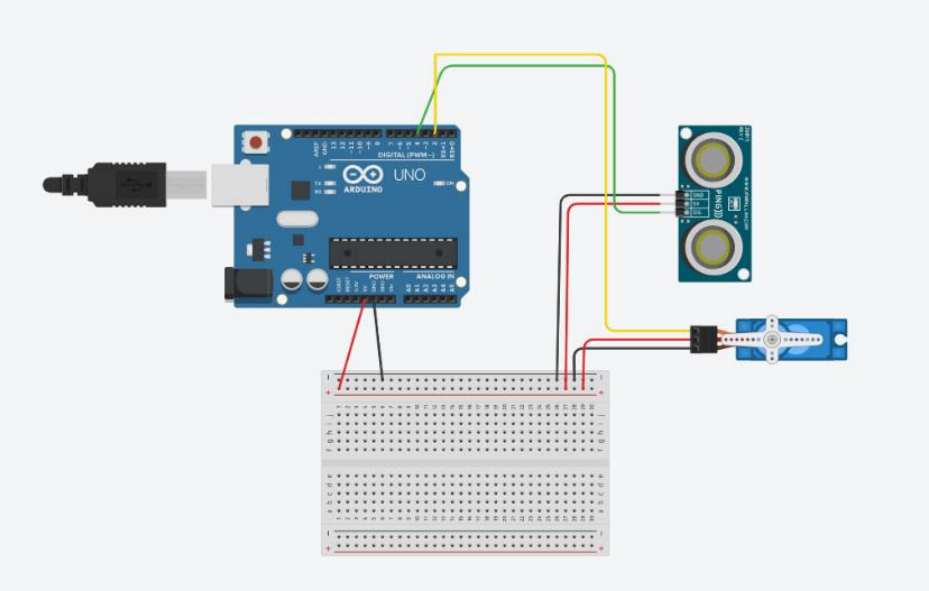


Fig-3.3.3 Simulation Circuit Connection between Ultra-Sonic Sensor, Servo Motor and Arduino:



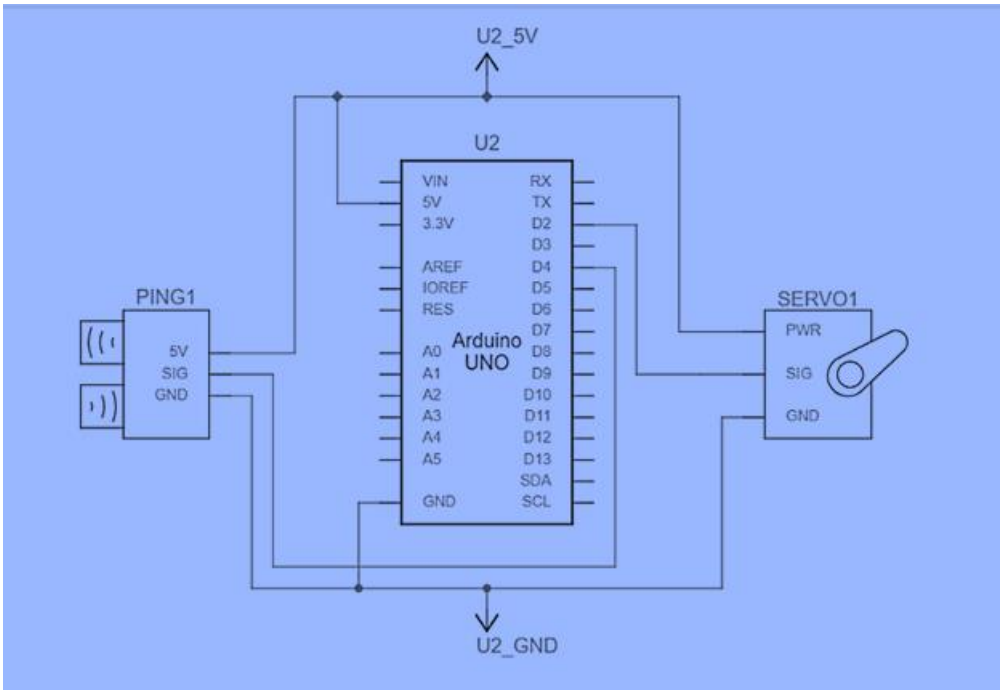
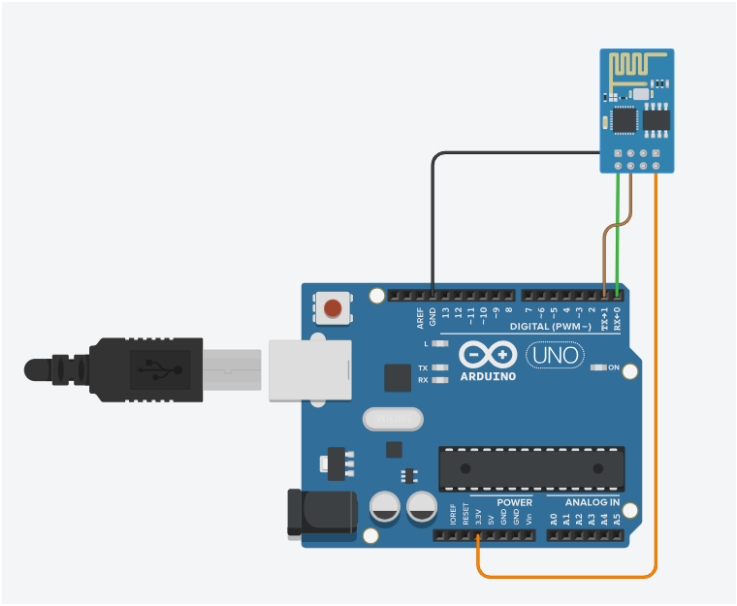
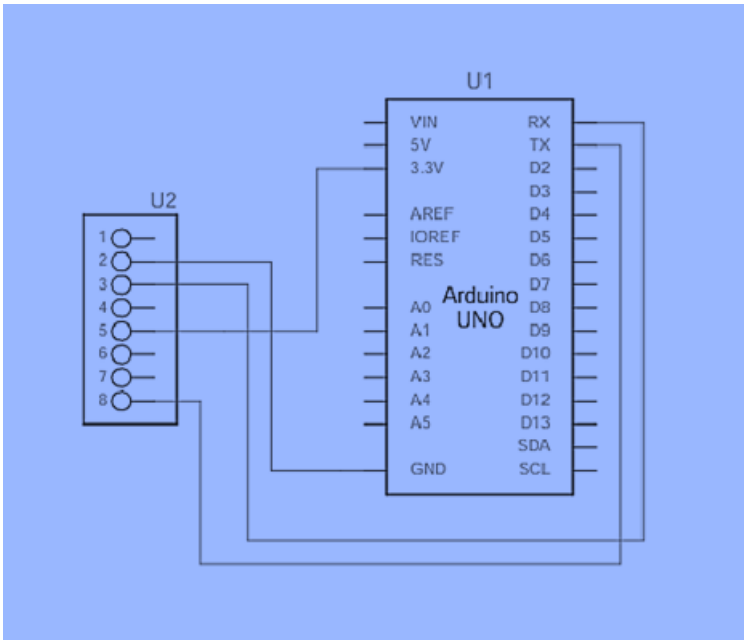


Fig-3.3.3 Simulation Circuit Connection between Bluetooth Module and Arduino:





3.4. Program Code:

Mode 1:

```
#include<string.h>
#include<EEPROM.h>
// IR Sensors
int irSens1 = 52;      // right most sensor
int irSens2 = 53;
int irSens3 = 50;
int irSens4 = 51;      // left most sensor

int read1, read2, read3, read4;

int en1=2;
int in1=27;
int in2=23;
int in3=24;
int in4=25;
int en2=3;

int motorSpeed=70;
int pauseTime=1000;
int leftSpeed, rightSpeed;
int wTime=110;
int addr=0;
int trigPin=9;
int echoPin=10;
float distance;
float tRead;

char path[50];


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  //right motor
  pinMode(en1, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);

  //left motor
  pinMode(en2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  //IR sensors
  pinMode(irSens1, INPUT);
  pinMode(irSens2, INPUT);
  pinMode(irSens3, INPUT);
  pinMode(irSens4, INPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  //Serial.println("Gadi gudirako cha");

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  tRead=pulseIn(echoPin, HIGH);
  digitalWrite(trigPin, LOW);
  distance=(0.0343*tRead)/2;
```

```
  Serial.print("Distance: ");
  Serial.println(distance);
}

void ultraSonic(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  tRead=pulseIn(echoPin, HIGH);
  digitalWrite(trigPin, LOW);
  distance=(0.0343*tRead)/2;
}

void readIRSens(){
  read1=digitalRead(irSens1);
  read2=digitalRead(irSens2);
  read3=digitalRead(irSens3);
  read4=digitalRead(irSens4);
}

void assignSpeed(){
  if (read4==0 && read3==0 && read2==0 && read1==0){
    //0
    // EEPROM.update(addr, 'U');
    path[addr]='U';
    addr++;
    littleStop();
    Serial.println("Taking U-turn");
    takeYouTurn();
  }
  else if(read4==0 && read3==0 && read2==0 && read1==1){
    //1
    //This means the sensor either the bot has turned very quickly while taking sharp left turn or while moving straight it moved very quickly that
    //inertia of the robot pulled the robot toward left. In this case you have to turn toward right until read3=1
    read3=digitalRead(irSens3);
    leftSpeed=motorSpeed;
    rightSpeed=-motorSpeed;
    Serial.println("Case 1 occured");
    while(read3!=1){
      driveMotor(leftSpeed, rightSpeed);
      read3=digitalRead(irSens3);
    }
  }
  else if(read4==0 && read3==0 && read2==1 && read1==0){
    //2
    //slightly turn right
    leftSpeed=motorSpeed;
    rightSpeed=-motorSpeed;
    read3=digitalRead(irSens3);
    while(read3!=1){
      driveMotor(leftSpeed, rightSpeed);
      read3=digitalRead(irSens3);
    }
  }
  else if(read4==0 && read3==0 && read2==1 && read1==1){
    //3
    //This can't be full left turn probably the bot moved very fast and IR sensor 2 and IR sensor 1 came to the track
    //In this case you will have to turn toward right
    leftSpeed=motorSpeed;
    rightSpeed=-motorSpeed;
    read3=digitalRead(irSens3);
```

```
  while(read3!=1){
    driveMotor(leftSpeed, rightSpeed);
    read3=digitalRead(irSens3);
  }
 }
 else if(read4==0 && read3==1 && read2==0 && read1==0){
  //4
  //This means IR sensor 2 is out of place so sligthy turn left;
  leftSpeed=-motorSpeed;
  rightSpeed=motorSpeed;
  read2=digitalRead(irSens2);
  while(read2!=1){
    driveMotor(leftSpeed, rightSpeed);
    read2=digitalRead(irSens2);
  }
 }
 else if(read4==0 && read3==1 && read2==0 && read1==1){
  //5
  //This can never happen
 }
 else if(read4==0 && read3==1 && read2==1 && read1==0){
  //6
  //Means both sensor are at correct position so just move forward;
  leftSpeed=motorSpeed;
  rightSpeed=motorSpeed;
  driveMotor(leftSpeed, rightSpeed);
 }
 else if(read4==0 && read3==1 && read2==1 && read1==1){
  //7
  //This is probable right turn here you need to check if there is road ahead or not;
  littleStop();
  smallForwardMovement();
  read2=digitalRead(irSens2);
  read3=digitalRead(irSens3);
  if (read3==1 && read2==1){
    // EEPROM.update(addr,'S');
    path[addr]='S';
    addr++;
    Serial.println("Taking straight instead of right1");
    leftSpeed=motorSpeed;
    rightSpeed=motorSpeed;
    driveMotor(leftSpeed, rightSpeed);
  }
  else if(read3==0 && read2==1){
    Serial.println("Taking straight instead of right2");
    // EEPROM.update(addr,'S');
    path[addr]='S';
    addr++;
    littleStop();
    leftSpeed=motorSpeed;
    rightSpeed=-motorSpeed;
    while(read3!=1){
      driveMotor(leftSpeed, rightSpeed);
      read3=digitalRead(irSens3);
    }
  }
  else if(read3==1 && read2==0){
    Serial.println("Taking straight instead of right3");
    // EEPROM.update(addr,'S');
    path[addr]='S';
    addr++;
    littleStop();
```

```
    leftSpeed=-motorSpeed;
    rightSpeed=motorSpeed;
    while(read2!=1){
      driveMotor(leftSpeed, rightSpeed);
      read2=digitalRead(irSens2);
    }
   }
   else{
    // EEPROM.update(addr, 'R');
    path[addr]='R';
    addr++;
    littleStop();
    Serial.println("Taking right turn");
    sharpRightTurn();
   }
  }
  else if(read4==1 && read3==0 && read2==0 && read1==0){
   //8
   //This condition can come during the sharp right turn or when the bot moves so fast so that it goes to very right
   //So here you need to turn the bot to left;
   read2=digitalRead(irSens2);
   leftSpeed=-motorSpeed;
   rightSpeed=motorSpeed;
   while(read2!=0){
    driveMotor(leftSpeed, rightSpeed);
    read2=digitalRead(irSens2);
   }
  }
  else if(read4==1 && read3==0 && read2==0 && read1==1){
   //9
   //This condition must not come at any cost.
  }
  else if(read4==1 && read3==0 && read2==1 && read1==0){
   //A
   //If the is the sensor output then the sensor is probably not working good so tune your sensor properly.
  }
  else if(read4==1 && read3==0 && read2==1 && read1==1){
   //B
   //If the is the sensor output then the sensor is probably not working good so tune your sensor properly.
  }
  else if(read4==1 && read3==1 && read2==0 && read1==0){
   //C
   //This condition means the robot moved a bit faster so you'll need to turn left your robot toward left until read4=0
   read2=digitalRead(irSens2);
   leftSpeed=-motorSpeed;
   rightSpeed=motorSpeed;
   while(read2!=1){
    driveMotor(leftSpeed, rightSpeed);
    read2=digitalRead(irSens2);
   }
  }
  else if(read4==1 && read3==1 && read2==0 && read1==1){
   //D
   //This condition can't come. This comes only if the IR sensor 2 is not sensing properly.
  }
  else if(read4==1 && read3==1 && read2==1 && read1==0){
   //E
   //This means left turn occured
   // EEPROM.update(addr, 'L');
```

```
    path[addr]='L';
    addr++;
    littleStop();
    Serial.println("Taking left turn as there is no road ahead");
    sharpLeftTurn();
  }
  else if(read4==1 && read3==1 && read2==1 && read1==1){
    //F
    //This means the robot has reached to the T join or Plus(+) join;
    // EEPROM.update(addr,'L');
    path[addr]='L';
    addr++;
    littleStop();
    Serial.println("Taking sharp left turn though there is road ahead");
    sharpLeftTurn();
  }
}
void littleStop(){
  leftSpeed=0;
  rightSpeed=0;
  driveMotor(leftSpeed, rightSpeed);
  delay(pauseTime);
}
void takeYouTurn(){
  leftSpeed=-(motorSpeed+5);
  rightSpeed=motorSpeed+5;
  read3=digitalRead(irSens3);
  while(read3!=1){
    driveMotor(leftSpeed, rightSpeed);
    read3=digitalRead(irSens3);
  }
}
void smallForwardMovement(){
  rightSpeed=motorSpeed;
  leftSpeed=motorSpeed;
  driveMotor(leftSpeed, rightSpeed);
  delay(wTime);
}
void sharpRightTurn(){
  smallForwardMovement();
  read3=digitalRead(irSens3);
  leftSpeed=motorSpeed+5;
  rightSpeed=-(motorSpeed+5);
  while(read3!=1){
    driveMotor(leftSpeed, rightSpeed);
    read3=digitalRead(irSens3);
  }
}
void sharpLeftTurn(){
  smallForwardMovement();
  leftSpeed=-motorSpeed;
  rightSpeed=motorSpeed;
  read2=digitalRead(irSens2);
  // while(read!=0) loop is for T junction or + type of turns
  while(read2!=0){
    digitalWrite(leftSpeed, rightSpeed);
    read2=digitalRead(irSens2);
  }
  while(read2!=1){
    driveMotor(leftSpeed, rightSpeed);
    read2=digitalRead(irSens2);
  }
}
void driveMotor(int lSpeed, int rSpeed){
  analogWrite(en1, abs(rSpeed));
  analogWrite(en2, abs(lSpeed));
```

```
  if(lSpeed>0){
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
  }
  else if(lSpeed<0){
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
  }
  else{
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
  }
  if(rSpeed>0){
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
  }
  else if(rSpeed<0){
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
  }
  else{
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
  }
}
```

Mode 2:

```
#include <stdio.h>
#include <string.h>
int main(){
char path[]="SRULLSLRRULLLLLLR";
   int len=strlen(path);
   char replace[4];
   char prevPath[20];
   while(strcmp(prevPath,path)!=0){
      strcpy(prevPath, path);
      int i;
      for(i=0;i<len;i++){
         if (path[i]=='U'){
            // printf("U found\n");
            replace[0]=path[i-1];
            replace[1]=path[i];
            replace[2]=path[i+1];
            if (strcmp("LUL",replace)==0){
               path[i-1]='S';
               // printf("Matched!\n");
               // break;
               len=len-2;
               int j=0;
               for(j=i;j<len;j++){
                  path[j]=path[j+2];
               }
               path[j]='\0';
            }
            if (strcmp("SUS", replace)==0){
               path[i-1]='U';
               len=len-2;
               int j=0;
               for(j=i;j<len;j++){
                  path[j]=path[j+2];
               }
               path[j]='\0';
            }
            if (strcmp("SUL", replace)==0){
               path[i-1]='R';
               len=len-2;
               int j=0;
               for(j=i;j<len;j++){
```

```
            path[j]=path[j+2];
        }
        path[j]='\0';
    }
    if (strcmp("RUL",replace)==0){
        path[i-1]='U';
        len=len-2;
        int j=0;
        for(j=i;j<len;j++){
            path[j]=path[j+2];
        }
        path[j]='\0';
    }
    if (strcmp("LUS", replace)==0){
        path[i-1]='R';
        len=len-2;
        int j=0;
        for(j=i;j<len;j++){
            path[j]=path[j+2];
        }
        path[j]='\0';
    }
    if (strcmp("LUR",replace)==0){
        path[i-1]='U';
        len=len-2;
        int j=0;
        for(j=i;j<len;j++){
            path[j]=path[j+2];
        }
        path[j]='\0';
    }
    }
    }
    }
    printf("need to travel:%s", path);
    return 0;
}
```

3.5. Algorithms

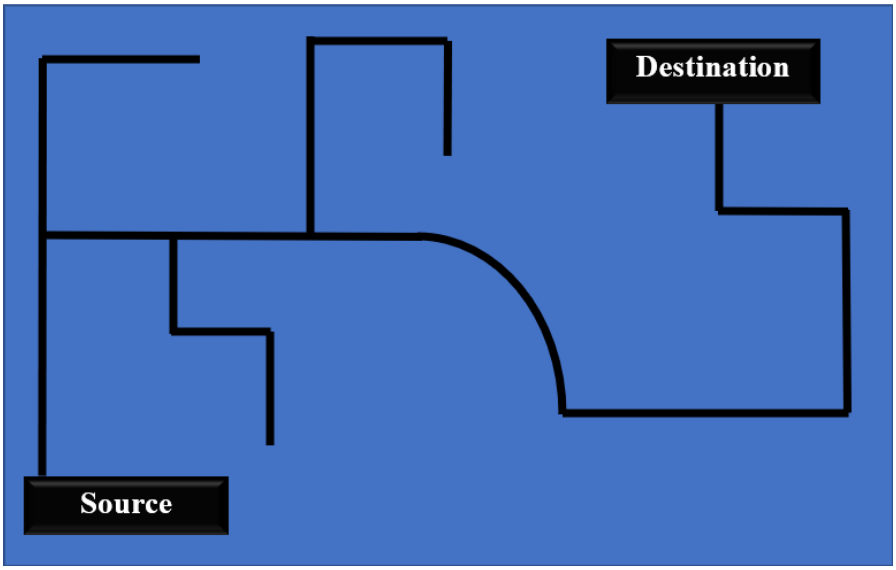


Fig-3.5 Final Path

Shortest Path Finder Algorithm

Two methods can be applied for traversing the path made up of black tape on white cloth. First is Right Hand Method and Second is Left Hand Method.

3.5.1 Right Hand Method

In right hand method, the robot will take right turn first. If there is no possibility of right turn then it will go straight and at last if there is only left turn then it will take that turn.

The precedence order is as follows:

Right > Straight > Left

It can be illustrated as follows:

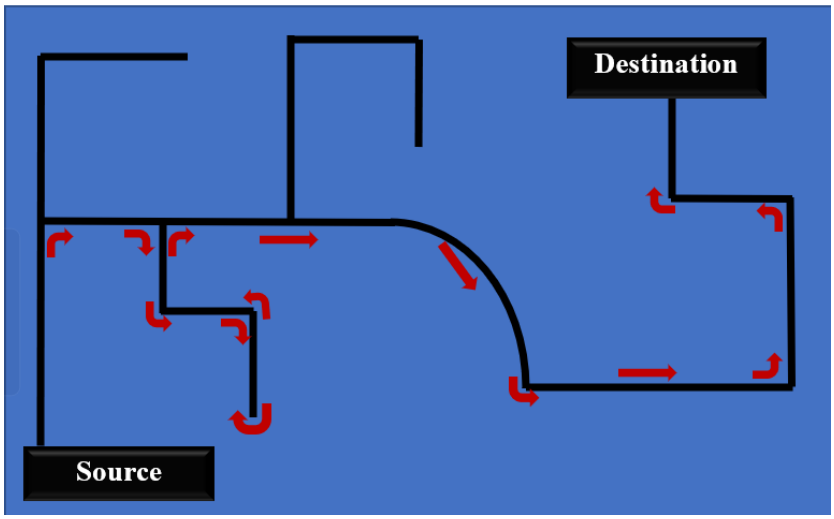


Fig-3.5.1 Right Hand Path

3.5.2 Left Hand Method

In left hand method, the robot will take left turn first. If there is no left turn then it will choose straight path and at last if there is only right turn then it will take that turn.

The precedence order is as follows:

Left > Straight > Right

It can be illustrated as follows:

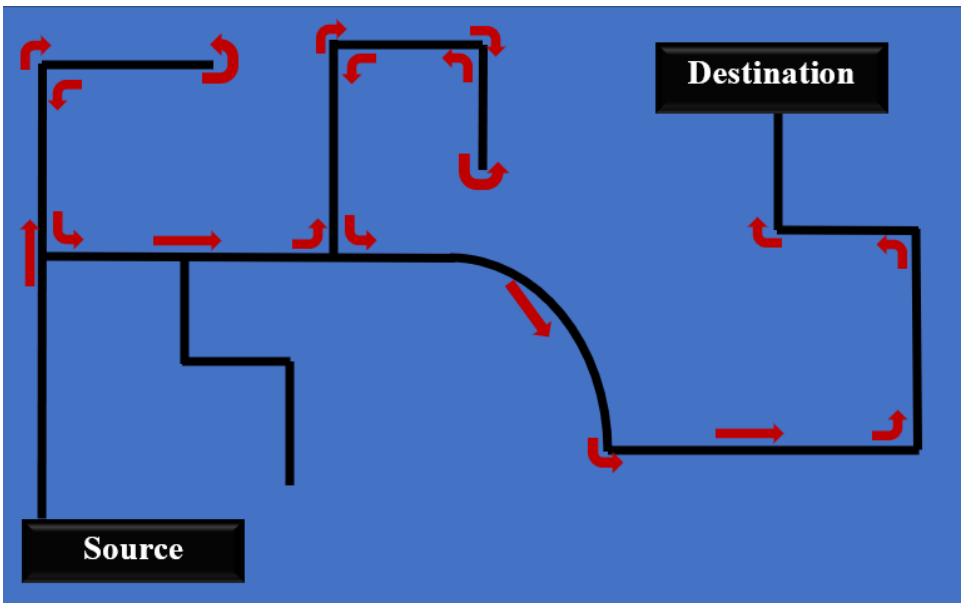


Fig-3.5.2 Left Hand Path

3.6. Theory

In this project we will use left hand method. While traversing the path following cases can be encountered. These are dead end, bridge, T, ends, curve, etc.

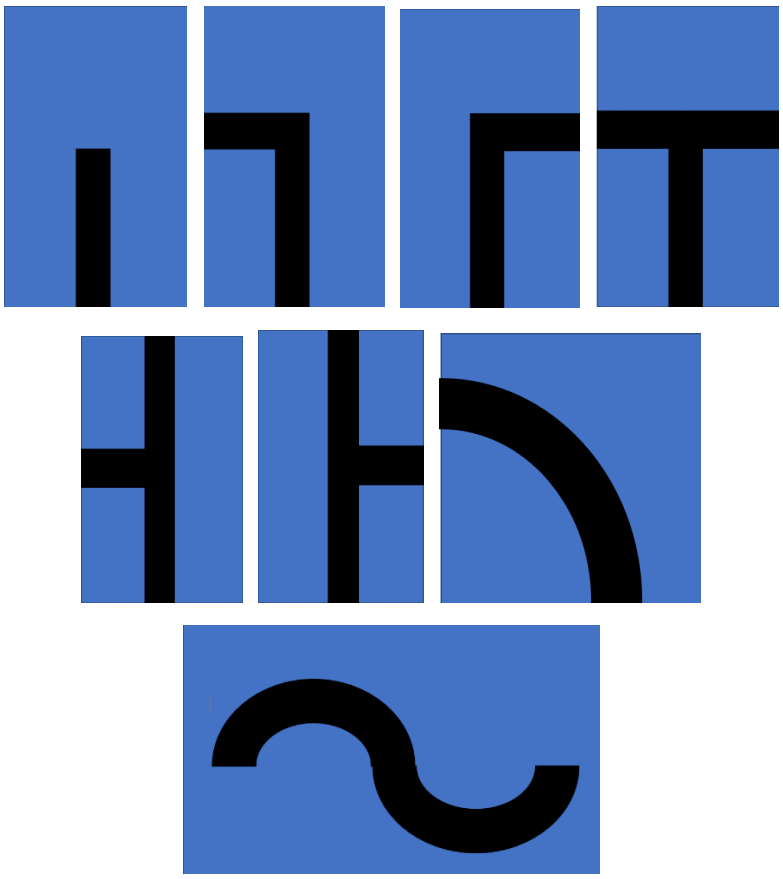


Fig-3.6.1 Patterns in Path

To memorize the path codes are given to them.

Left turn : L

Right turn: R

Straight : S

U-turn : U

For example,

In given path this code can be written as: -

- SRULLSLRRULLLLLLR

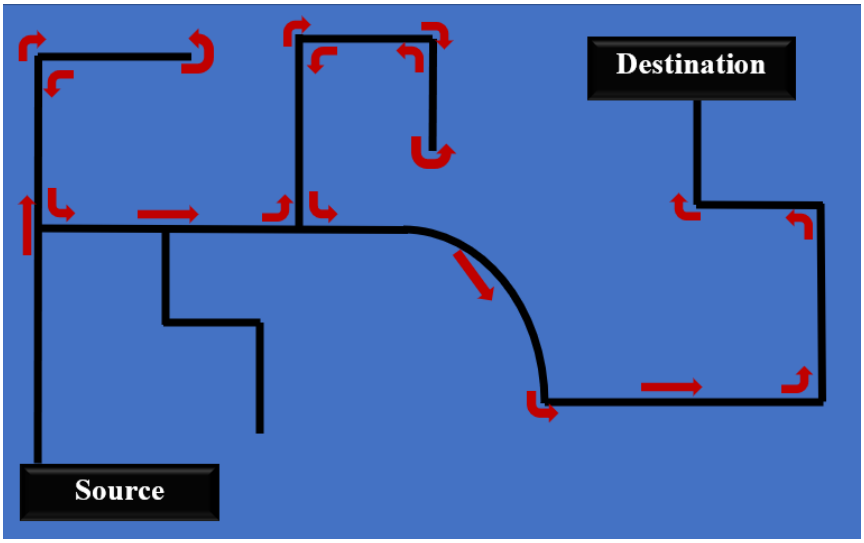


Fig-3.5.2 Left Hand Path

Now this path can be reduced by replacing with following Substitution:

LUR: U LUS: R RUL: U

SUL: R SUS: U LUL: S

The path can be optimized by:

- SRULLSLRRULLLLLLR
- SULSLRRULLLLLLR
- RSLRRULLLLLLR
- RSLRULLLLLR
- RSLULLLLR
- RSSLLLR

So, the path can be written as RSSLLLR.

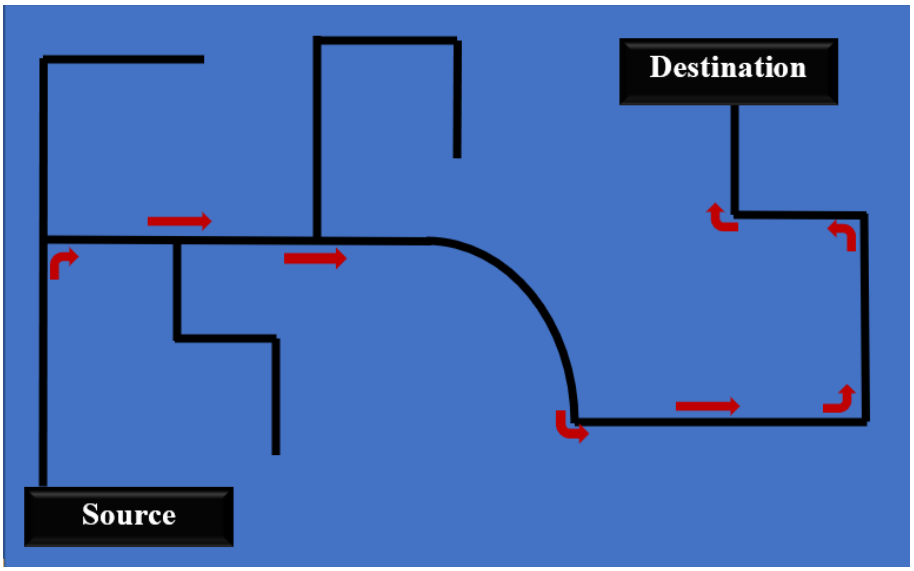


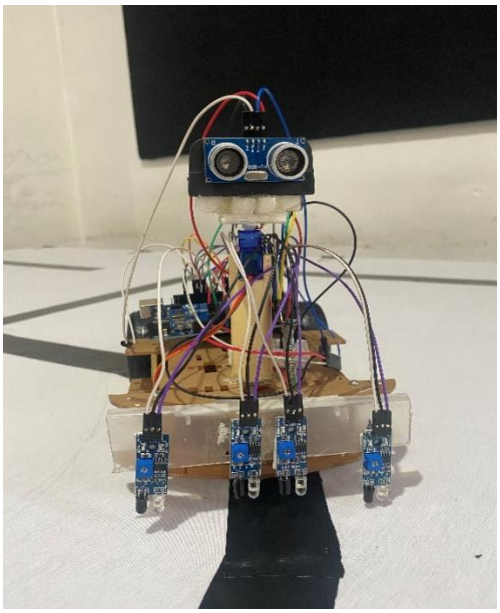
Fig-3.6.2 Final Shortest Path

Chapter 4: Result and Analysis

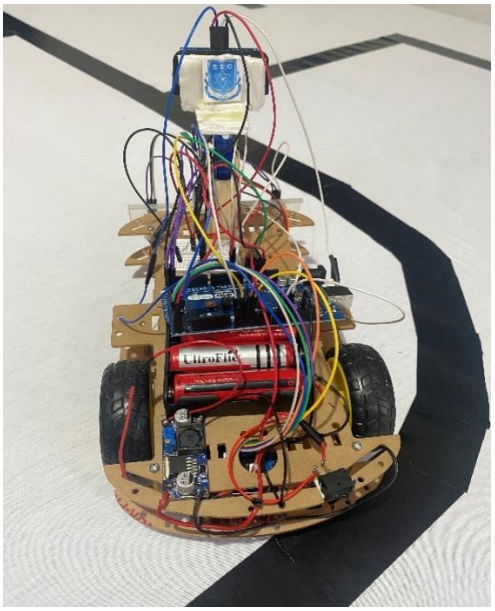
4.1 Result

Final completed Pictures of Shortest-Path Bot:

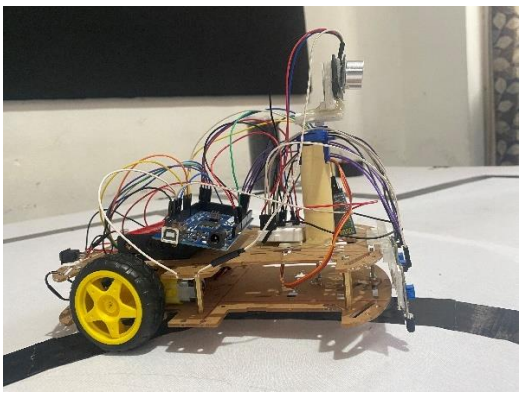
Front

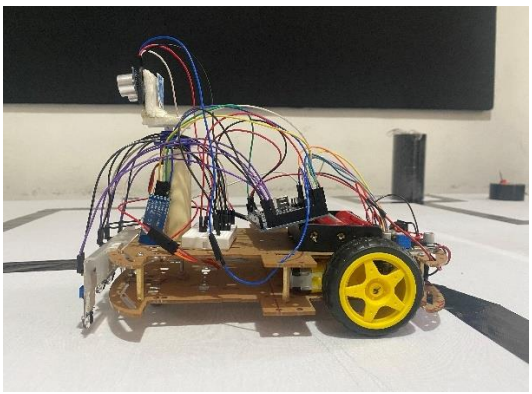


Back

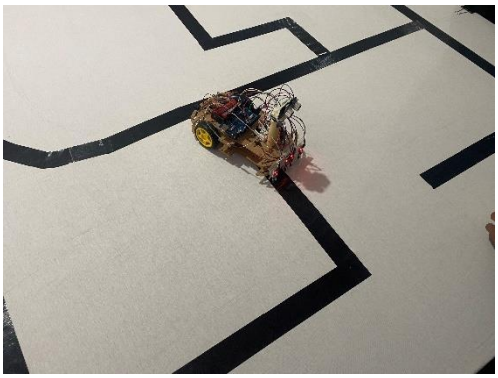


Side

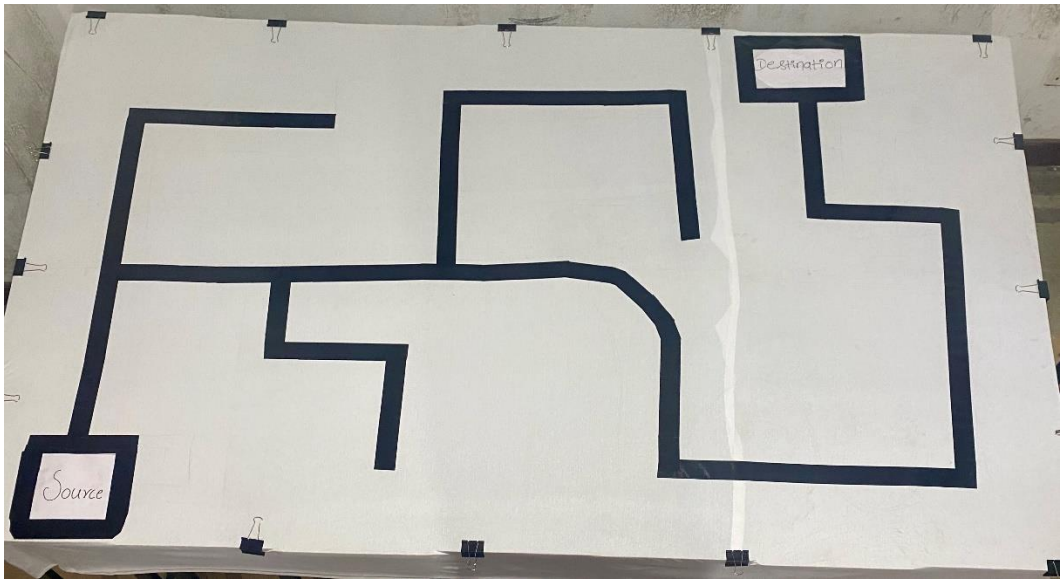




Bot on operation



Final Environment



4.2 Work Analysis

Table-4.2.1 IR Sensor Working mechanism:

We have used 4 IR Sensors. They work accordingly.

IR 1	IR 2	IR 3	IR 4	Remarks
0	0	0	0	U-turn
0	0	0	1	Slightly turn right
0	0	1	0	Slightly turn right
0	0	1	1	Turn right until detection
0	1	0	0	Slightly turn left
0	1	0	1	Not possible
0	1	1	0	Straight
0	1	1	1	Check ahead then turn right
1	0	0	0	Slightly turn left
1	0	0	1	Not possible
1	0	1	0	Not possible
1	0	1	1	Not possible
1	1	0	0	Turn left until detection
1	1	0	1	Not possible
1	1	1	0	Check ahead then turn left
1	1	1	1	T-turn (turn left)

Works Shortest-Path Bot can perform:

- **Follow the Black line smoothly**
  Bot detect and follow the black line in white surface with the help of IR sensor. Arduino analyze the data sent by IR sensor and sent signal to the motor driver which make bot movement smooth.
- **Detect the path, sharp turn and turn the bot accordingly**
  When IR sensor detects sharp turn, PWM frequency of Arduino changes which increases the speed of motor helping in turning of bot.
- **Autonomously detect the shortest path**
  After the detection of Source and Destination, bot autonomously uses left hand algebra reduce steps. Bot move according to the reduced steps to reach destination.
- **Send information covered by bot wirelessly through Bluetooth module**
  All the steps taken by the bot is sent to mobile/laptop by the help of Bluetooth module.

Problems Encountered:

- **Light Interference / IR Noise**
  Light detection of IR sensor is very sensitive. So, when we operate bot in high lighting condition or reflected condition, IR sensor struggle to detect the path.
- **Torque Problem**
  Due to uneven RPM of Bo motor, if we decrease the PWM there is risk of low torque which does not spin the wheel.
- **Inertia**
  Due to heavy weight, sometimes bot cannot stop in time.
- **Power Consumption**
  Use of different component drains the battery, due to which the component does not work properly. Batteries are connected in serial so drainage is heavy in one battery only.
- **Problem of Traction control**
  Wheel is made up of plastic so sometime wheel loses traction. Wheel spins but stays in one place.
- **Low synchronization of Bo Motor**
  Bo motor is very hard to synchronize, when the bot moves straight it turns automatically due to low synchronization.

Solution to those problems:

- **Exchanged older IR Sensor to new and more precise IR Sensor**
- **Covered IR Sensor with card board**
- **Check and changed the use of pins in Arduino**
- **Reduced the number of motor driver, Bo motor and Wheels**
- **Changed the environment for testing**
- **Changed the PWM frequency of Arduino**
- **Used DC-DC Buck Boost as a voltage regulator**

Problems not solved:

- **Problem of Traction Control**
  Due to use of plastic wheel in marble surface. There is still problem of wheel traction.
- **Power Consumption not fully solved**
  Even though we used DC-DC Buck Boost converter to regulate the voltage fluctuation. There is still problem of Power Consumption.
- **Low synchronization of Bo Motor**
  Synchronization of Bo motor is very hard so we used PWM frequency to control the Bo motor. But we were unable to make precise and smooth movement of the bot.

Table-4.2.2 Total Cost

Materials used in this Project with Their Total Cost:

S. N	Name of Materials	Price per piece (in Rs.)	Required Quantity	Total Price (in Rs.)
		College Funded		
1.	Arduino (Mega)	2000	1	**2000**
2.	Motor Driver L293D	300	1	**300**
3.	Ultrasonic Sensor	180	1	**180**
4.	Infrared Sensor	95	4	**380**
5.	Jumper Wire	5	48	**240**
6.	Bluetooth Module	700	1	**700**
7.	4 Wheel Set Chassis	1300	1	**1300**
8.	Rechargeable Battery (Lithium-Ion)	150	6	**900**
9.	Lithium-Ion Battery Charger	300	1	**300**
10.	Servo Motor	260	3	**780**
11.	2s Battery Holder	120	2	**120**
12.	Rocker Switch	30	1	**30**
Total				**7,230**
		Self-Funded		
14.	3s Battery Holder	140	1	**140**
15.	2s Battery Holder	120	1	**120**
16.	Motor Driver L298N	400	1	**400**
17.	Motor Driver L293D Chip	90	1	**90**
18.	DC-DC Buck Boost	175	1	**175**
19.	Binding Tape	100	10	**1000**
20.	Small Black Tape	25	10	**250**
21.	Chart Paper	30	4	**120**
22.	Double Tape	80	2	**160**
23.	Infrared Sensor	100	2	**200**
24.	Jumper Wire	5	50	**250**
25.	White Cloth	360	1	**360**
26.	Caster Wheel	75	2	**150**
27.	Soldering Wire	110	1	**110**
28.	Small Breadboard	45	2	**90**
Total				**3,615**
Sub Total				**10,845**

The above quotation is taken reference from SL Tech, Gusingal Chowk, Lalitpur.

Chapter 5: Conclusion and Future Enhancement

5.1. Conclusion

The development of a Shortest-Path Bot using Arduino provides a cost-effective and efficient solution for autonomous navigation of bot. By using Arduino, the system can be easily customized and modified according to specific requirements. However, the interaction with sensor has some limitations. Because due to intensity of environmental lighting effect leads to error in sensor signal. More over the battery should be recharged frequently making it suitable for various applications such as warehouse management, automated guided vehicles, and home automation. We are planning to further improve our Bot by implementing more advanced algorithms like Dijkstra Algorithm, A* Algorithm and Floyd-Warshall Algorithm and adding more sensors to optimize the path planning process in future project. Overall, the project provides a solid foundation for the development of autonomous navigation systems using Arduino and can serve as a starting point for more advanced robotics applications.

5.2. Future Enhancement

- **Proper implementation of Dijkstra Algorithm**
  In this project due to lack of time and resource we could not implement proper Dijkstra Algorithm. In future project we are planning to implement it and make our bot find shortest path more efficiently.

- **Convert Hysteresis Controller to PID Control**
  Our bot is using Hysteresis Controller (2-step or on-off controller) in other word bang-bang controller. It is a feedback controller that switches abruptly between two states. It is a control strategy that employs only two states: on (when the measurement is below the setpoint) and off (otherwise). This is roughly equivalent to a proportional loop with infinite gain.
  In future we will use PID (Proportional-Integral Derivative) controller. It uses a control loop feedback mechanism to control process variables and are the most accurate and stable controller.

- **Further Improvement of Bot using machine learning**
  We are planning to use machine learning in the future project. It will help identify our bot to detect source and destination itself. Bot can be made fully automatic. Processing, analyzing and implementing can be done by bot itself.

REFERENCES

[1] Mustafa Engin, Dilsad Engin (2012), *PATH-PLANNING OF LINE FOLLOWER ROBOT*, Bornova, 35100, Izmir, Turkey, 5th European DSP Education and Research Conference, https://www.researchgate.net/publication/261448382 .

[2] Khin Khin Saw, Lae Yin Mon (2019), *DESIGN AND CONSTRUCTION OF LINE FOLLOWING ROBOT USING ARDUINO,* Myanmar, International Journal of Trend in Scientific Research and Development (IJTSRD), https://www.ijtsrd.com/papers/ijtsrd23977 .

[3] Abhijit Pathak, Refat Khan Pathan, Amaz Uddin Tutul, Nishat Tahsin Tousi, Afsari Sultana Rubaba, Nahida Yeasmin Bithi (2018), *LINE FOLLOWER ROBOT FOR INDUSTRIAL MANUFACTURING PROCESS,* Chittagong, Bangladesh, Department of Computer Science and Engineering, BGC Trust University Bangladesh, https://www.researchgate.net/publication/327965269 .

[4] Mehran Pakdaman, M. Mehdi Sanaatiyan, Mahdi Rezaei Ghahroudi (2014), *A LINE FOLLOWER ROBOT FROM DESIGN TO IMPLEMENTATION: TECHNICAL ISSUES AND PROBLEMS*, Babol, Iran, https://www.researchgate.net/publication/224132741 .

[5] Tanisha Chawada, Aayush Sharma, Aditya Medatwal (2023), *SHORTEST PATH FINDER ROBOT USING ARDUINO UNO,* Department of Electronics and Communication Engineering, Indore Institute of Science and Technology, Indore 453331, India, International Journal of Creative Research thoughts (IJCRT), http://www.ijcrt.org/ .

[6] Gilles Brassard and Paul Bratley (2006), *FUNDAMENTALS OF ALGORITHMICS,* New Delhi – 110 00, Practice Hall of India Private Limited. P.P. (198-202)

[7] Yedidyah Langsam, Moshe J. Augenstein and Aaron M. Tenenbaum (2013), *DATA STRUCTURES USING C AND C++,* Brooklyn College, Delhi-110092, PHI Learning Private Limited, Second Edition. P.P. (547-549)

[8] E Balagurusamy (2021), *OBJECT-ORIENTED PROGRAMMING WITH C++*, Butt Road, St. Thomas Mount, Chennai-600016, Tamil Nadu, India, McGraw Hill Education (India) Private Limited.