

Project Report



Submitted by

Muhammad Usman-211280

Ahmed Javid-211282

Wadha Saddal-211268

Amna Zafar-211207

Submitted to

Sir Umer Farooq

**Department of Mechatronics Engineering
Air University
Islamabad**

INDEX

1 Preliminaries

- 1.1 Proposal
- 1.2 Initial feasibility
- 1.3 Technical Standards
- 1.4 Team Roles & Details
- 1.5 Work Break down structure
- 1.6 Gantt Chart
- 1.7 Estimated budgets

2 Project Conception

- 2.1 Introduction
- 2.2 Literature Review
- 2.3 List of features and operational specification of our project
- 2.4 Project Development Process
- 2.5 Basic block diagrams of whole system and subcomponents in Draw.io or similar tool

3 Mechanical Design

- 3.1 Mechanism selection
- 3.2 Platform Design
- 3.3 Material Selection and choices
- 3.4 3D CAD design and Analysis Screen Shots and Explanation
- 3.5 Actuators with speciation and datasheet
- 3.6 Deliverable of Complete CAD with discussion

4 Software/Firmware Design

- 4.1 Input Output pin outs
- 4.2 Controller Selections with features
- 4.3 Software Design details & user Requirements
- 4.4 State Machine & System flow diagram
- 4.5 Block Diagram
- 4.6 User cases for running your system (Test Cases)
- 4.7 Deliverable of complete commented code as per state machine and discussion

5 Simulations and final Integrations

- 5.1 Integrations and testing all hardware and software component separately
- 3.5 Simulation PCB and 3D view
- 5.3 Simulation Cad
- 5.4 Actual wiring plan with color codes
- 3.5 Discussion on Simulations with possible Challenges

6 System Test phase

- 6.1 Final testing
- 6.2 Project actual Pictures

7 Project management

- 7.1 Everyone must write one page about how he executed his role in project
- 7.2 Comment individually about success / failure of your project
- 7.3 A paragraph about team member positive and negative aspect
- 7.4 Final bill of material list and paragraph about project budget allocation
- 7.5 Give a word count how many words each member write in final report
- 7.6 Risk management that you learned

8 Feedback for project and course

CHAPTER 1

PRELIMINARIES

1.1 Proposal

The objective was to construct a line follower robot according to the guidelines provided Line follower robot is one kind of autonomous robot which follows a line until that line exists. Generally, the line is drawn on the floor. It can be either black or white. The line can also be normal visible color or invisible magnetic field or electric field. The robot follows the line by using Infra-Red Ray (IR) sensors. There are five IR sensors which makes it an IR sensor array. These sensors read the line and send that reading to Arduino Nano and then control the robot movement. We also using ESP 32 for different tasks like to display the voltage and current values through OLED, Color sensor to deduct the red and black obstacle, Encoders to measure the distance that follow by the robot and store the distance in the SD card module. In this paper, the authors will explain about the robot design, implementation, coding, testing, problems they faced and their solutions.

1.2 Initial feasibility

The initial phase of the project was divided in to two parts first we understood the turtle robot simulation and programmed it and delivered our understandings of the simulation in the form of a video

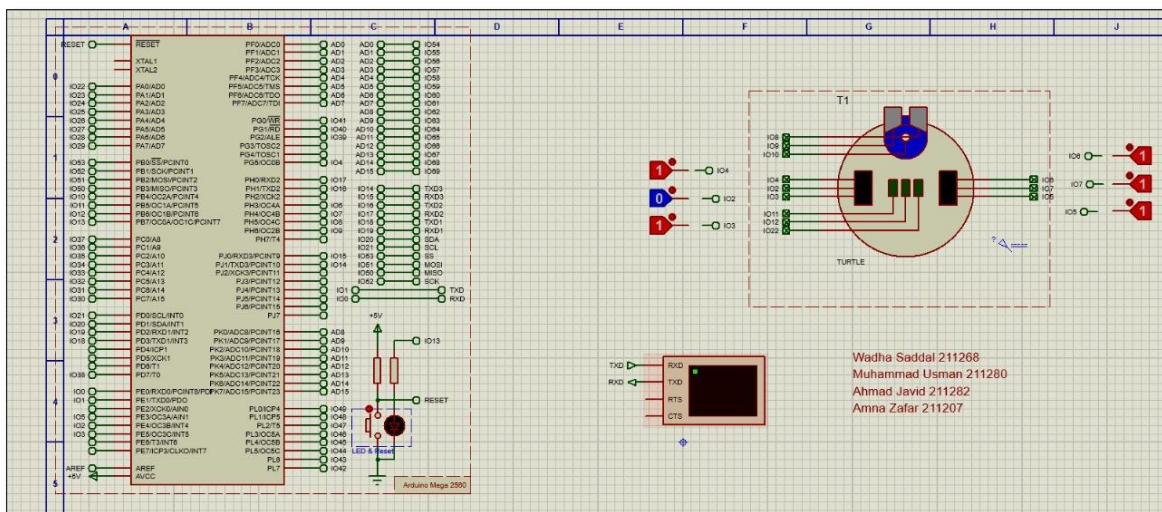
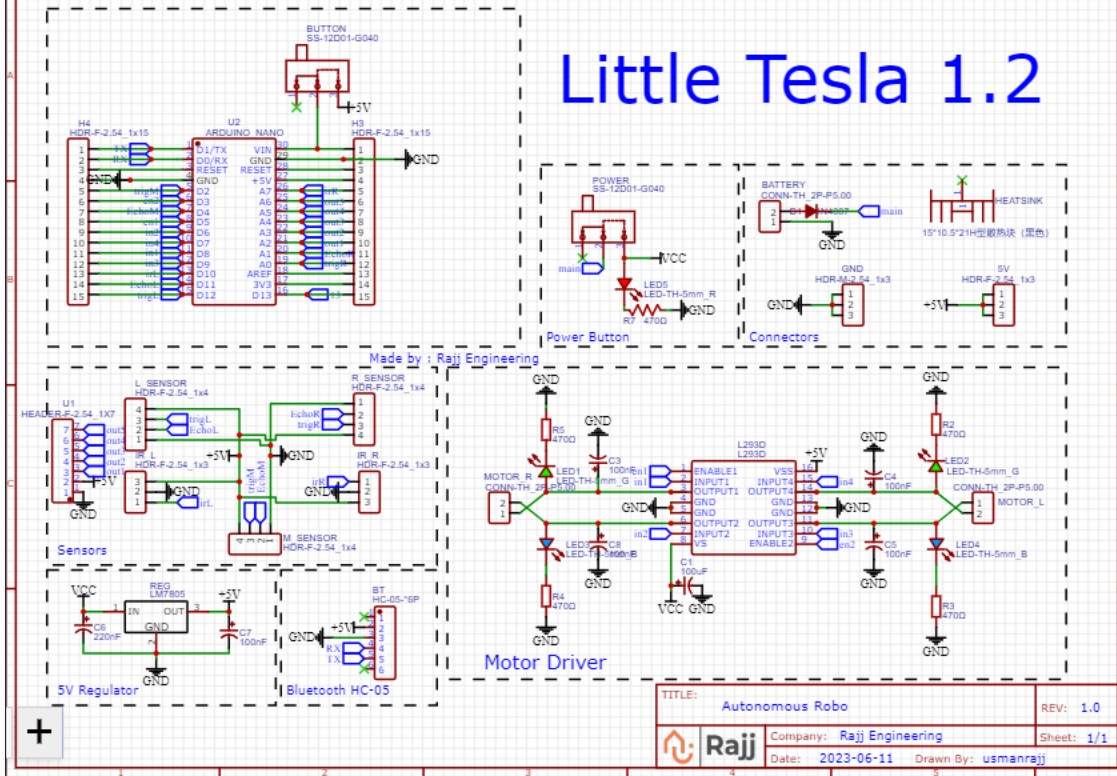

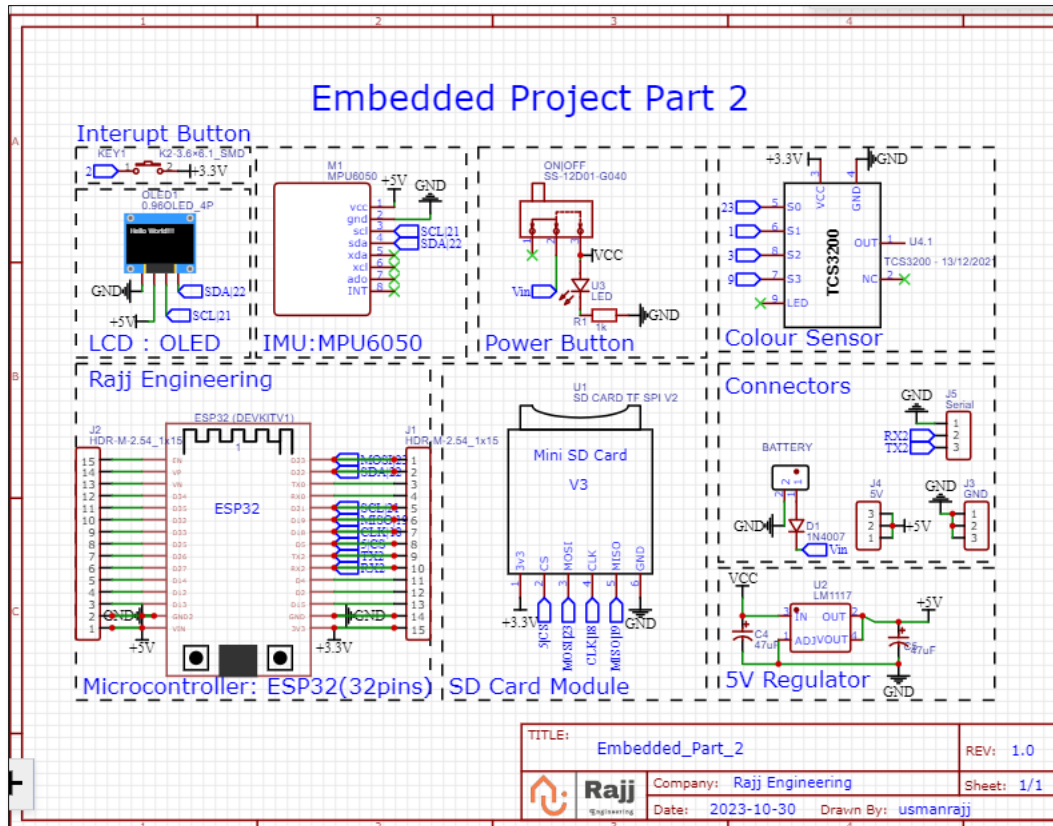


Figure 1

Little Tesla 1.2



TITLE:	Autonomous Robo	REV:	1.0
 Raji	Company: Raji Engineering	Sheet:	1/1
Date: 2023-06-11	Drawn By: usmanraji		



For the part two of this phase, we designed a basic line follower design using 5 IR sensor array 2 motors OLED and L293D motor driver this basic simulation helps us to understand the core working of the line follower robot we won't be adding complete simulation specifications for the scope of this report as that was only used as a medium to develop a better understanding.

```
#define irleft 10
#define irright A7
#define ltrig 12
#define lecho 11
#define rtrig A0
#define recho A1
#define mtrig 2
```

```
#define mecho 4

#define in1 8
#define in2 6
#define in3 9
#define in4 7
#define enA 5
#define enB 3
#define out5 A6
#define out4 A5
#define out3 A4
#define out2 A3
#define out1 A2


int ir1,ir2,ir3,ir4,ir5;    // Variable to store data from IR sensors


int black=0;                // Black line means Ouput of sensor = 0
int white=1;                // White line means Ouput of sensor = 1
int line=white;             // Which Line you want to Follow


int SPEED=140;              // Set Speed B/W 0-255


void linefollowing();

void ir();

void left();
```

```

void forward_left();

void backward_left();

void right();

void forward_right();

void backward_right();

void forward();

void backward();

void Stop();


void setup() {

    pinMode(out1,INPUT);        // Ouput from IR sensor
    pinMode(out2,INPUT);        // Ouput from IR sensor
    pinMode(out3,INPUT);        // Ouput from IR sensor
    pinMode(out4,INPUT);        // Ouput from IR sensor
    pinMode(out5,INPUT);        // Ouput from IR sensor


    pinMode(in1, OUTPUT);       // CounterClockwise Motor A
    pinMode(in2, OUTPUT);       // Clockwise Motor A
    pinMode(in3, OUTPUT);       // CounterClockwise Motor B
    pinMode(in4, OUTPUT);       // Clockwise Motor B
    pinMode(enA, OUTPUT);       // Enable Ouputs for Motor A (PWM)
    pinMode(enB, OUTPUT);       // Enable Ouputs for Motor B (PWM)
    Serial.begin(9600);         // initialize serial monitor
    analogWrite(enA,SPEED);
    analogWrite(enB,SPEED);

```



```
    delay(10);

}

void loop() {

    ir();
    linefollowing();
}

void linefollowing(){

    if (ir1!=line && ir2!=line && ir3!=line && ir4!=line && ir5!=line ){
        //00000

        Stop();

    }

    else if (ir1!=line && ir2!=line && ir3!=line && ir4!=line && ir5==line ){
        //00001

        forward_right();

        while(ir3!=line){

            ir();

        }

    }

    else if (ir1!=line && ir2!=line && ir3!=line && ir4==line && ir5!=line ){
        //00010

        forward_right();

    }

    else if (ir1!=line && ir2!=line && ir3!=line && ir4==line && ir5==line ){
        //00011

        forward_right();
```

```
    while(ir3!=line){  
        ir();  
    }  
}  
  
else if (ir1!=line && ir2!=line && ir3==line && ir4!=line && ir5!=line ){  
//00100  
  
    forward();  
}  
  
else if (ir1!=line && ir2!=line && ir3==line && ir4!=line && ir5==line ){  
//00101  
  
    forward();  
}  
  
else if (ir1!=line && ir2!=line && ir3==line && ir4==line && ir5==line  
) { //00111  
  
    forward_right();  
}  
  
else if (ir1!=line && ir2==line && ir3!=line && ir4!=line && ir5!=line ){  
//01000  
  
    forward_left();  
  
    while(ir3!=line){  
  
        ir();  
    }  
}  
  
else if (ir1!=line && ir2==line && ir3!=line && ir4!=line && ir5==line ){  
//01001  
  
    forward_left();
```

```
    while(ir3!=line){  
        ir();  
    }  
}  
  
else if (ir1!=line && ir2==line && ir3!=line && ir4==line && ir5!=line ){  
//01010  
    forward();  
}  
  
else if (ir1!=line && ir2==line && ir3!=line && ir4==line && ir5==line  
) { //01011  
    forward_right();  
    while(ir3!=line){  
        ir();  
    }  
}  
  
else if (ir1!=line && ir2==line && ir3==line && ir4!=line && ir5!=line ){  
//01100  
    forward_left();  
}  
  
else if (ir1!=line && ir2==line && ir3==line && ir4!=line && ir5==line  
) { //01101  
    forward_left();  
}  
  
else if (ir1!=line && ir2==line && ir3==line && ir4==line && ir5!=line  
) { //01110  
    forward();  
}
```

```

}

else if (ir1!=line && ir2==line && ir3==line && ir4==line && ir5==line
){ //01111

    forward_right();

}

else if (ir1==line && ir2!=line && ir3!=line && ir4!=line && ir5!=line ){
//10000

    forward_left();

    while(ir3!=line){

        ir();

    }

}

else if (ir1==line && ir2!=line && ir3!=line && ir4!=line && ir5==line ){
//10001

    forward_right();                                // prefer right

    while(ir3!=line){

        ir();

    }

}

else if (ir1==line && ir2!=line && ir3!=line && ir4==line && ir5!=line ){
//10010

    forward_left();

    while(ir3!=line){

        ir();

    }

}

```

```
else if (ir1==line && ir2!=line && ir3!=line && ir4==line && ir5==line
){ //10011

    forward_right();

    while(ir3!=line){

        ir();

    }

}

else if (ir1==line && ir2!=line && ir3==line && ir4!=line && ir5!=line ){
//10100

    forward();

}

else if (ir1==line && ir2!=line && ir3==line && ir4!=line && ir5==line
){ //10101

    forward();

}

else if (ir1==line && ir2!=line && ir3==line && ir4==line && ir5!=line
){ //10110

    forward_right();

}

else if (ir1==line && ir2!=line && ir3==line && ir4==line && ir5==line
){ //10111

    forward_right();

}

else if (ir1==line && ir2==line && ir3!=line && ir4!=line && ir5!=line ){
//11000

    forward_left();

    while(ir3!=line){
```

```
    ir();  
  }  
}  
  
else if (ir1==line && ir2==line && ir3!=line && ir4!=line && ir5==line  
) { //11001  
  forward_left();  
  while(ir3!=line){  
    ir();  
  }  
}  
  
else if (ir1==line && ir2==line && ir3!=line && ir4==line && ir5!=line  
) { //11010  
  forward_left();  
  while(ir3!=line){  
    ir();  
  }  
}  
  
else if (ir1==line && ir2==line && ir3!=line && ir4==line && ir5==line  
) { //11011  
  forward();  
}  
  
else if (ir1==line && ir2==line && ir3==line && ir4!=line && ir5!=line  
) { //11100  
  forward_left();  
}
```

```

    else if (ir1==line && ir2==line && ir3==line && ir4!=line && ir5==line
){ //11101

    forward_left();

}

    else if (ir1==line && ir2==line && ir3==line && ir4==line && ir5!=line
){ //11110

    forward_left();

}

    else if (ir1==line && ir2==line && ir3==line && ir4==line && ir5==line
){ //11111

    Stop();

}

}

void ir(){

    ir1=analogRead(out1); //most left

    ir2=analogRead(out2); //most mid left

    ir3=analogRead(out3); //most mid

    ir4=analogRead(out4); //most mid right

    ir5=analogRead(out5); //most right


    if(ir1>200)

        ir1=1;

    else

        ir1=0;

    if(ir2>200)

        ir2=1;

```

```
else
    ir2=0;
if(ir3>200)
    ir3=1;
else
    ir3=0;
if(ir4>200)
    ir4=1;
else
    ir4=0;
if(ir5>200)
    ir5=1;
else
    ir5=0;
// Print IR sensor values of Serial Monitor :-
Serial.print("IR 1 = ");
Serial.print(ir1);
Serial.print(" | IR 2 = ");
Serial.print(ir2);
Serial.print(" | IR 3 = ");
Serial.print(ir3);
Serial.print(" | IR 4 = ");
Serial.print(ir4);
Serial.print(" | IR 5 = ");
Serial.println(ir5);
```



```
}  
  
void forward()  
{  
    digitalWrite(in1, HIGH);  
    digitalWrite(in3, HIGH);  
    digitalWrite(in2, LOW);  
    digitalWrite(in4, LOW);  
}  
  
void backward()  
{  
    digitalWrite(in2, HIGH);  
    digitalWrite(in4, HIGH);  
    digitalWrite(in1, LOW);  
    digitalWrite(in3, LOW);  
  
}  
  
void left()  
{  
    digitalWrite(in1, HIGH);  
    digitalWrite(in4, HIGH);  
    digitalWrite(in2, LOW);  
    digitalWrite(in3, LOW);  
}  
  
void forward_left()  
{
```

```
digitalWrite(in1, HIGH);  
digitalWrite(in4, LOW);  
digitalWrite(in2, LOW);  
digitalWrite(in3, LOW);  
}  
void backward_left()  
{  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, HIGH);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, LOW);  
}  
void right()  
{  
    digitalWrite(in2, HIGH);  
    digitalWrite(in3, HIGH);  
    digitalWrite(in1, LOW);  
    digitalWrite(in4, LOW);  
  
}  
void forward_right()  
{  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, LOW);  
    digitalWrite(in3, HIGH);
```

```
    digitalWrite(in4, LOW);  
}  
void backward_right()  
{  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, LOW);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, HIGH);  
}  
void Stop()  
{  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, LOW);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, LOW);  
}
```

Above table shows that which direction the robot should move according to the conditions of IR sensors this simulation then made the base to code the line follower robot using 5 IR sensors and also other components

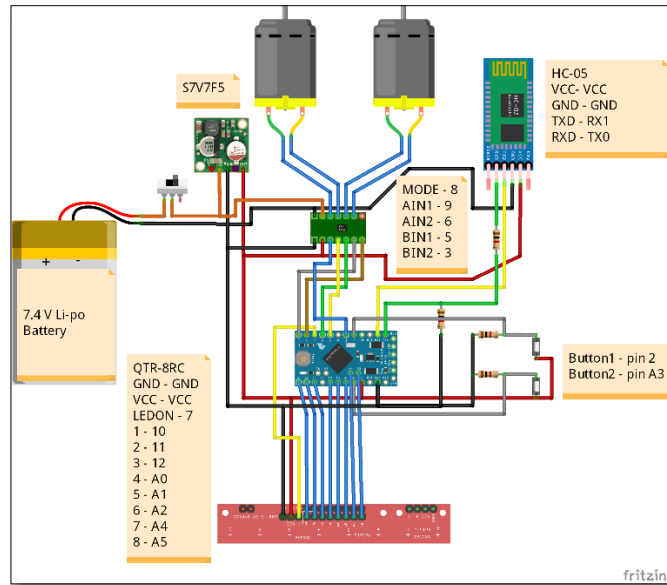
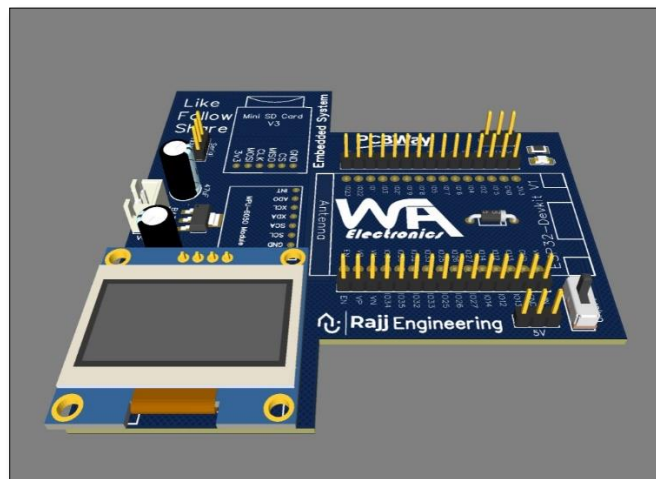
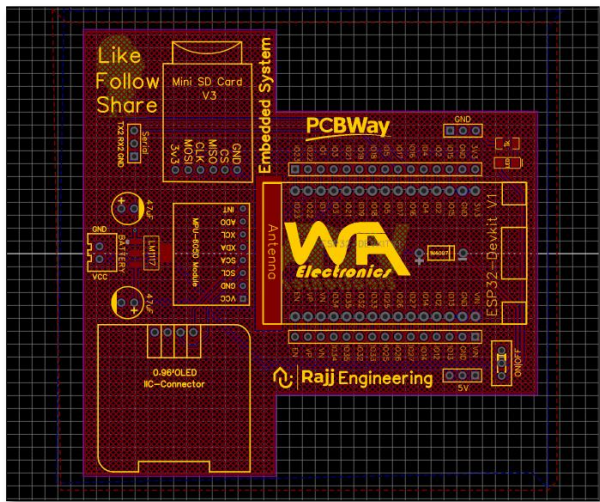
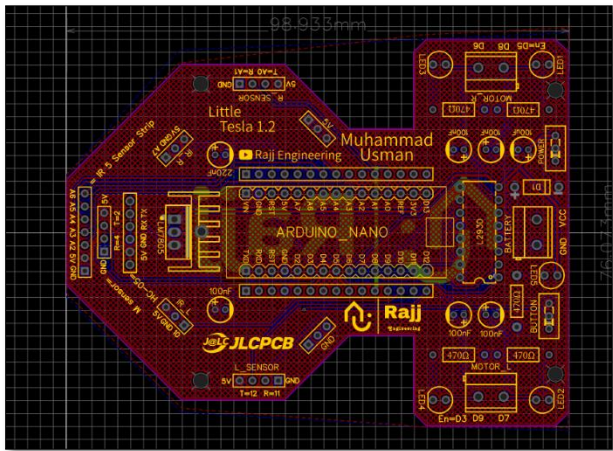
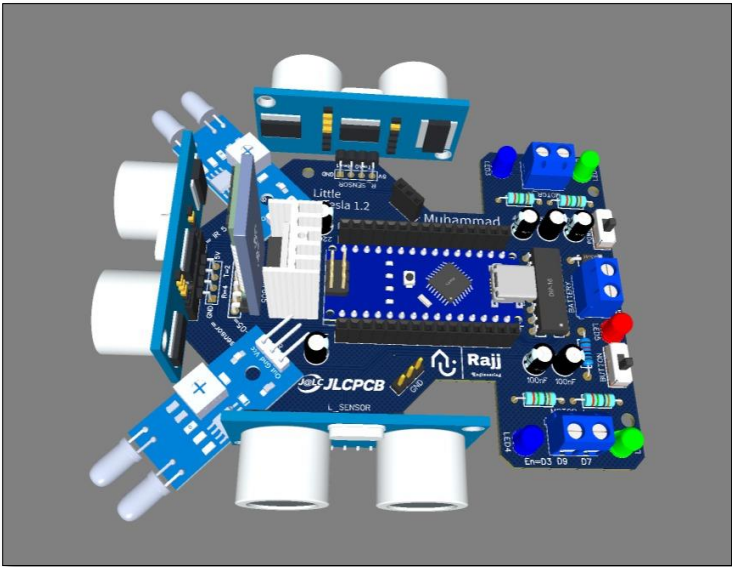


Figure 2

1.3 Specifications of deliverables

1. You must design custom PCB by yourself for this project.
2. Simulation should be done on EasyDEA and also design PCB in the same software with linked schematic.





3. Robotic Base of any suitable structure can be bought from market (Acrylic with 2 or 4 motors)
4. Dual channel H bridges can be used as modules or can be customized and designed by you
5. It should operate on battery for at least 20 Min (preferably you can use Li-Po/Li-Ion batteries with at least 2200mah capacity or you can use your own power supplies for powering the bot).
6. Micro SD card module must be used to store the information for logging.
7. IR, Ultrasonic, Color Sensors can be used to detect red, black colored Obstacle.
8. On Embedded side we must use I/O, ADC, Timers, Interrupts, PWM in REHBER
9. Brain. Robot should send data wirelessly to on a remote computer using wireless module.
10. A report describing your effort Block diagram with pins /Algorithms/State machine, Schematics, component list, Bill of Material, Limitation, future works, references (Sample will be given) should be submitted on GCR along with individual video presentation that should include the video of your project following the line at the end of the presentation.

1.4 Team Roles & Details

Name	Roll no	Role
Muhammad Usman	211280	Team lead
Muhammad Usman, Ahmed Javid	211280,211282	Hardware handling
Wadha Saddal, Amna Zafar	211268,211207	Sensor interfacing
Wadha Saddal, Amna Zafar, Muhammad Usman, Ahmed Javid	211268,211207, 211282,211280	Project assembly
Muhammad Usman, Ahmed Javid	211280, 211282	Responsible software designs

1.5 Work Break down structure

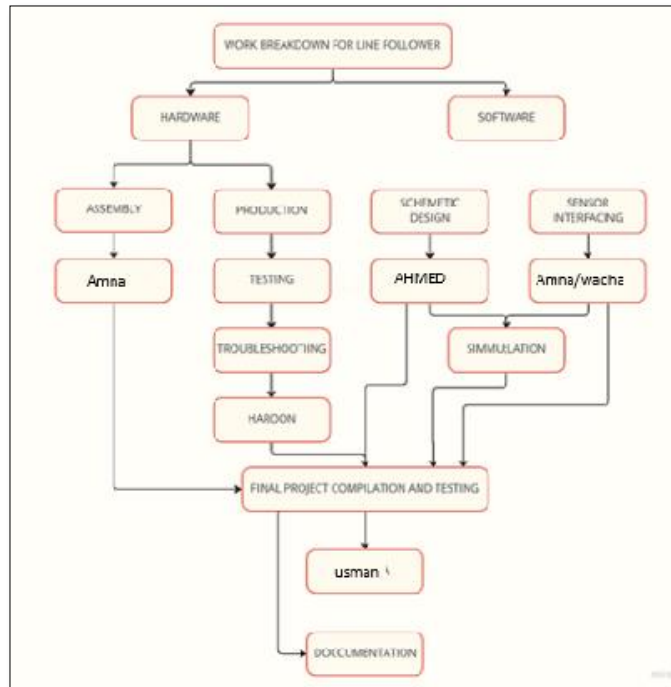


Figure 3

1.6 Gantt Chart and

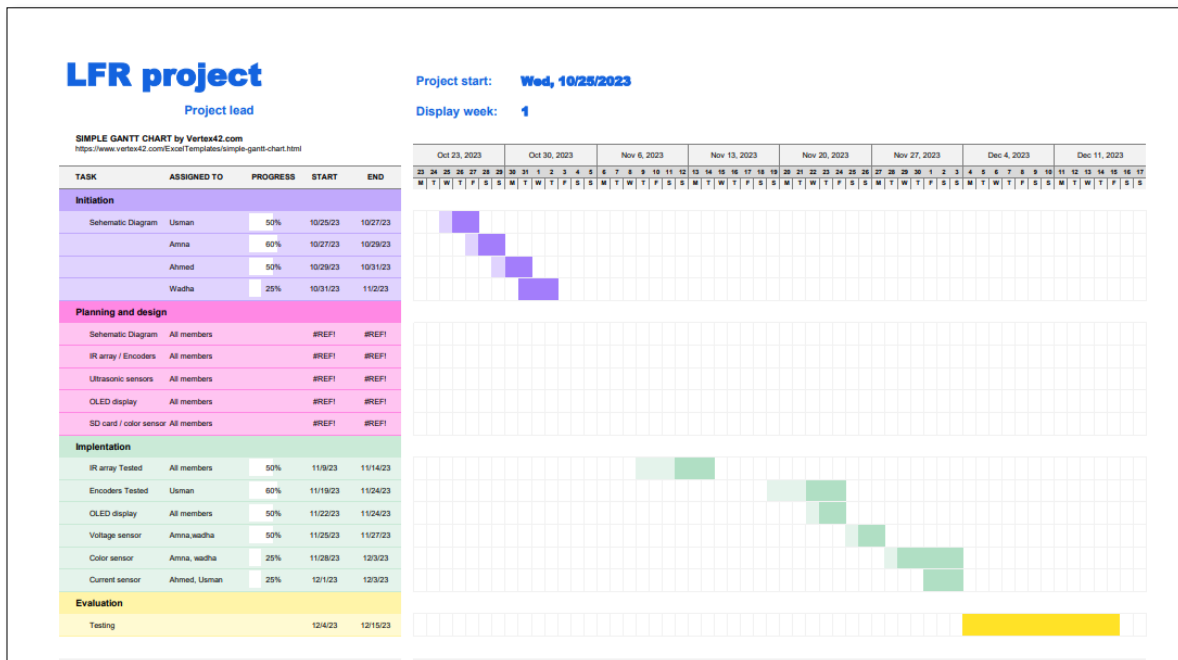


Figure 4

1.7 Estimated budgets

The estimated budget was the sum near to the actual cost that came in making the project

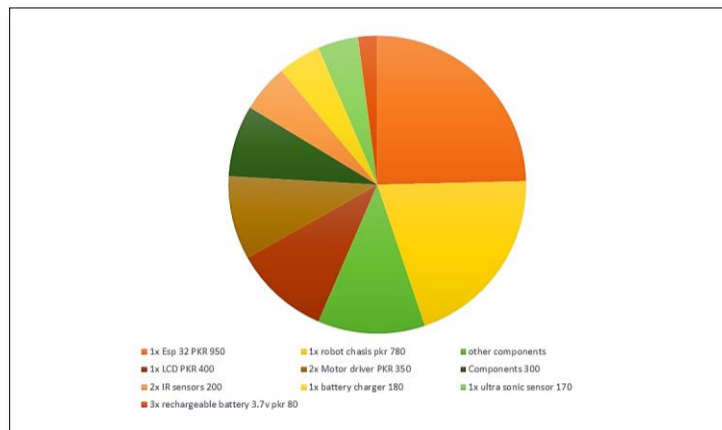


Figure 5

CHAPTER 2

PROJECT CONCEPTION

2.1 Introduction

A line follower robot is a type of autonomous robot designed to follow a predefined path or track marked with a contrasting line on the surface. These robots are widely used in various applications, including industrial automation, education, and entertainment. The primary function of a line follower robot is to detect and track a line using sensors and make necessary adjustments in its movements to stay on the path.

Key Components:

1. **Sensors:** Line follower robots are equipped with sensors that are capable of detecting the contrast between the line and the surrounding surface. The most commonly used sensors are infrared (IR) sensors, which emit and receive infrared light to identify the line.
2. **Microcontroller:** A microcontroller, such as Arduino or Raspberry Pi, serves as the brain of the line follower robot. It processes the information received from the sensors and controls the motors to maintain the robot's alignment with the line.
3. **Motors:** Motors are responsible for driving the wheels of the robot. The microcontroller adjusts the speed and direction of the motors based on the sensor inputs, allowing the robot to follow the line accurately.

Working Principle:

The line follower robot operates on a closed-loop control system. The sensors continuously monitor the surface beneath the robot. When a sensor detects the line, it sends a signal to the microcontroller. The microcontroller processes this information and adjusts the motor speeds accordingly to keep the robot aligned with the line. If a sensor detects deviation from the line, the microcontroller makes corrections to bring the robot back on track.

2.2 List of features and operational specification of our project

Features:

The project's simulation was designed on EasyEDA the project comprises of following features

- 5 IR sensors are used to detect white line with great precision.
- The project is also capable to avoid any obstacles as ultrasonic sensor is used.
- Encoders are used to determine the speed of motors
- We have used two motors for two wheels mounted
- We have used SD card and esp32 to create a data log and control the robot via Bluetooth and Wi-Fi connectivity

- Finally, the project will also be displaying outputs as OLED display panel is used for this purpose.

Operation:

The basic operations of line follower are as follows:

- Capture line position with optical sensors mounted at front end of the robot. For this a combination of IR-LED and Photodiode called an optical sensor has been used.
- This makes sensing process of high resolution and high robustness.
- Steer robot requires steering mechanism for tracking. Two motors governing wheel motion are used for achieving this task.
- This system has LCD display panel to show the distance that it covers.
- On the detecting no black surface robot move in a circular motion until line is found.

Components:

- | | |
|-------------------------|----------------------------------|
| • 1x Arduino Nano | • 1x Robot Chassis |
| • 1x ESP 32 | • 2x Motors 3V to 6V |
| • 1x OLED Display | • Jumper Wires |
| • 1x SD Card Module | • Header Pins |
| • 1x Motor Driver L283D | • Resistors Capacitors |
| • 5x IR Sensors | • On/Off Button |
| • 1x Ultrasonic Sensor | • Screws and Nuts |
| • 1x Encoder | • 3x Rechargeable Battery 3.7V |
| • 1x Voltage Sensor | Connecting We Made A 12V Battery |
| • 1x Current Sensor | |

2.3 Project Development Process

Developing a line follower robot involves a systematic process that includes planning, designing, building, testing, and refining. Here's a step-by-step project development process for creating a line follower robot:

- **Select Components:** Identify the necessary components for robot, including sensors, microcontroller, motor drivers, motors, wheels, chassis, and a power source. Commonly used sensors for line following are infrared (IR) sensors.
- **Choose a Microcontroller:** Select a suitable microcontroller for your project, such as Arduino, At mega 2560, or any other microcontroller that supports the required functionalities.
- **Design the Circuit:** Create a circuit diagram that includes the connection of sensors, motor drivers, and the microcontroller. Ensure proper voltage levels and current requirements are met.

- **Assemble the Hardware:** Build the physical structure of the robot by assembling the selected components based on your circuit design. Attach the motors to the chassis, place the sensors strategically, and connect all the components following the circuit diagram.
- **Write the Code:** Develop the program or code that will run on the microcontroller. Program the microcontroller to read data from the sensors, make decisions based on the input, and control the motors accordingly. Implement a proportional-integral-derivative (PID) control algorithm for precise line following.
- **Calibration and Testing:** Calibrate the sensors to ensure they can accurately detect the line. Test the robot on a simple track to verify that it can follow the line effectively. Make any necessary adjustments to the code or hardware based on the initial testing results.
- **Optimization:** Fine-tune the robot's performance by optimizing the code and making adjustments to sensor positions. Address any issues with speed, turning, or line detection accuracy.
- **Expand Functionality (Optional):** If desired, add extra features to your line follower robot, such as obstacle avoidance, Bluetooth control, or line color detection.
- **Documentation:** Document the project thoroughly. Include details about the components used, the circuit diagram, the code, and any modifications made during the development process. This documentation will be useful for troubleshooting and future reference.

2.4 Basic block diagrams of whole system and subcomponents

Below is the basic block diagram that shows main components of the circuit design and to give a basic understanding of how the project is planned and designed

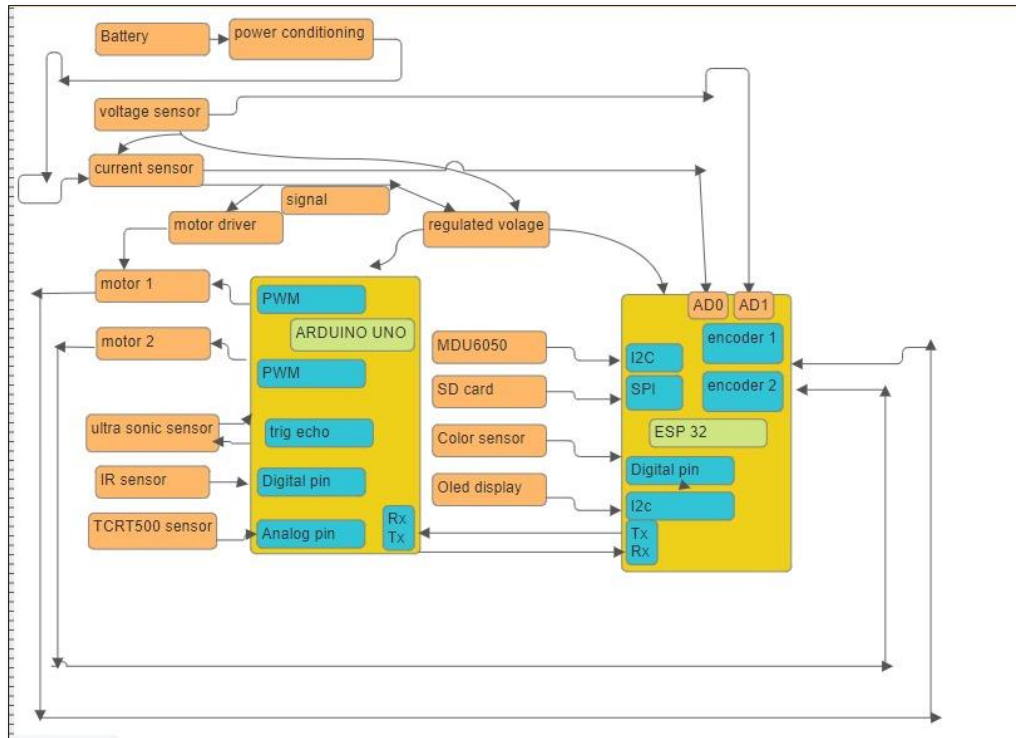


Figure 6

2.5 Literature Review

Ultrasonic sensor:

An ultrasonic sensor is a device that uses ultrasonic waves to measure the distance between the sensor and an object. It works on the principle of echolocation, similar to how bats navigate. Ultrasonic sensors are commonly used in various applications, including robotics, industrial automation, and proximity sensing. Here's an overview of how ultrasonic sensors work and their key features:

Working Principle:

1. **Emission of Ultrasonic Waves:** The ultrasonic sensor emits high-frequency sound waves (typically in the range of 40 kHz) from a transducer or a piezoelectric crystal. These sound waves travel through the air until they encounter an object in their path.
2. **Reflection and Reception:** When the sound waves hit an object, they are reflected back towards the sensor. The sensor's receiver, typically the same transducer that emitted the waves, detects the reflected waves.
3. **Time Measurement:** The sensor measures the time it takes for the emitted waves to travel to the object and back. Using the speed of sound in the air, the sensor calculates the distance to the object based on the time of flight.

4. **Distance Calculation:** The distance (D) is calculated using the formula: $D = (\text{speed of sound} * \text{time of flight}) / 2$. The division by 2 is necessary because the time measured includes the round trip (to the object and back).

Key Features:

1. **Non-Contact Measurement:** Ultrasonic sensors allow for non-contact distance measurement, making them suitable for applications where physical contact is not desirable.
2. **Range:** Ultrasonic sensors can have varying ranges, from a few centimeters to several meters, depending on the model.
3. **Accuracy:** The accuracy of ultrasonic sensors is generally good, but it can be affected by factors such as temperature, humidity, and the material properties of the objects being detected.
4. **Versatility:** Ultrasonic sensors are versatile and can be used in a wide range of applications, including object detection, proximity sensing, and liquid level measurement.
5. **Output:** The sensor typically provides an output signal that corresponds to the distance measurement. This can be an analog voltage, digital signal, or a pulse width modulation (PWM) signal.
6. **Applications:** Common applications include obstacle detection in robotics, parking assist systems in cars, liquid level measurement in tanks, and industrial automation.

Considerations:

1. **Environment:** Ultrasonic sensors may face challenges in environments with high levels of ambient noise, as they rely on the detection of reflected sound waves.
2. **Object Characteristics:** The material, size, and shape of objects can affect the sensor's ability to accurately measure distance.
3. **Multiple Sensors:** When using multiple ultrasonic sensors in close proximity, care must be taken to avoid interference between the sensors.

Ultrasonic sensors play a crucial role in enabling precise and non-contact distance measurement in a variety of applications, making them a popular choice in the field of sensors and automation.



Figure 7

Pin Configuration:

Pin Number	Pin Name	Description
1	Vcc	The Vcc pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3	Echo	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.
4	Ground	This pin is connected to the Ground of the system.

Arduino Nano:

The Arduino Nano is a compact, breadboard-friendly development board based on the Atmel ATmega328P microcontroller. It is part of the Arduino family of open-source hardware and software products designed to make electronics prototyping accessible to artists, designers, hobbyists, and anyone interested in creating interactive projects. Here are some key features and information about the Arduino Nano:

Key Features:

1. **Microcontroller:** The Arduino Nano is typically based on the ATmega328P microcontroller, which is the same microcontroller used in the Arduino Uno board.
2. **Form Factor:** The Nano has a small form factor, making it suitable for projects with space constraints. Its compact size makes it easy to integrate into various applications.
3. **Pin Configuration:** The Nano provides digital and analog input/output pins, as well as power pins, I2C, SPI, and UART communication interfaces.

4. **USB Connectivity:** It includes a built-in USB interface for programming and communication with a computer. This is usually a mini or micro-USB connector.
5. **Voltage Regulator:** The onboard voltage regulator allows the Nano to be powered from an external power source or the USB connection. It supports a wide voltage range.
6. **Clock Speed:** The ATmega328P microcontroller on the Nano typically operates at 16 MHz
7. **Flash Memory:** The microcontroller has a flash memory of 32 KB, which is used for storing the Arduino sketch (program).
8. **Ecosystem Compatibility:** Arduino Nano is compatible with the Arduino Integrated Development Environment (IDE), making it easy for users to write and upload code.
9. **Breadboard-Friendly:** The Nano has a pin layout that is compatible with standard breadboards, allowing for easy prototyping.
10. **Variants:** There are different variants of the Arduino Nano, including those with or without headers, and some with additional features like built-in Bluetooth or sensors.

Programming Arduino Nano:

1. **Arduino IDE:** To program the Arduino Nano, you can use the Arduino IDE. The IDE provides a simple and user-friendly interface for writing, compiling, and uploading code to the board.
2. **USB Connection:** Connect the Nano to your computer using a USB cable. Select the correct board and port in the Arduino IDE before uploading your code.
3. **Programming Language:** Arduino programming uses a simplified version of C/C++ with a set of predefined functions and libraries, making it accessible to beginners.

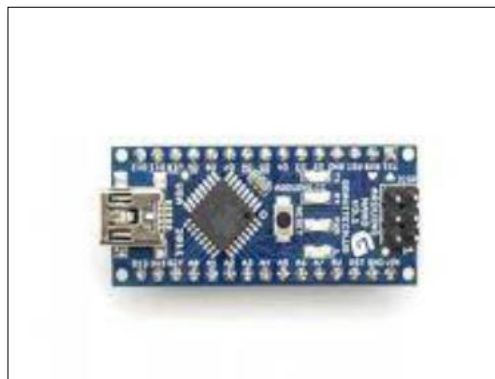


Figure 8

Pin Configuration:

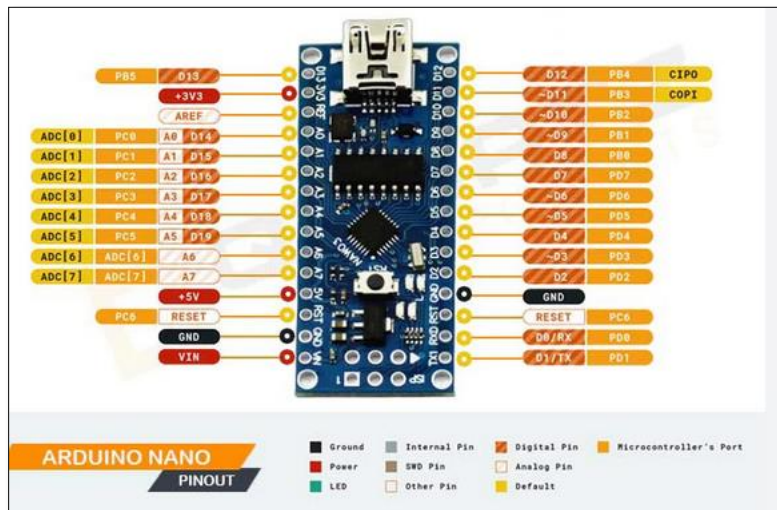


Figure 9

Robotic Chassis (2 Wheel with DC Motor):

This robotic chassis kit contains of an acrylic base with two gear motors, two compatible wheels, a ball caster, and other accessories.

Package Contains:

1. 1 x Rubber wires
2. 1 x Deceleration motors
3. 1 x Aluminum fasteners
4. 1 x Nylon all-direction wheel
5. 1 x Chassis
6. 1 x Battery box (4 x AA batteries, not included)
7. 1 x Screwdriver.



Figure 10

IR sensor:

An Infrared (IR) sensor is a device that uses infrared radiation to detect the presence of an object or measure its distance. Infrared light is beyond the visible spectrum, and these sensors are commonly used for various applications, including proximity sensing, object detection, and communication. Here's an overview of IR sensors, how they work, and their applications:

Types of IR Sensors:

1. **Infrared Proximity Sensor:** These sensors are designed to detect the presence or absence of an object within a certain range without physical contact. They are commonly used in robotics and automation for obstacle detection.
2. **Infrared Distance Sensor:** IR distance sensors, such as Sharp distance sensors, use triangulation or time-of-flight principles to measure the distance between the sensor and an object.
3. **Infrared Reflective Sensor:** These sensors consist of an IR emitter and receiver pair. They are used to detect the presence or absence of an object based on the reflection of infrared light.
4. **Infrared Thermopile Sensor:** Used for temperature measurement, these sensors detect infrared radiation emitted by an object to determine its temperature.

How IR Sensors Work:

1. **Emitter and Receiver:** IR sensors typically consist of an IR emitter and an IR receiver. The emitter emits infrared light, and the receiver detects the reflected or emitted infrared radiation.

2. **Reflection and Absorption:** In proximity and reflective sensors, the emitted infrared light reflects off an object and is received by the sensor. The presence or absence of the reflected light is used to determine the object's characteristics.
3. **Distance Measurement:** In distance sensors, the time taken for the emitted infrared light to travel to the object and back is measured. This time is used to calculate the distance based on the speed of light.
4. **Analog or Digital Output:** IR sensors can provide either analog or digital output. Analog output varies continuously based on the intensity of the reflected or emitted IR light, while digital output typically provides a binary signal indicating the presence or absence of an object.

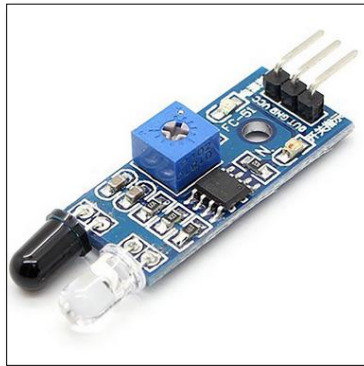


Figure 11

L293D Motor Driver

The L293D is a popular dual H-bridge motor driver integrated circuit (IC) that is widely used to control DC motors and stepper motors in various electronic projects. It allows bidirectional control of two motors or the control of one stepper motor. Here's an overview of the L293D motor driver, its features, and how it is commonly used:

Features:

1. **H-Bridge Configuration:** The L293D contains two H-bridge circuits, each capable of controlling the direction (forward or reverse) of a DC motor.
2. **Dual Motor Control:** It can control two DC motors independently, making it suitable for applications such as robot vehicles or other projects requiring multiple motor control.
3. **Stepper Motor Control:** The L293D can also be used to control a single stepper motor. It provides the necessary circuitry for interfacing with stepper motors.
4. **Voltage and Current Ratings:** The IC typically operates within a voltage range (commonly 4.5V to 36V) and can handle motor currents up to 600mA per channel (1.2A peak).

5. **Built-In Diodes:** Integrated diodes (often referred to as flyback diodes or freewheeling diodes) are present across each H-bridge to protect the IC and other components from voltage spikes generated by the motors.
6. **Enable Pins:** Each H-bridge has an enable input, allowing the user to control the motor outputs by enabling or disabling the respective H-bridge.

Pin Configuration:

The L293D IC has several pins, and the basic pinout is as follows:

- **Pin 1, 9, 16 (VCC):** Power supply for the IC.
- **Pin 4, 5, 12, 13 (GND):** Ground.
- **Pin 2, 7 (Input 1, Input 2):** Input pins for Motor 1.
- **Pin 10, 15 (Input 3, Input 4):** Input pins for Motor 2.
- **Pin 3 (Enable 1):** Enable input for Motor 1.
- **Pin 14 (Enable 2):** Enable input for Motor 2.
- **Pin 6, 11 (Output 1, Output 2):** Output pins for Motor 1.
- **Pin 8, 16 (Output 3, Output 4):** Output pins for Motor 2.

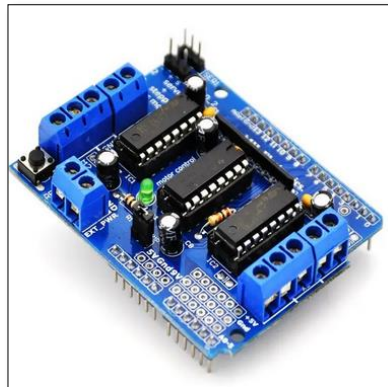


Figure 12

TT Gear Motor:

The term "TT gear motor" typically refers to a specific type of gear motor that is commonly used in robotics and hobbyist projects. The "TT" in TT gear motor stands for "Tamiya Twin," and these motors are often associated with Tamiya, a well-known manufacturer of hobbyist and educational products. Here's an overview of the key features and applications of TT gear motors:

Features:

1. **Dual Shafts:** TT gear motors are characterized by dual shafts, allowing for versatility in mounting and coupling with other mechanical components.

2. **Gearbox:** The motors are often coupled with a gearbox that provides gear reduction, increasing torque while reducing speed. The gearbox design can vary, but it typically consists of plastic gears that provide reliable and smooth operation.
3. **DC Motor:** TT gear motors are generally powered by a DC (direct current) motor, which is a common choice for robotics and small electronic projects.
4. **Voltage Range:** They are designed to operate within a specific voltage range, usually compatible with common power sources such as batteries used in hobbyist projects.
5. **Compact Size:** TT gear motors are known for their compact size, making them suitable for applications with limited space, such as small robots and vehicles.

Considerations:

1. **Voltage Compatibility:** Ensure that the voltage supplied to the TT gear motor falls within its specified operating range to prevent damage.
2. **Torque and Speed:** Consider the torque and speed requirements of your project, as different TT gear motors may have variations in these parameters.
3. **Mounting and Integration:** The dual shaft design allows for flexible mounting options, but it's essential to consider how the motor will be integrated into your specific project.



Figure 13

Lm393 IR Sensor

The LM393 IR Speed Sensor Encoder Module is a simple but effective device designed for speed measurement applications, particularly in conjunction with rotary encoders. This module combines an infrared (IR) sensor, LM393 dual comparator IC, and other components to create a sensor system capable of detecting and counting rotations, enabling speed measurement in various applications. Here's a brief note on the LM393 IR Speed Sensor Encoder Module:

Key Components:

1. **IR Sensor:** The IR sensor in this module typically consists of an IR LED and a photodetector. The reflective surface of a rotating object, equipped with alternating black and white markings, reflects the emitted infrared light back to the sensor.
2. **LM393 Dual Comparator IC:** The LM393 is a dual voltage comparator IC. It is used to process the analog signal from the IR sensor, converting it into a clean digital signal that can be easily interpreted by a microcontroller or other digital circuitry.
3. **Potentiometer:** The module often includes a potentiometer to adjust the sensitivity of the IR sensor, allowing users to optimize performance based on their specific application and environmental conditions.
4. **Indicator LEDs:** Some modules come with indicator LEDs to provide visual feedback on the module's operation. These LEDs may indicate power status or signal activity.

Working Principle:

1. **Reflective Sensing:** The IR sensor emits infrared light towards a rotating object. The surface of the object reflects the light back to the sensor, creating a varying analog signal based on the alternating black and white markings on the object.
2. **LM393 Voltage Comparison:** The LM393 compares the voltage levels of the analog signal from the IR sensor. When the voltage exceeds a certain threshold (set by the potentiometer), the LM393 produces a digital output signal, indicating the detection of a reflective surface.
3. **Pulse Generation:** As the object rotates and the IR sensor detects each reflective marking, the LM393 generates pulses. These pulses can be counted to determine the speed or rotation rate of the object.

Connection to Microcontroller:

1. Connect the power supply and ground pins of the module to the corresponding pins on a microcontroller or external power source.
2. Connect the output pin of the module to a digital input pin on the microcontroller.
3. Use the microcontroller to count pulses and calculate speed based on the time between pulses.

Considerations:

1. **Adjust Sensitivity:** Use the potentiometer to adjust the sensitivity of the IR sensor to optimize performance.
2. **Calibration:** Calibration may be necessary based on the specific reflective properties of the rotating object and the desired accuracy of speed measurement.

The LM393 IR Speed Sensor Encoder Module offers a cost-effective solution for speed measurement applications, particularly those involving rotational motion. Its simplicity and ease of integration make it suitable for a wide range of projects and applications.



Figure 14

Rotary incremental encoder

An incremental rotary encoder is a type of electromechanical device that converts the angular motion or position of a rotary shaft into analogue or digital code that represents that motion or position. It can be used for motor speed and position feedback applications that include a servo control loop and for light- to heavy-duty industrial applications.

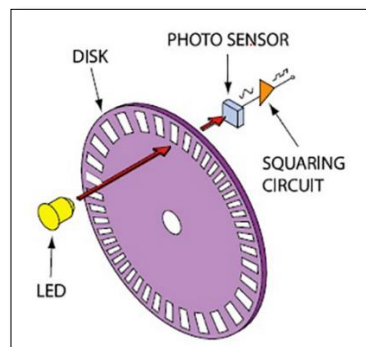


Figure 15

SD Card Module

An SD (Secure Digital) card module, often referred to as an "SD card reader module" or "SD card module," is a small electronic device that provides a convenient interface for using SD cards with microcontrollers, Arduino boards, or other embedded systems. The module simplifies the

process of reading and writing data to SD cards, making it easier to incorporate external storage into your projects. Here's an overview of the SD card module:

Key Components and Features:

1. **SD Card Slot:** The module includes a slot where you can insert a standard-sized SD card. Some modules also support microSD cards through an additional adapter.
2. **SPI Interface:** Communication with the SD card is typically done using the Serial Peripheral Interface (SPI) protocol. The module provides pins for MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock), and CS (Chip Select) to establish communication with the microcontroller.
3. **Voltage Regulator:** Some SD card modules include a voltage regulator to provide a stable voltage to the SD card, ensuring compatibility with different power supply levels.
4. **Level Shifters:** Level shifters may be integrated into the module to adapt the voltage levels between the SD card and the microcontroller, ensuring proper communication.
5. **Indicator LEDs:** Some modules have indicator LEDs to signal power and card activity, providing visual feedback on the module's status.
6. **Data Logging Capability:** SD card modules are commonly used in data logging projects where sensor data is stored on the SD card for later analysis.

Pinout:

The pin configuration may vary slightly between different SD card modules, but a typical pinout includes:

- **CS (Chip Select):** Connects to the microcontroller to enable communication with the SD card.
- **SCK (Serial Clock):** Transmits clock pulses for synchronizing data transfer.
- **MOSI (Master Out Slave In):** Carries data from the microcontroller to the SD card.
- **MISO (Master In Slave Out):** Transmits data from the SD card to the microcontroller.
- **VCC (Power):** Connects to the power supply (usually 3.3V or 5V).
- **GND (Ground):** Connects to the ground.

Usage:

1. **Power Supply:** Connect the VCC and GND pins to the power supply, ensuring proper voltage levels.
2. **SPI Communication:** Connect the SCK, MOSI, and MISO pins to the corresponding SPI pins on the microcontroller.
3. **Chip Select (CS):** Connect the CS pin to a digital pin on the microcontroller, which will be used to enable or disable communication with the SD card.

4. **Initialization:** Initialize the SD card in your code, including opening the file system and specifying file operations.
5. **Read and Write Operations:** Use the appropriate libraries and functions in your programming environment to read data from or write data to the SD card.
6. **Error Handling:** Implement error handling mechanisms to deal with potential issues such as card removal, write errors, or communication failures.

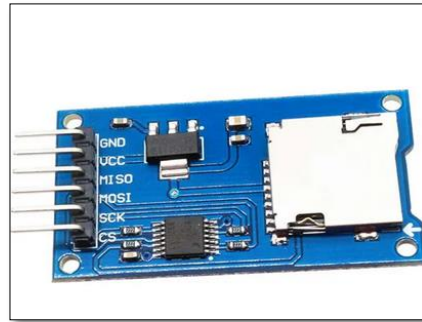


Figure 16

OLED Display:

An Organic Light-Emitting Diode (OLED) display is a type of flat-panel display technology that uses organic compounds to emit light when an electric current is applied. OLED displays offer several advantages over traditional display technologies, such as LCD (Liquid Crystal Display), including better contrast, faster response times, and more flexibility in design. Here's an overview of OLED displays:

Key Features:

1. **Organic Materials:** OLED displays use organic compounds, which emit light when an electric current is applied. These organic materials are carbon-based and emit light on their own, eliminating the need for a backlight found in LCDs.
2. **Pixel Structure:** Each pixel in an OLED display is a tiny organic light-emitting diode. This pixel structure allows for individual control of each pixel's brightness, resulting in higher contrast ratios and better color accuracy.
3. **Contrast Ratio:** OLED displays can achieve high contrast ratios because individual pixels can be turned off completely, resulting in true blacks and vibrant colors.
4. **Flexible and Thin:** OLED displays are flexible and can be manufactured on flexible substrates, allowing for curved or foldable displays. They are also thinner and lighter compared to traditional LCDs.
5. **Wide Viewing Angles:** OLED displays offer wide viewing angles with consistent color and brightness, making them suitable for various applications and viewing conditions.

6. **Response Time:** OLEDs have faster response times compared to LCDs, resulting in smoother motion in videos and reduced motion blur.
7. **Energy Efficiency:** OLED displays are energy-efficient because each pixel is individually controlled, and power is only consumed for the illuminated pixels. Black pixels consume almost no power since they are turned off.

Types of OLED Displays:

1. **Passive Matrix OLED (PMOLED):** PMOLED displays are simpler and suitable for small screens. They use a simpler control scheme but may not be as power-efficient or have the same lifespan as AMOLED.
2. **Active-Matrix OLED (AMOLED):** AMOLED displays are more complex and used in larger screens. Each pixel has its own thin-film transistor (TFT) to control the current flowing to the organic layer.

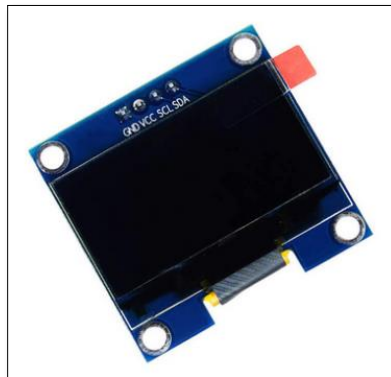


Figure 17

ACS712 Current Sensor

The **ACS712 Module** uses the famous **ACS712 IC** to **measure current** using the Hall Effect principle. The module gets its name from the IC (ACS712) used in the module, so for your final products use the IC directly instead of the module.

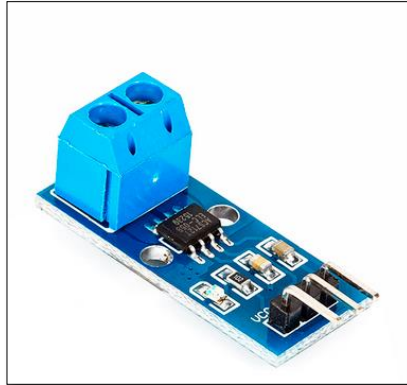


Figure 18

VOLTAGE SENSOR

Voltage Sensor is a precise low-cost sensor for measuring voltage. It is based on the principle of resistive voltage divider design. It can make the red terminal connector input voltage to 5 times smaller.



Figure 19

CHAPTER 3

MECHANICAL DESIGN

Mechanical design plays very important rule in any project of engineering. Our line following robot is an engineering project that can be affected by a lame or bad model of mechanical design. We can say that the mechanical design is responsible for the sustainability, weight lifting, long lasting life of the robot. So, designing the robot in a most sufficient way so that external condition or internal condition on that material used and that design are minimum.

3.1. Mechanism selection and Platform Design

The robot chassis is simple and easy to setup that lets us create a mobile robotics platform in short time. They are a perfect solution when time or equipment's do not allow fabrication of your own robotics chassis. The robot chassis usually have lots of pre-drilled holes and slot that allow other parts like sensors to be quickly attached.

So, we used robot chassis as our platform and then placed a PCB on it containing our all circuitry



Figure 21

3.2 Actuators with speciation and datasheet

We used Gear motors as actuators whose specs are as follows

Size	21 x 14.7 cm approx.
Voltage	3-6 V
Gear Motor Reduction Ratio	148

Wheel Size	6.6 x 2.6 cm approx.
Tire Center Hole	5.3mm Long, 3.5mm wide
No Load Speed (6V)	200RPM +-10%
No Load Current (6V)	Less Than 200mA
No Load Speed (3V)	90RPM +-10%
No Load Current (3V)	Less Than 150mA

Chapter Number 4

SOFTWARE/FIRMWARE DESIGN

4.1 Controller Selections with features

Arduino Nano:

The Arduino Nano is a compact and versatile microcontroller board based on the ATmega328P microcontroller, which is also used in the Arduino Uno. The Nano is designed for easy integration into various projects, especially those with space constraints.

Specifications:

1. Microcontroller:

ATmega328P

2. Operating Voltage:

5V

3. Input Voltage (recommended):

7-12V

4. Digital I/O Pins:

14 (of which 6 provide PWM output)

5. Analog Input Pins:

8

6. DC Current per I/O Pin:

40mA

7. Flash Memory:

32KB (ATmega328P)

8. SRAM:

2KB

9. Clock Speed:

16MHz

10. Communication:

USB, I2C, SPI, UART

Pinout Configuration:

The Arduino Nano has a compact form factor and comes with a variety of digital and analog I/O pins. Here is a simplified pinout configuration:

1. **Digital Pins (D2-D13):** General-purpose digital I/O pins.
2. **Analog Pins (A0-A7):** Analog input pins.
3. **PWM Pins (D3, D5, D6, D9, D10, D11):** Pins that support PWM output.
4. **Serial Pins (RX, TX):** UART communication pins.
5. **I2C Pins (A4, A5):** Pins for I2C communication.
6. **SPI Pins (D10, D11, D12, D13):** Pins for SPI communication.
7. **Power Pins (5V, GND):** Power supply pins.

ESP32:

The ESP32 is a powerful microcontroller and system-on-chip (SoC) that integrates Wi-Fi and Bluetooth capabilities. It is part of the ESP8266 and ESP32 family of chips developed by Espressif Systems. The ESP32 is widely used in various applications, including IoT (Internet of Things) devices, wireless communication, and embedded systems.

Specifications:

1. Microcontroller:

ESP32 dual-core Ten silica LX6 processor

2. Operating Voltage:

3.3V

3. Digital I/O Pins:

36 (of which 34 can be used for GPIO)

4. Analog Input Pins:

16

5. **DC Current per I/O Pin:**

12mA

6. **Flash Memory:**

4MB

7. **SRAM:**

520KB

8. **Clock Speed:**

160MHz

9. **Communication:**

Wi-Fi, Bluetooth, UART, SPI, I2C, I2S

Pinout Configuration:

The ESP32 has a more extensive pinout compared to the Arduino Nano due to its additional features. Here's a simplified representation:

1. **Digital Pins (D0-D23):** General-purpose digital I/O pins.
2. **Analog Pins (A0-A15):** Analog input pins.
3. **PWM Pins (D1-D19):** Pins that support PWM output.
4. **Serial Pins (RX0, TX0, RX1, TX1):** UART communication pins.
5. **I2C Pins (SDA, SCL):** Pins for I2C communication.
6. **SPI Pins (SCK, MOSI, MISO, SS):** Pins for SPI communication.
7. **I2S Pins (I2S, LRCK, BCLK, DOUT, DIN):** Pins for I2S audio communication.
8. **Power Pins (3.3V, 5V, GND):** Power supply pins.
9. **USB Pins (D-/D+, GND):** USB communication pins.

4.2 Software Design details & user Requirements

Components

Components that we used in the software design

- Ultrasonic sensor
- SD card module
- Servo motor
- IR sensors
- L293D motor driver
- OLED Display
- Motors
- Current sensor.
- Voltage sensor
- Potentiometer

Inputs

- Left Sensor (LS)
- Right Sensor (RS)
- Right Centre Sensor (RC)
- Left Centre Sensor (LS)
- Centre Sensor (CS)
- Ultrasonic Sensor
- Voltage sensor
- Current sensor
- Rotary encoder (Tachometer)

Outputs

- Left Motor
- Right Motor
- SD card
- 16 X 4 LCD

4.3 State Machine & System flow diagram

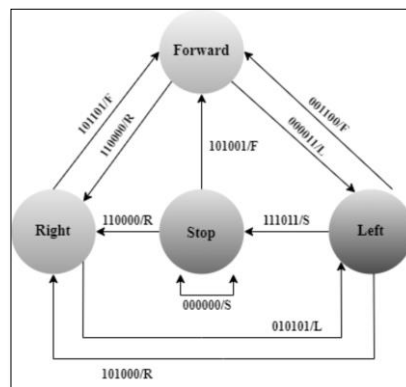
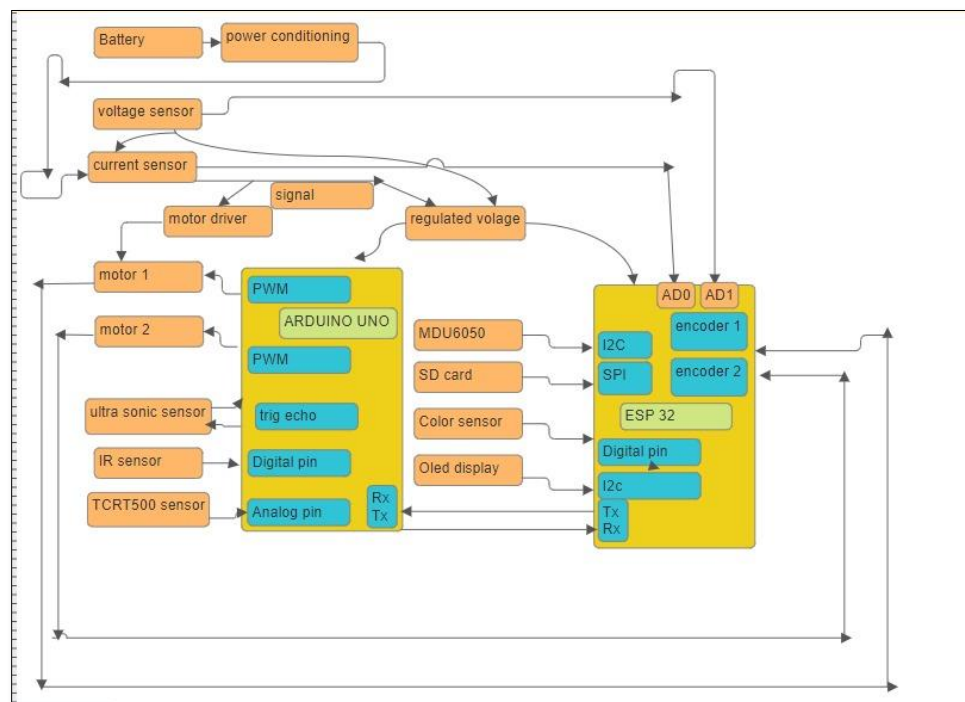


Figure 24

Current state	S1(LS)	S2(LS1)	S3(CS)	S4(CS1)	S5(RS1)	S6(RS)	Next state	Output
stop	0	0	0	0	0	0	stop	S
stop	1	0	1	0	0	1	Forward	F
Forward	1	1	0	0	0	0	Right	R
Right	0	1	0	1	0	1	Left	L
Left	1	1	1	0	1	1	Stop	S
Stop	1	1	0	0	0	0	Right	R
Right	1	0	1	1	0	1	forward	F
Forward	0	0	0	0	1	1	Left	L
Left	1	0	1	0	0	0	Right	R
Right	1	1	1	1	1	1	Stop	S

4.4 Detailed block diagram



CHAPTER 5

SIMULATIONS AND FINAL INTEGRATIONS

5.1 Integrations and testing all hardware and software component separately

We firstly read all the data sheets of our components and noted down the optimal voltage and current for each of our components. Then tested all the components with DMM separately. Most of the components were working properly. The components which were not in working condition were replaced. Then after checking the components, we integrated them together. While integration we took a good care that no components should be short or not plugged-in wrong pin as it could lead to burning of that component.

Compiled Arduino code:

```
#define irleft 10
#define irright A7
#define ltrig 12
#define lecho 11
#define rtrig A0
#define recho A1
#define mtrig 2
#define mecho 4
#define in1 8
#define in2 6
#define in3 9
#define in4 7
#define enA 5
#define enB 3
#define out5 A6
#define out4 A5
#define out3 A4
#define out2 A3
#define out1 A2
```

```

int ir1,ir2,ir3,ir4,ir5;    // Variable to store data from IR sensors

int black=0;                // Black line means Ouput of sensor = 0
int white=1;                // White line means Ouput of sensor = 1
int line=white;             // Which Line you want to Follow

int SPEED=140;              // Set Speed B/W 0-255

void linefollowing();

void ir();
void left();
void forward_left();
void backward_left();
void right();
void forward_right();
void backward_right();
void forward();
void backward();
void Stop();

void setup() {
    pinMode(out1,INPUT);      // Ouput from IR sensor
    pinMode(out2,INPUT);      // Ouput from IR sensor
    pinMode(out3,INPUT);      // Ouput from IR sensor
    pinMode(out4,INPUT);      // Ouput from IR sensor
    pinMode(out5,INPUT);      // Ouput from IR sensor

    pinMode(in1, OUTPUT);     // CounterClockwise Motor A

```

```

pinMode(in2, OUTPUT);      // Clockwise Motor A
pinMode(in3, OUTPUT);      // CounterClockwise Motor B
pinMode(in4, OUTPUT);      // Clockwise Motor B
pinMode(enA, OUTPUT);      // Enable Ouputs for Motor A (PWM)
pinMode(enB, OUTPUT);      // Enable Ouputs for Motor B (PWM)
Serial.begin(9600);        // initialize serial monitor
analogWrite(enA,SPEED);
analogWrite(enB,SPEED);
delay(10);
}
void loop() {

ir();
linefollowing();
}
void linefollowing(){
if (ir1!=line && ir2!=line && ir3!=line && ir4!=line && ir5!=line ){
//00000
    Stop();
}
else if (ir1!=line && ir2!=line && ir3!=line && ir4!=line && ir5==line ){
//00001
    forward_right();
    while(ir3!=line){
        ir();
    }
}
else if (ir1!=line && ir2!=line && ir3!=line && ir4==line && ir5!=line ){
//00010
    forward_right();

```

```

}

else if (ir1!=line && ir2!=line && ir3!=line && ir4==line && ir5==line ){
//00011

    forward_right();

    while(ir3!=line){

        ir();

    }

}

else if (ir1!=line && ir2!=line && ir3==line && ir4!=line && ir5!=line ){
//00100

    forward();

}

else if (ir1!=line && ir2!=line && ir3==line && ir4!=line && ir5==line ){
//00101

    forward();

}

else if (ir1!=line && ir2!=line && ir3==line && ir4==line && ir5==line ){
//00111

    forward_right();

}

else if (ir1!=line && ir2==line && ir3!=line && ir4!=line && ir5!=line ){
//01000

    forward_left();

    while(ir3!=line){

        ir();

    }

}

else if (ir1!=line && ir2==line && ir3!=line && ir4!=line && ir5==line ){
//01001

    forward_left();

    while(ir3!=line){

```

```
    ir();
}
}
else if (ir1!=line && ir2==line && ir3!=line && ir4==line && ir5!=line ){
//01010
    forward();
}
else if (ir1!=line && ir2==line && ir3!=line && ir4==line && ir5==line ){
//01011
    forward_right();
    while(ir3!=line){
        ir();
    }
}
else if (ir1!=line && ir2==line && ir3==line && ir4!=line && ir5!=line ){
//01100
    forward_left();
}
else if (ir1!=line && ir2==line && ir3==line && ir4!=line && ir5==line ){
//01101
    forward_left();
}
else if (ir1!=line && ir2==line && ir3==line && ir4==line && ir5!=line ){
//01110
    forward();
}
else if (ir1!=line && ir2==line && ir3==line && ir4==line && ir5==line ){
//01111
    forward_right();
}
```

```

else if (ir1==line && ir2!=line && ir3!=line && ir4!=line && ir5!=line ){
//10000

    forward_left();

    while(ir3!=line){

        ir();

    }

}

else if (ir1==line && ir2!=line && ir3!=line && ir4!=line && ir5==line ){
//10001

    forward_right();                                // prefer right

    while(ir3!=line){

        ir();

    }

}

else if (ir1==line && ir2!=line && ir3!=line && ir4==line && ir5!=line ){
//10010

    forward_left();

    while(ir3!=line){

        ir();

    }

}

else if (ir1==line && ir2!=line && ir3!=line && ir4==line && ir5==line ){
//10011

    forward_right();

    while(ir3!=line){

        ir();

    }

}

else if (ir1==line && ir2!=line && ir3==line && ir4!=line && ir5!=line ){
//10100

```

```

    forward();
}
else if (ir1==line && ir2!=line && ir3==line && ir4!=line && ir5==line ){
//10101
    forward();
}
else if (ir1==line && ir2!=line && ir3==line && ir4==line && ir5!=line ){
//10110
    forward_right();
}
else if (ir1==line && ir2!=line && ir3==line && ir4==line && ir5==line ){
//10111
    forward_right();
}
else if (ir1==line && ir2==line && ir3!=line && ir4!=line && ir5!=line ){
//11000
    forward_left();
    while(ir3!=line){
        ir();
    }
}
else if (ir1==line && ir2==line && ir3!=line && ir4!=line && ir5==line ){
//11001
    forward_left();
    while(ir3!=line){
        ir();
    }
}
else if (ir1==line && ir2==line && ir3!=line && ir4==line && ir5!=line ){
//11010
    forward_left();

```



```

    while(ir3!=line){
        ir();
    }
}

else if (ir1==line && ir2==line && ir3!=line && ir4==line && ir5==line ){
//11011
    forward();
}

else if (ir1==line && ir2==line && ir3==line && ir4!=line && ir5!=line ){
//11100
    forward_left();
}

else if (ir1==line && ir2==line && ir3==line && ir4!=line && ir5==line ){
//11101
    forward_left();
}

else if (ir1==line && ir2==line && ir3==line && ir4==line && ir5!=line ){
//11110
    forward_left();
}

else if (ir1==line && ir2==line && ir3==line && ir4==line && ir5==line ){
//11111
    Stop();
}
}

void ir(){
    ir1=analogRead(out1); //most left
    ir2=analogRead(out2); //most mid left
    ir3=analogRead(out3); //most mid
    ir4=analogRead(out4); //most mid right

```

```
ir5=analogRead(out5); //most right

if(ir1>200)
    ir1=1;
else
    ir1=0;
if(ir2>200)
    ir2=1;
else
    ir2=0;
if(ir3>200)
    ir3=1;
else
    ir3=0;
if(ir4>200)
    ir4=1;
else
    ir4=0;
if(ir5>200)
    ir5=1;
else
    ir5=0;
// Print IR sensor values of Serial Monitor :-
Serial.print("IR 1 = ");
Serial.print(ir1);
Serial.print(" | IR 2 = ");
Serial.print(ir2);
Serial.print(" | IR 3 = ");
Serial.print(ir3);
```

```
Serial.print(" | IR 4 = ");
Serial.print(ir4);
Serial.print(" | IR 5 = ");
Serial.println(ir5);
}
void forward()
{
    digitalWrite(in1, HIGH);
    digitalWrite(in3, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in4, LOW);
}
void backward()
{
    digitalWrite(in2, HIGH);
    digitalWrite(in4, HIGH);
    digitalWrite(in1, LOW);
    digitalWrite(in3, LOW);

}
void left()
{
    digitalWrite(in1, HIGH);
    digitalWrite(in4, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
}
void forward_left()
{

```

```
digitalWrite(in1, HIGH);
digitalWrite(in4, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
}
void backward_left()
{
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
}
void right()
{
digitalWrite(in2, HIGH);
digitalWrite(in3, HIGH);
digitalWrite(in1, LOW);
digitalWrite(in4, LOW);

}
void forward_right()
{
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, HIGH);
digitalWrite(in4, LOW);
}
void backward_right()
{
```

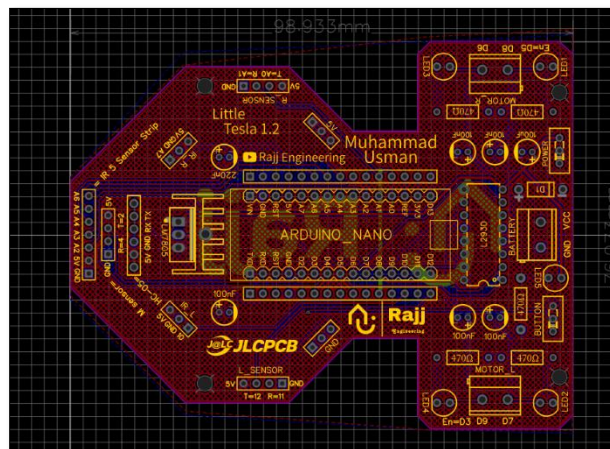
```

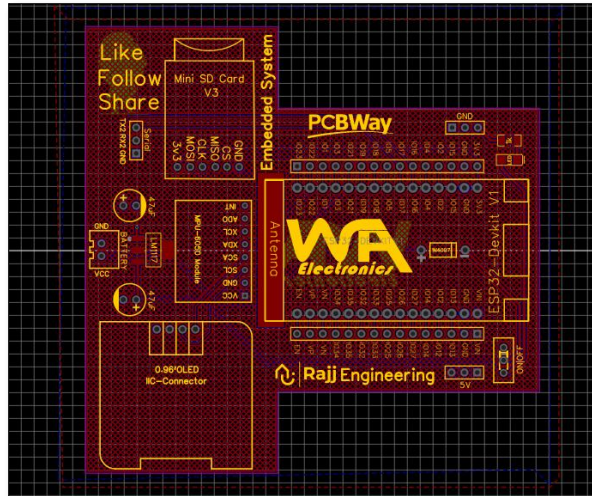
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
}
void Stop()
{
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
digitalWrite(in3, LOW);
digitalWrite(in4, LOW);
}

```

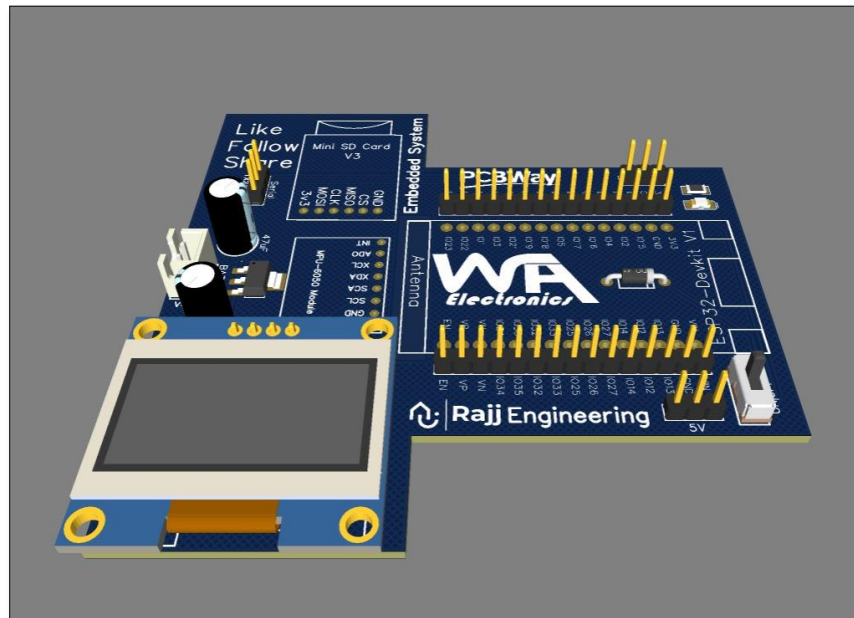
5.2 Simulation PCB and 3D view

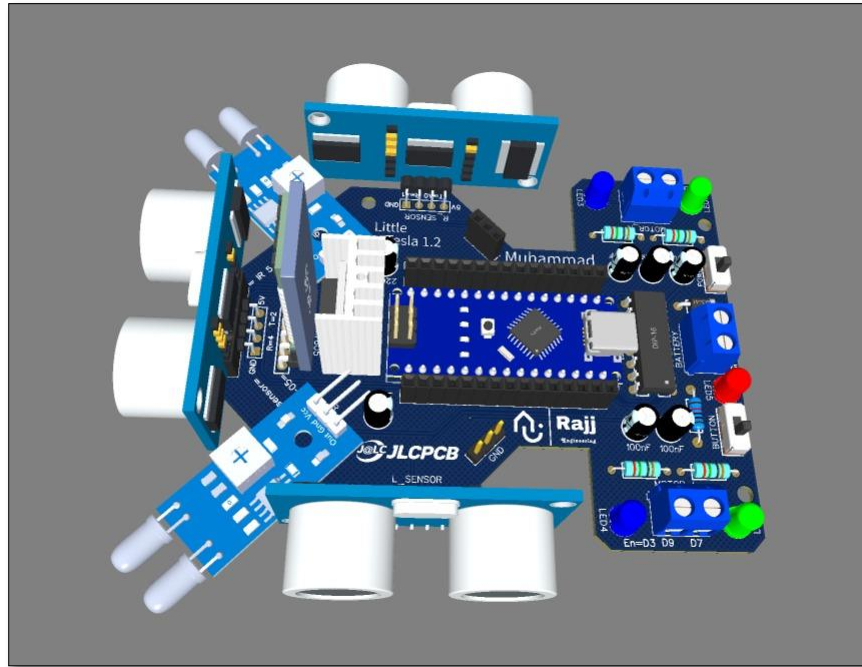
PCB design





3D View:





CHAPTER 6

SYSTEM TEST PHASE

6.1 Final testing

First stage

We first checked our components separately. Some of them were not working properly then we replaced them i.e. the motor driver we purchased first was burnt so we had to replace it. then we integrated all the components

Second stage

In our second stage we tested all the modules by placing them on bread board. All our modules were working properly. Our robot was following the line and, on our OLED, display all the values were displaying properly. I.e., current value voltage value and encoder value etc. we also checked our teacher the proper working.

Third stage

The third stage was testing on pcb board. We integrated the components on pcb while all working properly separately. After soldering we checked all the ports and pins of pcb board they were also fine all our sensors were giving the value properly. OLED was working fine also the other sensors. But our voltage was not being mapped on our motors by our motor driver while voltage was properly given to motor driver.

6.3 Project actual Pictures

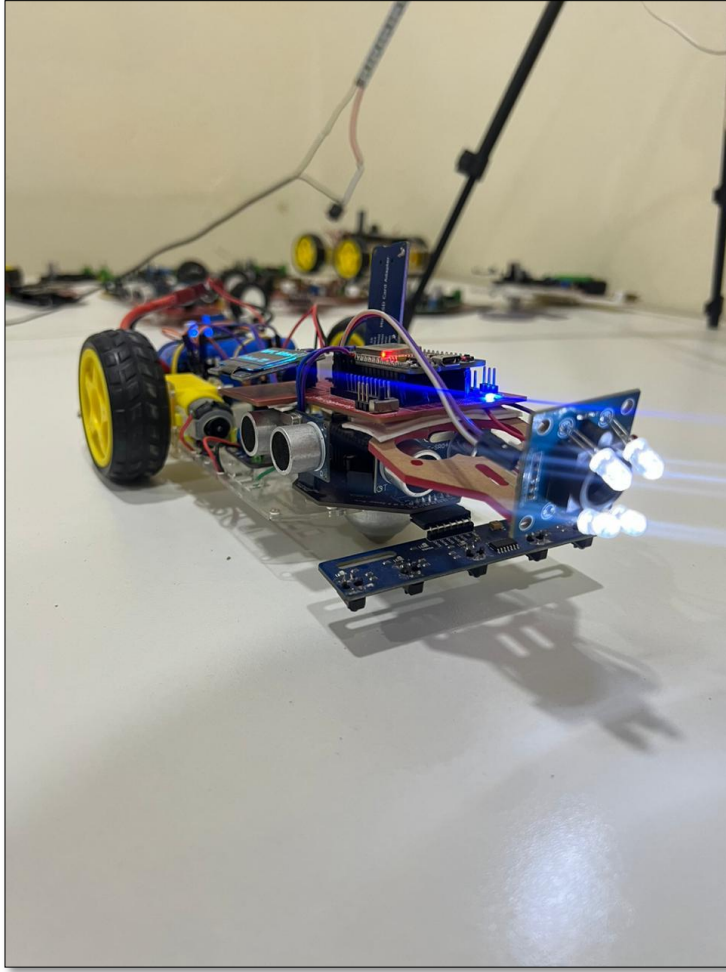


Figure 29

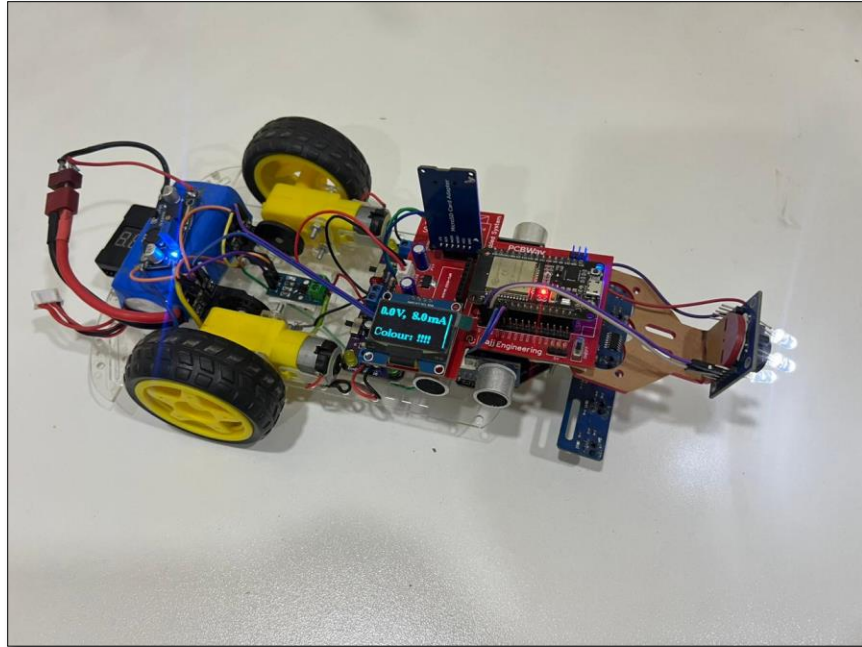


Figure 30

CHAPTER 7

PROJECT MANAGEMENT

7.1 Everyone must write one page about how he executed his role in project

Team lead:

Muhammad Usman

A team lead is a person who is responsible for leading a group of people to achieve a common goal. In a line follower project, I was responsible for the following tasks:

- Leading and managing the project team, including setting team goals and objectives, and monitoring performance.
- Assigning tasks and responsibilities to team members and ensuring that they are completed on time.
- Communicating with stakeholders and clients, and making sure that their needs are met.
- Coordinating the work of different team members to ensure that all aspects of the project are moving forward smoothly.
- Resolving any issues that arise within the team, and making decisions when necessary.
- Managing the resources of the project, such as budget, equipment, and tools.
- Providing guidance and mentorship to team members to help them develop their skills and grow professionally.
- Facilitating meetings and team-building activities to ensure that everyone is on the same page and working well together.
- Representing the team to management, and providing regular updates on the project's progress.

Hardware handling

Ahmed Javid:

We designed our PCB in proteus and then etched it on a copper board after that the drilling part of the process took place so we could connect our components. Then we soldered the components onto the PCB. Then we did the short testing to find out if the circuit was short. Then we did a test run of our project to see whether the PCB was working or not.

In a line follower project, the hardware handling role would involve designing, assembling, and maintaining the hardware components of the robot. This would include tasks such as:

- Designing the mechanical and electrical components of the robot, including the chassis, wheels, and any other mechanical parts.
- Assembling the robot using various hardware components, such as motors, microcontroller, sensors, and power supply.
- Programming and configuring the microcontroller and other electronic components.
- Testing and troubleshooting the robot's hardware, making adjustments as necessary.
- Ensuring the robot is properly maintained, such as keeping the robot clean, lubricating moving parts, and making any necessary repairs.
- Collaborating with other team members to integrate the hardware components into the overall control system of the robot.
- Continuously monitoring the robot's hardware performance and making any necessary adjustments to improve the robot performance.

Sensor interfacing

Amna Zafar and Wadha Saddal

In a line follower project, the sensor interfacing role would involve designing and implementing the interface between the sensors used in the robot and the control system.

I was responsible for tasks such as:

- Selecting appropriate sensors for the project, such as infrared, ultrasonic, or optical sensors.
- Designing the electrical interface between the sensors and the control system.
- Programming the control system to properly interpret and use sensor data.
- Testing and calibrating the sensors to ensure accurate and reliable data is being collected.
- Troubleshoot and debug any issues with the sensor interface.
- Continuously monitor the sensor data and make any necessary adjustments to improve the robot performance.
- Collaborating with other team members to integrate the sensor data into the overall control system of the robot.

Project assembly

Ahmed Javid, Muhammad Usman, Wadha Saddal and Amna Zafar

In a line follower project, the project assembly role would involve the physical assembly of the robot, ensuring that all the components are put together correctly and work as intended. This would include tasks such as:

- Assembling the robot according to the design specifications, using hardware components such as motors, microcontroller, sensors, and power supply.
- Testing the robot's functionality after assembly to ensure that it is working correctly.

- Troubleshooting and making repairs as necessary to fix any issues that arise during assembly.
- Collaborating with other team members to ensure that the robot is assembled in a timely and efficient manner.
- Documenting the assembly process and creating assembly instructions for future reference.
- Continuously monitoring the robot's performance and making any necessary adjustments to improve the robot performance.

Responsible software designs

Muhammad Usman

In a line follower project, the software design role would involve designing and implementing the software that controls the robot's behavior and functionality. This would include tasks such as:

- Designing the software architecture and creating flowcharts and diagrams to represent the software's structure and behavior.
- Writing code in the programming language chosen for the project, such as C, C++, or Python, to implement the robot's functionality and control system.
- Testing and debugging the software to ensure that it is working correctly and efficiently.
- Collaborating with other team members, such as the sensor interfacing and hardware handling roles, to ensure that the software is integrated properly with the robot's hardware.
- Continuously monitoring the software performance and making any necessary adjustments to improve the robot's performance.
- Documenting the software design, code, and any changes made to it during the development process.
-

7.2 Comment individually about success /failure of your project

Muhmmad Usman	The project was a success
Ahmed Javid	The project was a success
Amna Zafar	The project was a success
Wadha Saddal	The project was a success

7.3 Risk management that you learned

It is the practice of identifying, evaluating, and preventing project risks that have the potential to impact the desired outcomes. Typically, project managers are in charge of overseeing the risk management process throughout the life of a project.

Several risks were encountered during this project, including PCB etching, which if not done correctly required us to purchase a new one from a store, which was inconvenient. Another is the drilling and soldering of PCB holes.

Because our PCB was double-layered, PCB alignment was a major challenge for us. Because many components were not available in proteus, we created them ourselves.

Typically, project managers are in charge of overseeing the risk management process throughout the life of a project.

Several risks were encountered during this project, including PCB etching, which if not done correctly required us to purchase a new one from a store, which was inconvenient. Another is the drilling and soldering of PCB holes. In Order to avoid issues when using acids, we took care of all the precautionary measures.

Because our PCB was double-layered, PCB alignment was a major challenge for us. Because many components were not available in proteus, we created them ourselves.

CHAPTER 8

FEEDBACK FOR PROJECT AND COURSE

Our assignment tested our prior knowledge and abilities while being quite enlightening. Additionally, it greatly aided us in thinking widely and carefully about every alternative. We picked up new abilities like soldering that we had only studied or saw on displays. Through this project, in addition to learning technical skills, we also learned how to manage time and money. Individually, every one of us gained knowledge on how to collaborate and function as a team. Overall, we had a positive experience.

