Semester Project Report

Group Members:

(211234) – Malik Maaz Khan

(211240) – Ahmad Hassan

(211241) – Danish Abdullah

(211291) – Muhammad Salman

Microcontroller & Embedded System Project Report

DEPARTMENT OF MECHATRONICS ENGINEERING

SEMESTER: (BEMTS A_B) Fall – 2023

Date : 18th – December 2023

INDEX

7.2 Comment individually about success /failure of your project

7.3 Final Bill of material list and paragraph about project budget allocation

7.4 Give a word count how many words each member write in final report in their allocated color as assigned

7.5 Risk management that you learned

8 Feedback for project and course

9 Detailed YouTube video explaining our work of each member

Appendix-

References-

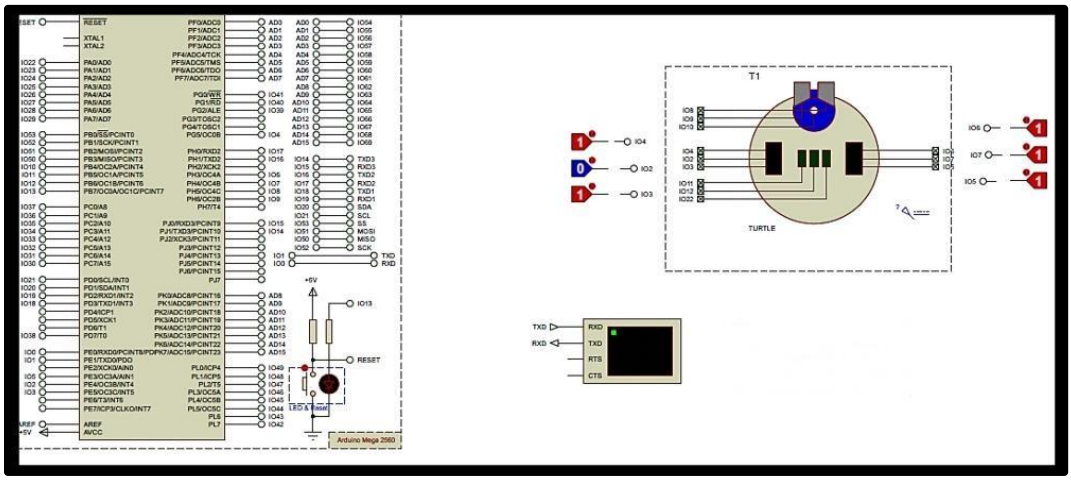
1.1 Proposal

The main objective of this project was to design a line follower according to the instructions provided. Line follower robot basically follows a provided line until that 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 three IR sensors used in our project, could be an array of 5. These sensors read the line and send that reading to Arduino and then control the robot movement.

1.2 Initial feasibility

The initial phase was divided in to two parts:

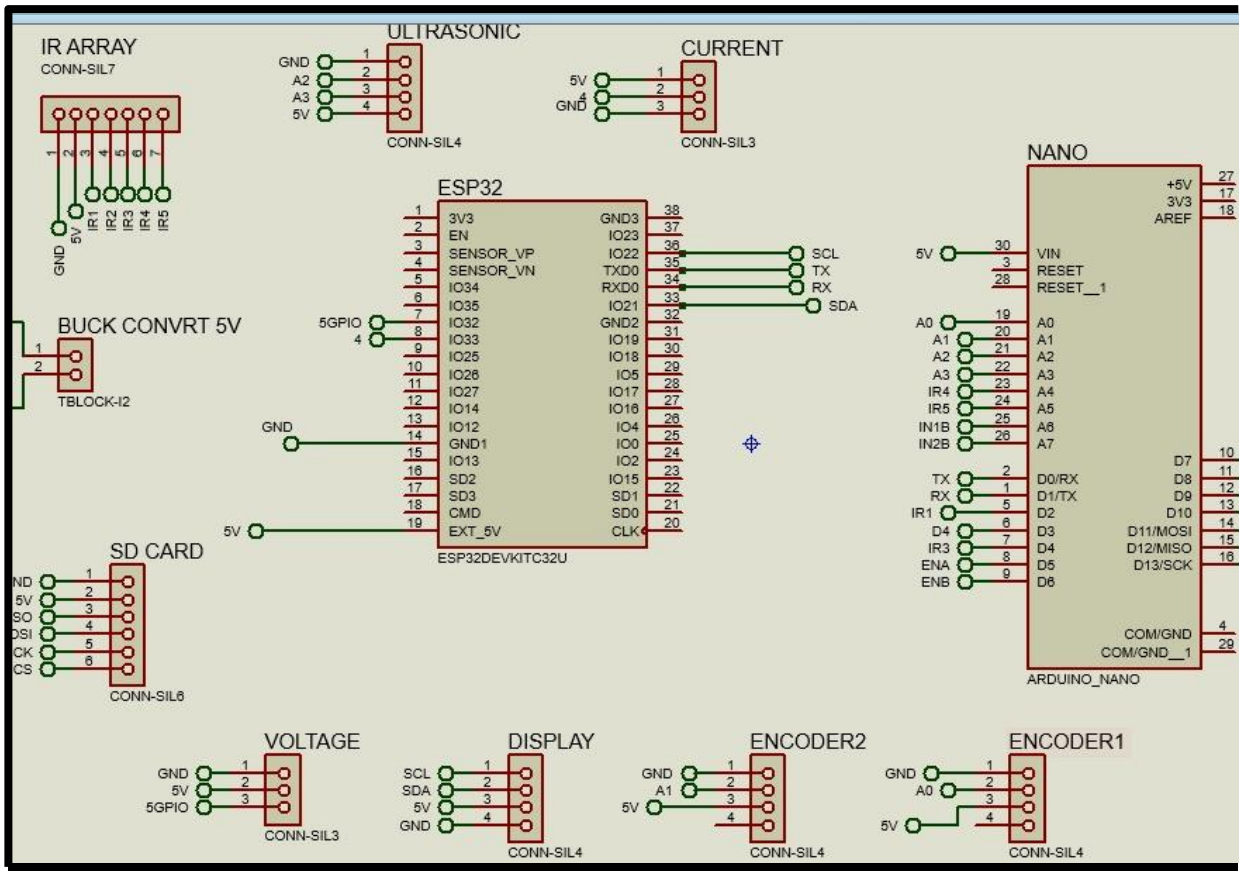
First was to understood the turtle robot simulation and programming it.



Secondly , To design a basic line follower using (3 IR sensors), (2 motors), (LCD) and (L3298 motor driver). This stimulation was aim to understand the core working of the line follower.

Above all describes the basic direction, that the robot should move according.

Specifications of deliverables



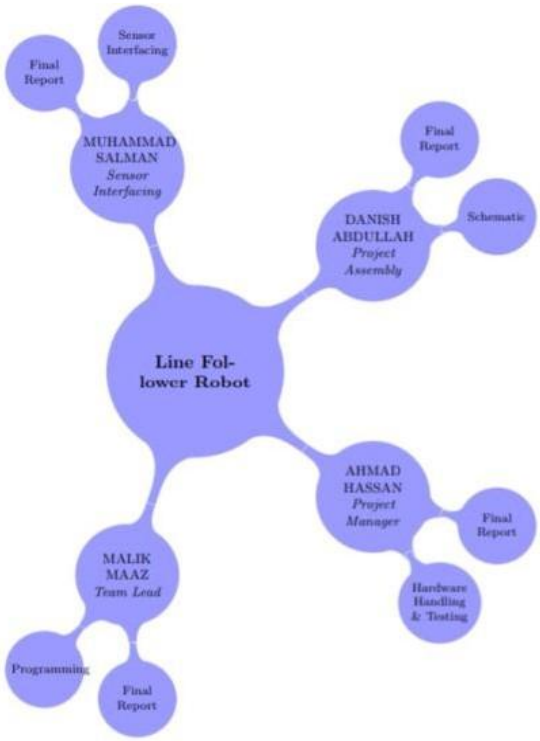
1. You must design custom ATMEL controller-based board by yourself for this project.
2. Simulation should be done on Proteus and you can design PCB in any software.
3. Dual channel H bridges can be used as modules or can be customized and designed by you
4. It should operate on battery for at least 20 Min (preferably you can use Li-Po/Li-Ion batteries with at least 2000mAh capacity or you can use your own power supplies for powering the bot).
5. Micro SD card module must be used to store the information for logging.
6. IR, Ultrasonic, Color Sensors can be used to detect red colored Obstacle.

7. On Embedded side we must use I/O, ADC, Timers, Interrupts, and PWM.
8. Robot should send data wirelessly to on a remote computer using wireless module.
9. A report describing your effort Block diagram with pins /Algorithms/State machine, Schematics, component list, Bill of Material, Limitation, future works, references should be submitted.

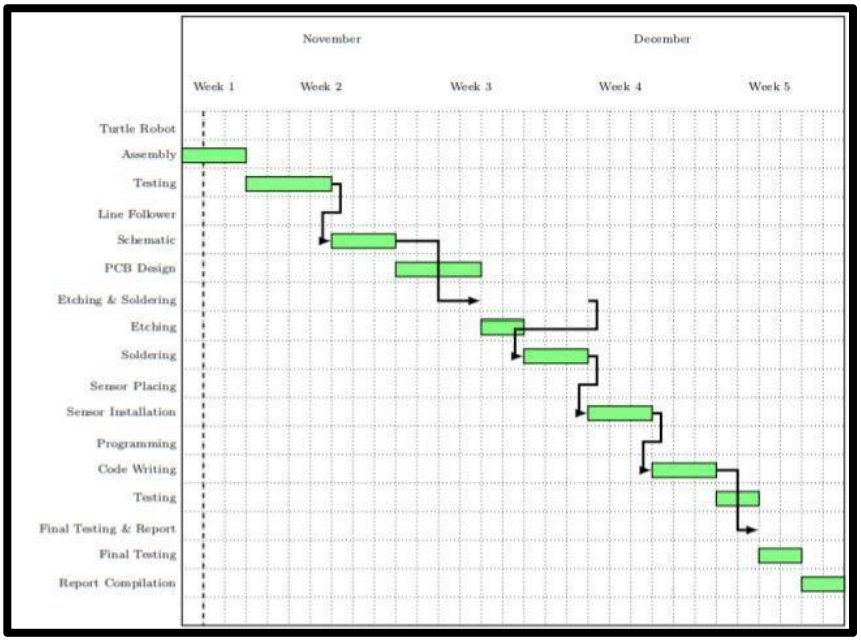
1.3 Team Roles & Details

Team Member Name	Role	Word Count
MALIK MAAZ	Team Lead and programming	
AHMAD HASSAN	Hardware Handling and Testing	
DANISH ABDULLAH	Project Assembly and Schematic	
MUHAMMAD SALMAN	Sensor interfacing and programming.	

1.4 Work Break down structure

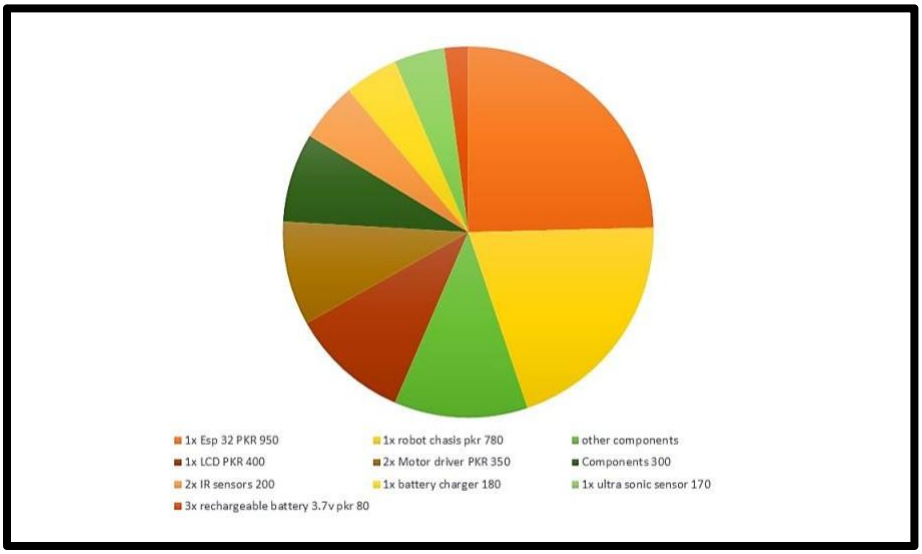


1.5 Gantt Chart and



1.6 Estimated budgets

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



2.1 Introduction

Line follower is a machine that can follow a path. The path can be visible like a black line on a white surface. Sensing a line and maneuvering the robot to stay on course, while constantly correcting wrong moves using feedback from the sensor forms a simple yet effective system. It can be used in automobile, industrial automations, guidance, etc.

Line follower robot is autonomous that means it automatically follows a line which is pre-defined. Generally, it follows a black line on a white surface or a white line on a black surface. Some of the basic operation of a line follower is given below:

· Reading the pre-defined line by IR sensor array which is installed on the front-down side of the robot and sends those readings to the Arduino. The AT Mega microcontroller which is built in on Arduino analyzes those readings and do the particular operations.

The steering mechanism is simple in this robot. Three wheels are used, two wheels are on the back part connected with the motors and one independent wheel on the front-middle part of the robot. · On Straight line, the speed is fast and on a turn, speed is relatively slow depending on turn angel. Good motor quality and good sensing quality will increase the robot movement performance.

2.3 List of features and operational specification of our project

Features:

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

- 3 IR sensors are used to detect white line with great precision thus the angles on path such as a 90 turn covered.
- 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.
- Finally, the LCD panel helps to display the input Voltage and current and also the output voltage.

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 make 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:

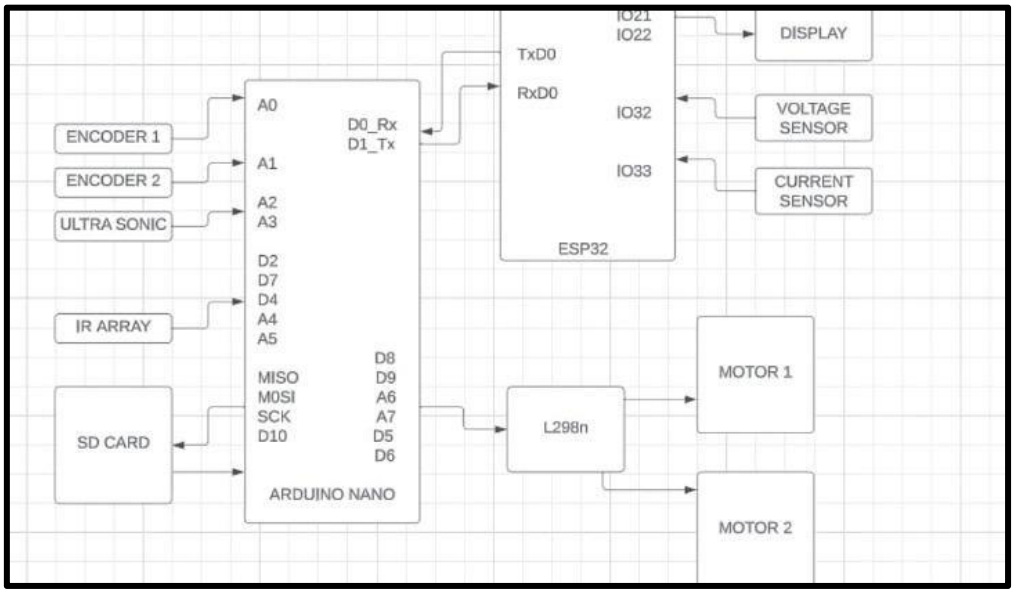
- Arduino NANO
- 1x ESP 32
- 12C Module for LCD interfacing
- 16x2 LCD Module
- 1x SD Card Module
- 3x IR Sensors
- 1x Ultrasonic Sensor
- 1x Encoder
- 1x Voltage Sensor
- 1x Current Sensor
- 1x Robot Chassis
- 2x Motors 3V to 6V
- Jumper Wires
- Header Pins
- Resistors Capacitors
- On/Off Button
- Screws and Nuts
- 1x Motor Driver LM298
- 3x Rechargeable Battery 3.7V Connecting We Made A 12V Battery

2.4 Project Development Process

- After the detail literature survey through the books, periodical, journal, magazine, websites. The idea of the project is well defined.
- The logic is derived for the intelligence of the robot. It is programmed and burn it to the Arduino by using the software Arduino]
- The accuracy and viability of the program and electronic components is tested in the simulation software Proteus
- After the successful simulation result it is implemented in the hardware.
- For the hardware we have designed our Pcb board to keep the robot more professional and fictional
- After the finishing the programming, electrical and electronics part, the stable, reliable and flexible mechanical design and fabrication is completed.
- Finally, system is tested and encountered error is omitted.

2.5 Basic block diagrams of whole system and subcomponents

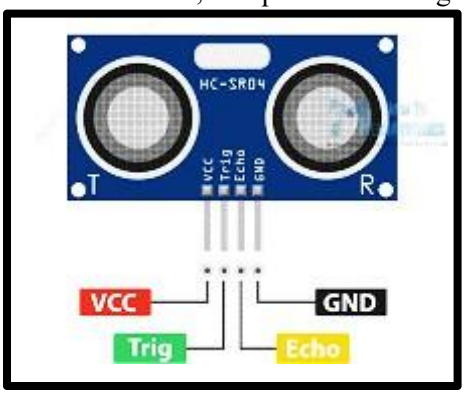
Below is the basic block diagram that shows main components of the circuit design and to give a basic understanding of how the project is planned and designed



2.6 Literature Review

Ultrasonic sensor:

Ultrasonic transducers and ultrasonic sensors are devices that generate or sense ultrasound energy. An ultrasonic sensor sends a high pulse (signal) and then a low pulse (signal) in a continuous manner. Once these signals hit an obstacle, the signals reflect back and are received by ultrasonic sensor. The time taken by the signals to return is used to calculate the distance between the sensor and the obstacle. The closer the obstacle is to the sensor, the quicker these signals return.



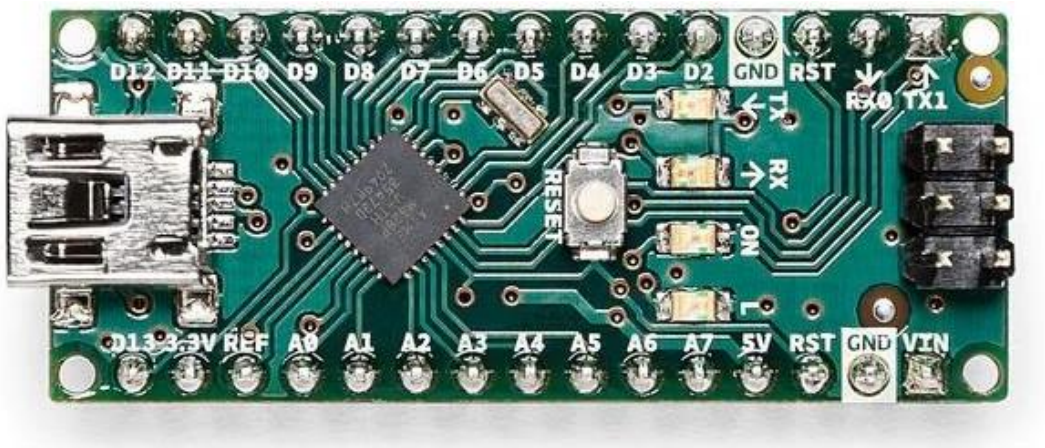
Pin Configuration:

Pin Number	Pin Name	Description
1	Vcc	The Vcc pin powers the sensor, typically with +5V

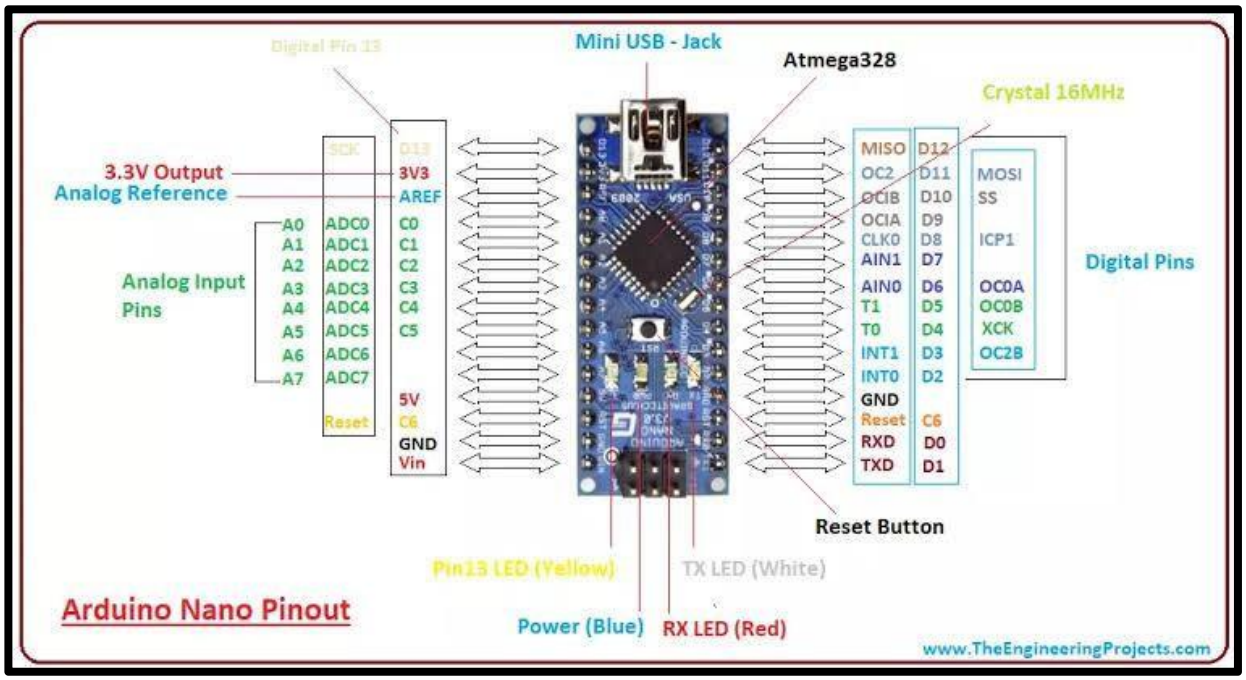
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	Ground	This pin is connected to the Ground of the system.

Arduino Nano:

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. There are total 14 digital Pins and 8 Analog pins on your Nano board. The digital pins can be used to interface sensors by using them as input pins or drive loads by using them as output pins. A simple function like pinMode() and digitalWrite() can be used to control their operation. The operating voltage is 0V and 5V for digital pins. The analog pins can measure analog voltage from 0V to 5V using any of the 8 Analog pins using a simple function like analogRead().



Pin Configuration:



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.

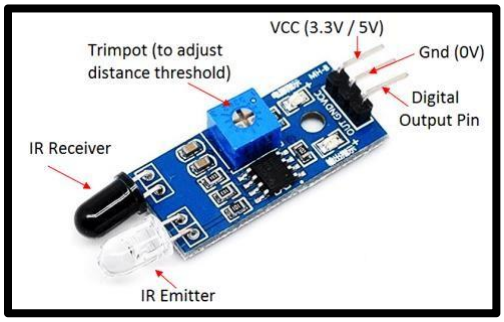
Package Contains:

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.

IR sensor:

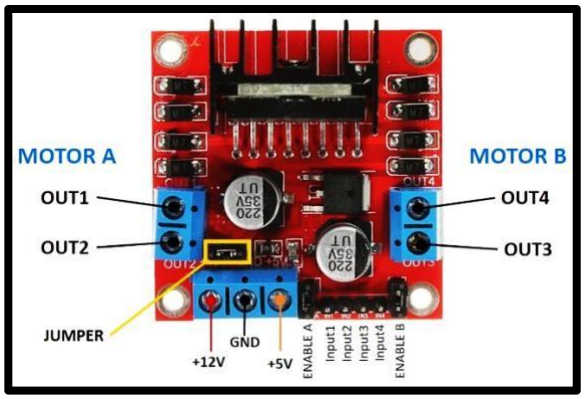
Typically, in an IR sensor module, an IR transmitter or IR Led sends an infrared signal in certain frequency compatible with an IR receiver. IR sensors are typically used for obstacle detection or for distance measurement.

The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode which is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, The resistances and these output voltages, change in proportion to the magnitude of the IR light received.



L298N Motor Driver

This **L298N Motor Driver Module** is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. **L298N Module** can control up to 4 DC motors, or 2 DC motors with directional and speed control.

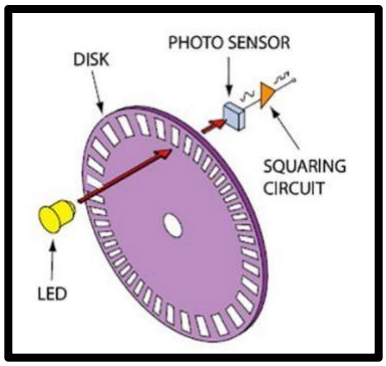


TT Gear Motor:

The controller i.e., L298N is basically used to control this DC motor. Its speed and direction are controlled by the l298N. D.C. motors are built to operate in the steady state in a speed range close to their no-load speed this speed is generally too high for applications like the robot we are making so that is why we merge the DC motor with gearbox to reduce its speed. The basic working principle of the DC motor is that whenever a current carrying conductor places in the magnetic field, it experiences a mechanical force. The principle remains the same but using them both as a single component is very helpful in performing variety of functions.

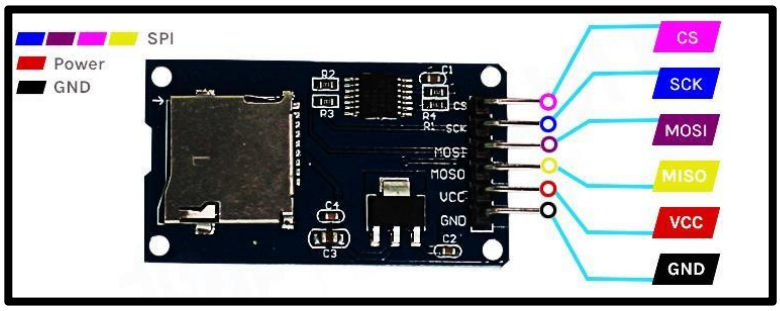
Rotary incremental encoder

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.



SD Card Module

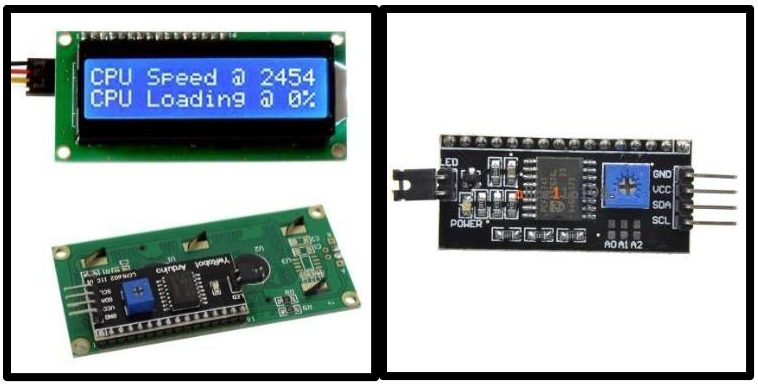
SD cards or Micro SD cards are widely used in various applications, such as data logging, data visualization, and many more. Micro SD Card Adapter modules make it easier for us to access these SD cards with ease. The Micro SD Card Adapter module is an easy-to-use module with an SPI interface and an on-board 3.3V voltage regulator to provide proper supply to the SD card.



16 x 2 LCD Display with I2C module

16X4 CHARACTER LCD 1604 GREEN LCD DISPLAY is a dot-matrix liquid crystal display module specially used for displaying letters, numbers, symbols, etc. Divided into 4-bit and 8-bit data transmission methods. 1604 Green Character LCD provides rich command settings: clear display; cursor return to origin; display on/o; cursor on/o; display character ashes; cursor shift; display shift, etc. It can be used in any embedded systems, industrial device, security ,medical and hand-held equipment.

Due to limited pin resources in a microcontroller/microprocessor, controlling an LCD panel could be tedious. Serial to Parallel adapters such as the I2C serial interface adapter module with PCF8574 chip makes the work easy with just two pins. The serial interface adapter can be connected to a 16x2 LCD and provides two signal output pins (SDA and SCL) which can be used to communicate with an MCU/MPU.



LCD Pin Configuration:

Pin No.	Symbol	Description
1	V_{SS}	Ground
2	V_{DD}	Power supply for logic
3	V_O	Contrast Adjustment
4	RS	Data/ Instruction select signal
5	R/W	Read/Write select signal
6	E	Enable signal
7~14	DB0~DB7	Data bus line
15	A	Power supply for B/L +
16	K	Power supply for B/L -

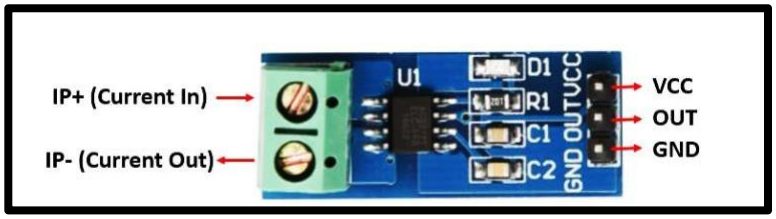
I2C ModulePin Configuration:

Pin Name	Pin Type	Pin Description
GND	Power	Ground
VCC	Power	Voltage Input

SDA	I2C Data	Serial Data
SCL	I2C Clock	Serial Clock
A0	Jumper	I2C Address Selection 1
A1	Jumper	I2C Address Selection 2
A2	Jumper	I2C Address Selection 3
Backlight	Jumper	Control Backlight of panel

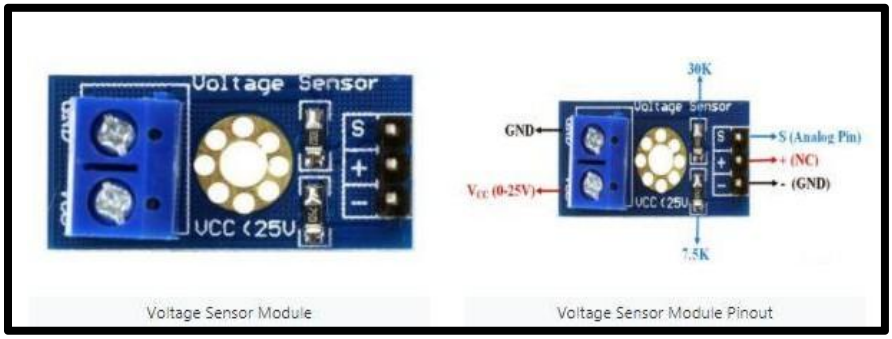
ACS712 Current Sensor

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 you final products use the IC directly instead of the module.



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.



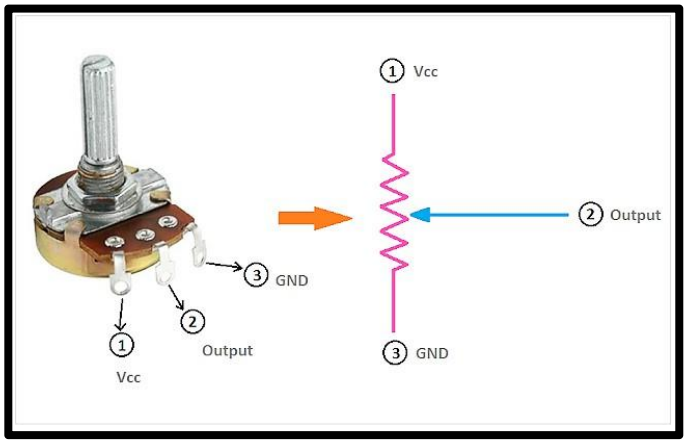
Voltage Sensor Module | Voltage Sensor Module Pinout

Potentiometer

A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

The measuring instrument called a potentiometer is essentially a voltage divider used for measuring electric potential (voltage); the component is an implementation of the same principle, hence its name.

Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment. Potentiometers operated by a mechanism can be used as position transducers, for example, in a joystick. Potentiometers are rarely used to directly control significant power (more than a watt), since the power dissipated in the potentiometer would be comparable to the power in the controlled load.



As most of us were new for this course so we needed a background check and some experienced person work. This was necessary so that we could understand what's really needed to be done. So for that purpose we went through some websites which were quite useful. \

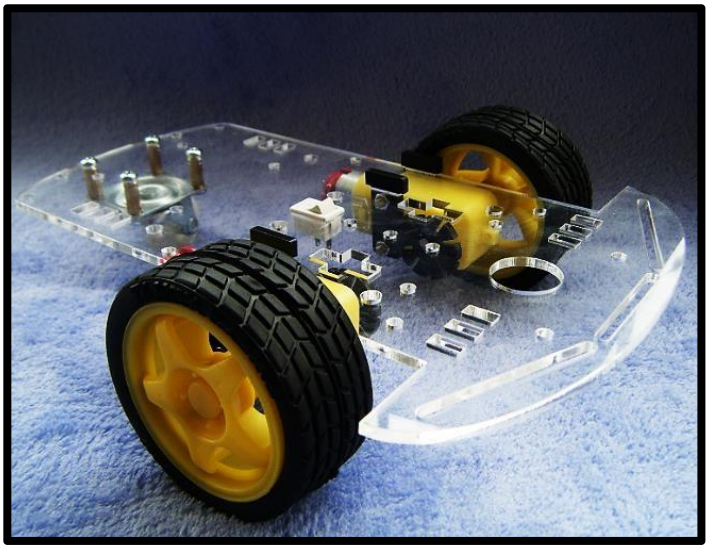
GitHub - MertArduino/Arduino-Line-Following-Robot
How to make and programme the fastest line following robot - Quora

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



3.2 Actuators with speciation and datasheet

We used Gear motors as actuators whose specs are as follows

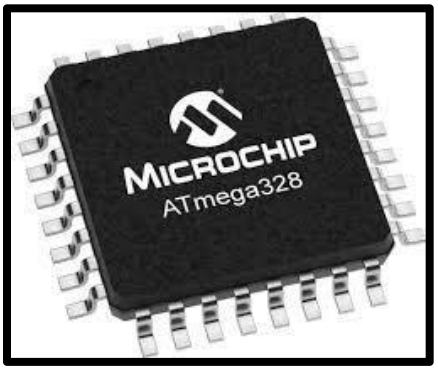
Table 6

Size	21 x 14.7 cm approx
Voltage	3-6 V

Gear Motor Reduction Radio	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

4.1 Controller Selections with features

The ATmega328P microcontroller is a key component powering the Arduino Nano, a popular compact development board in the Arduino ecosystem. Featuring 32KB of Flash memory, 2KB of SRAM, and 1KB of EEPROM, the ATmega328P operates at 16MHz clock speed, offering a range of digital and analog input/output pins, timers, serial communication interfaces, and other essential peripherals for building various embedded systems and DIY projects. Its versatility and ease of use make it a cornerstone in the maker community, providing a foundation for countless innovative applications in robotics, IoT, automation, and beyond.



Parametrics:

Name	Value
Program Memory Type	Flash
Program Memory Size (KB)	32KB
CPU Speed (MIPS/DMIPS)	16MHz
SRAM (KB)	2KB
Data EEPROM/HEF (bytes)	1KB (1024 Byte)
Digital Communication Peripherals	UART, SPI, I2C
Capture/Compare/PWM Peripherals	6PWM
Timers	3 Timers
Number of Comparators	1

Temperature Range (°C)	-40 to 85
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	22

Advantages:

- **Compact Size**: Its small form factor makes it suitable for projects with space constraints or where portability is essential.
- **Cost-effective**: Arduino Nano boards are relatively affordable, enabling cost-efficient prototyping and project development.
- **Versatile I/O**: Despite its size, the Nano offers a good number of digital and analog pins, allowing connectivity to various sensors, actuators, and peripherals.
- **Compatibility:** It's compatible with a wide range of shields and sensors designed for Arduino boards, enhancing its adaptability to various project needs**.**

Disadvantages:

- **Limited Processing Power**: The ATmega328P processor on the Arduino Nano might be limiting for complex computations or advanced projects requiring higher processing capabilities.
- **Limited Memory**: The onboard memory (Flash, SRAM, EEPROM) might not suffice for extensive data storage or handling large codebases.
- **Fewer Peripherals**: Compared to larger Arduino boards, the Nano has a limited number of onboard peripherals, which might restrict its applications in certain projects requiring more specialized features.

ATmega328P belongs in an umbrella of microcontrollers; ATmega8/16/32/128/2560. It does share common configurations such as the EEPROM and RAM but still consists of differences as shown below:

Device	Flash	General Purpose I/O pins	16-bit resolution PWM channels	Serial USARTs	ADC Channels
ATmega8	8KB	23	3	1	6
ATmega16	16KB	32	4	1	8
ATmega32	32KB	32	4	1	8
ATmega128	128KB	53	4	2	8
ATmega2560	256KB	86	15	4	16

4.2 Software Design details & user Requirements

Components

Components that we used it the software design

- Ultrasonic sensor
- SD card module
- Servo motor
- IR sensors
- L298 motor driver
- Tachometer
- 16x2 LCD display
- Motors
- Current sensor.
- Voltage sensor
- Potentiometer

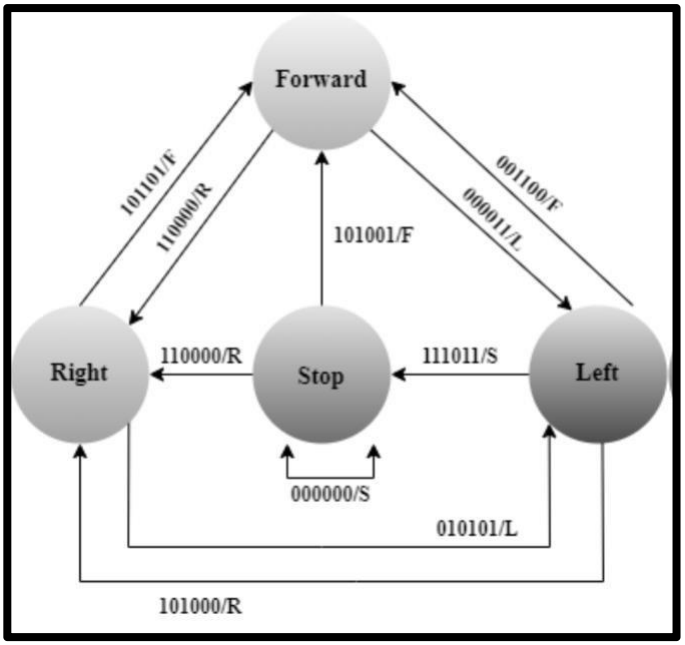
Inputs

- 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 2 LCD

4.3 State Machine & System flow diagram



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

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 burring of that component.

Compiled Arduino code:

Nano:

```
#include <SPI.h>
#include <SD.h>

// Motor A connections
int in1 = 8;
int in2 = 9;
// Motor B connections
int in3 = 10;
int in4 = 11;
```

```
int IR1 = A4;
// int IR2=7;//
int IR3 = 4;
// int IR4=A4;//
int IR5 = A5;
```

```
const int trigPin = A3;
const int echoPin = A2;
long duration;
int distance;
```

```
//sd card
File myFile;
```

```
// change this to match your SD shield or module;
const int chipSelect = 10;
```

```
void setup() {
  Serial.begin(9600);
  //  forward();
  //  delay(2000);
```

```
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
```

```
  pinMode(IR1, INPUT);
  pinMode(IR3, INPUT);
```

ESP:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
//color
// Define color sensor pins
#define S0 27
#define S3 25
#define S1 26
#define S2 12
#define sensorOut 14

// Variables for Color Pulse Width Measurements
int redPW = 0;
int greenPW = 0;
int bluePW = 0;
//voltage
// Define analog input
#define ANALOG_IN_PIN 32

// Floats for ADC voltage & Input voltage
float adc_voltage = 0.0;
float in_voltage = 0.0;

// Floats for resistor values in divider (in ohms)
float R1 = 30000;
float R2 = 7500;

// Float for Reference Voltage
float ref_voltage = 3.3;

// Integer for ADC value
int adc_value = 0;
 //
```

```
void setup() {
  lcd.init();
  lcd.backlight();
  lcd.print("MES");
```

```
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
```

```
  digitalWrite(S0,HIGH);
```

```
  pinMode(IR5, INPUT);
```

```
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
```

```
  Serial.print("Initializing SD card...");
```

```
  if (!SD.begin()) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
```

```
  // open the file. note that only one file can be
open at a time,
  // so you have to close this one before opening
another.
  myFile = SD.open("MES2.txt", FILE_WRITE);
```

```
  // if the file opened okay, write to it:
  if (myFile) {
    myFile.println("ADIOS !!!!");
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
}
```

```
void loop() {
  int distance;
  distance = Ultra();
```

```
  int value1, value3, value5;
  value1 = digitalRead(IR1);
  value3 = digitalRead(IR3);
  value5 = digitalRead(IR5);
```

```
// if (myFile) {
      if (value1 == 0 && value3 == 1 && value5 == 0)
{
      forward();
      myFile.println("FORWARD");
```

```
    } else if (value1 == 1 && value3 == 0 && value5
== 0) {

          right();
      myFile.println("LEFT");
```

```
    } else if (value1 == 0 && value3 == 0 && value5
== 1) {
    left();
      myFile.println("RIGHT");
```

```
    } else if (value1 == 0 && value3 == 0 && value5
== 0) {
      rev();
      myFile.println("STOP");
    }
```

```
  digitalWrite(S1,LOW);
  pinMode(sensorOut, INPUT);
}
```

```
void loop() {
float current,voltage;
current=getAnalogAmp();
voltage=volt();
color();
lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("VOLTAGE : ");
  lcd.setCursor(9,0);
  lcd.print(voltage);
  lcd.setCursor(0, 1);
  lcd.print("CURRENT : ");
  lcd.setCursor(9, 1);
  lcd.print(current);
}
```

VOLT:

```
float volt(){
   // Read the Analog Input
   adc_value = analogRead(ANALOG_IN_PIN);

   // Determine voltage at ADC input
   adc_voltage  = (adc_value * ref_voltage) / 4096.0;

   // Calculate voltage at divider input
   in_voltage = adc_voltage / (R2/(R1+R2)) ;

   // Print results to Serial Monitor to 2 decimal
places
  Serial.print("Input Voltage = ");
  Serial.println(in_voltage, 2);
  in_voltage=in_voltage+2;
  return in_voltage;
}
```

CURRENT:

```
float getAnalogAmp()
{
  //Measuring Current Using ACS712

  int   sampling = 250;
```

```
  double mVperAmp = 66; // use 185 for 5A Module, 100
for 20A Module, and 66 for 30A Module
  float ACSoffset = 2470.345;
```

```
  double RawValue = 0;
  double V = 0;
  double Amps = 0;
```

```
  for(int i=0; i<sampling; i++)
    {
    RawValue += analogRead(23);  //pin
    delay(1);
    }

  RawValue /= sampling;
```

```
    if (distance < 13) {
      stop();
```

```
      // right();
      // right();
      myFile.println("REVERSE BECAUSE THERE WAS
OBSTACLE OBSERVED");
    }
    myFile.close();
// }

}
```

MOTOR:

```
void forward() {
  analogWrite(enA, 85);
  analogWrite(enB, 85);

  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
void left() {
  analogWrite(enA, 60);
  analogWrite(enB, 60);
```

```
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
void right() {
  analogWrite(enA, 60);
  analogWrite(enB, 60);
```

```
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void stop() {
  analogWrite(enA, 85);
  analogWrite(enB, 85);
```

```
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void rev(){
   analogWrite(enA, 85);
  analogWrite(enB, 85);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
```

ULTRASONIC:

```
  V = (RawValue / 4096.0) * 3300; // Gets you mV
  Amps = ((V - ACSoffset) / mVperAmp);
```

```
  Serial.print( "Amps:");
  Serial.println( Amps );
  return Amps;
```

```
}
```

COLOR:

```
void color(){
    // Read Red Pulse Width
  redPW = getRedPW();
  // Delay to stabilize sensor
  delay(200);

  // Read Green Pulse Width
  greenPW = getGreenPW();
  // Delay to stabilize sensor
  delay(200);
```

```
  // Read Blue Pulse Width
  bluePW = getBluePW();
  // Delay to stabilize sensor
  delay(200);
```

```
  // Print output to Serial Monitor
  Serial.print("Red PW = ");
  Serial.print(redPW);
  Serial.print(" - Green PW = ");
  Serial.print(greenPW);
  Serial.print(" - Blue PW = ");
  Serial.println(bluePW);
}
```

```
// Function to read Red Pulse Widths
int getRedPW() {
  // Set sensor to read Red only
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
  return PW;
}
```

```
// Function to read Green Pulse Widths
int getGreenPW() {
  // Set sensor to read Green only
  digitalWrite(S2,HIGH);
  digitalWrite(S3,HIGH);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
```

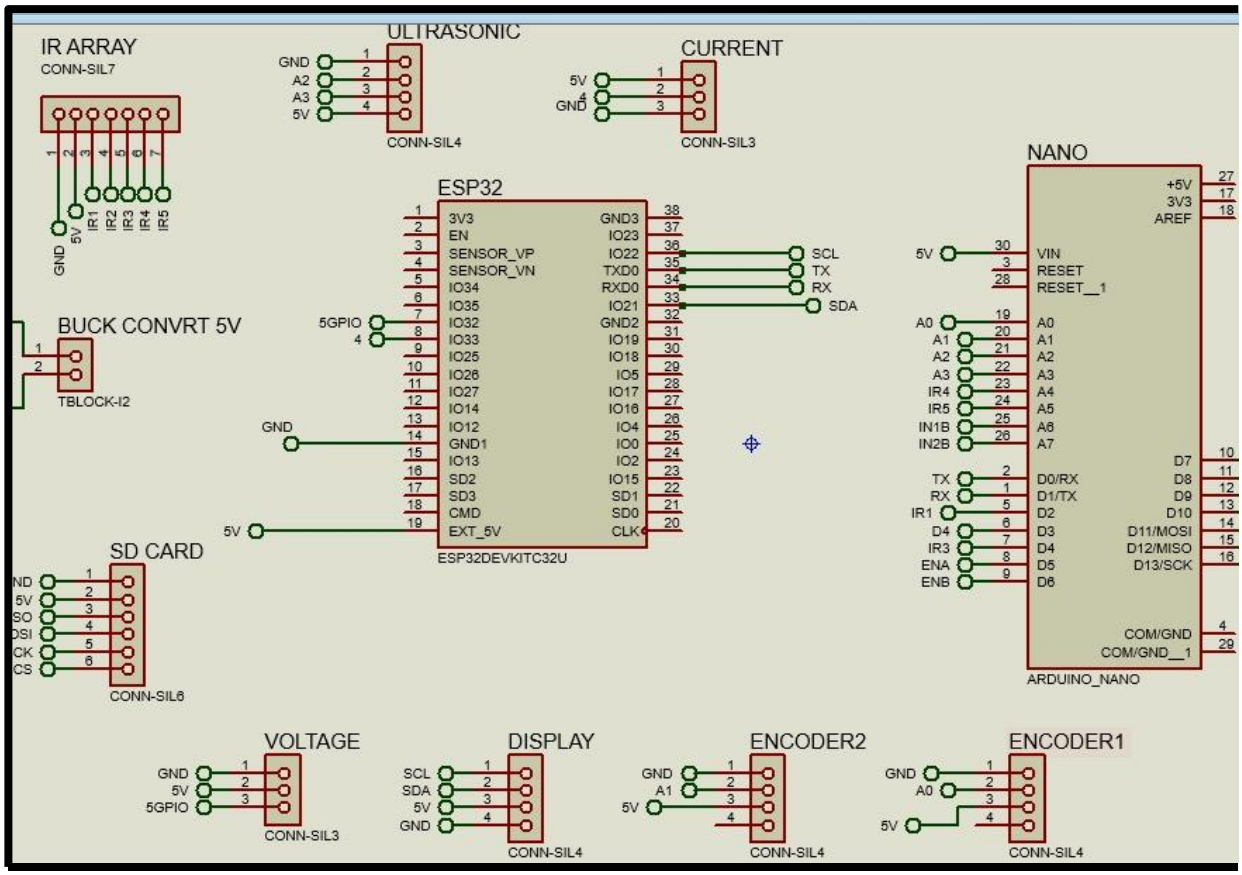
```
float Ultra() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro
seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel
time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance= duration*0.034/2;
// Prints the distance on the Serial Monitor
Serial.println(distance);
delay(1000);
return distance;
}
```

```
  return PW;
}

// Function to read Blue Pulse Widths
int getBluePW() {
  // Set sensor to read Blue only
  digitalWrite(S2,LOW);
  digitalWrite(S3,HIGH);
  // Define integer to represent Pulse Width
  int PW;
  // Read the output Pulse Width
  PW = pulseIn(sensorOut, LOW);
  // Return the value
  return PW;
}
```

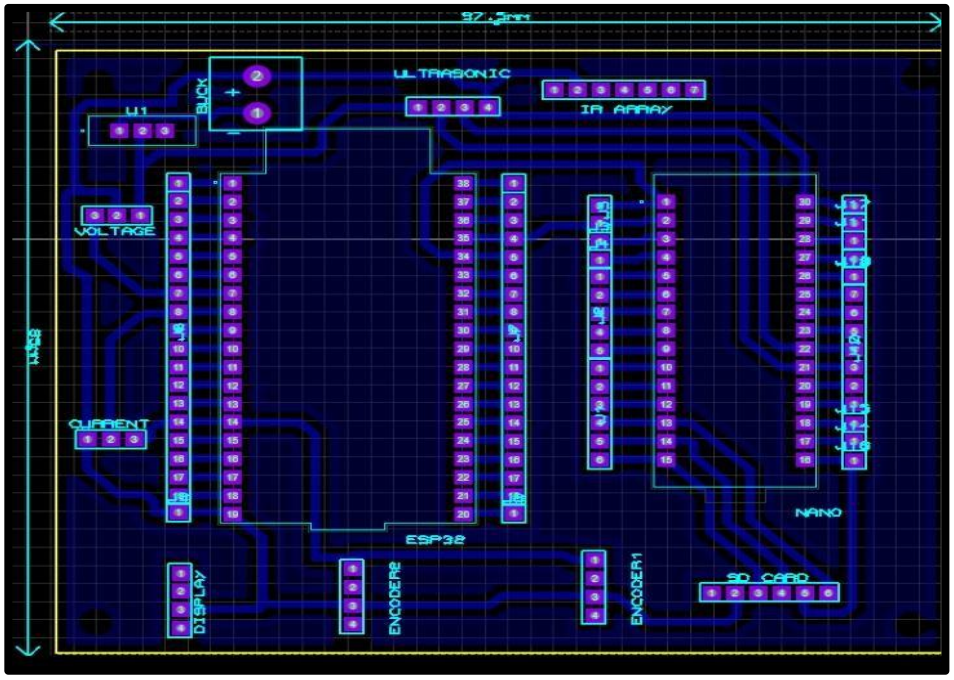
5.2 Simulations

We did our simulations in Proteus ISIS software. For our task many packages of microcontroller and sensors were required so we firstly added the libraries and then we integrated them together in our simulation according to the pinouts and block diagram we decided. Here is our diagram of our simulation in Proteus ISIS.

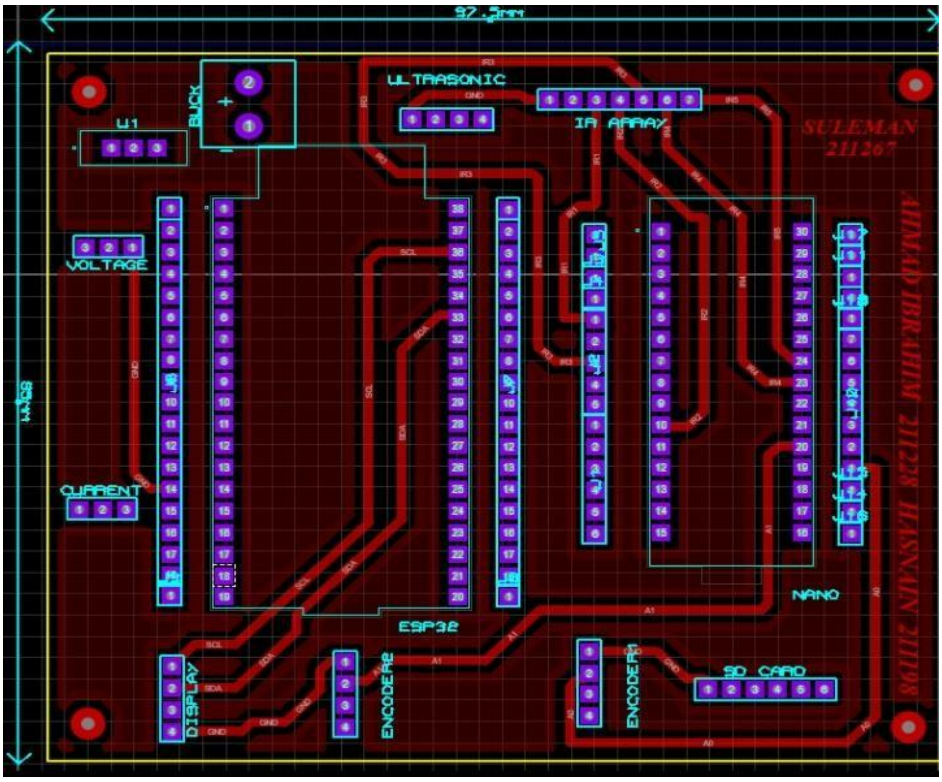


5.3 Simulation PCB

Bottom Copper



Top Copper



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 LCD all the values were displaying properly. I.e. current value voltage value and encoder value etc. we also checked our teacher the proper working.

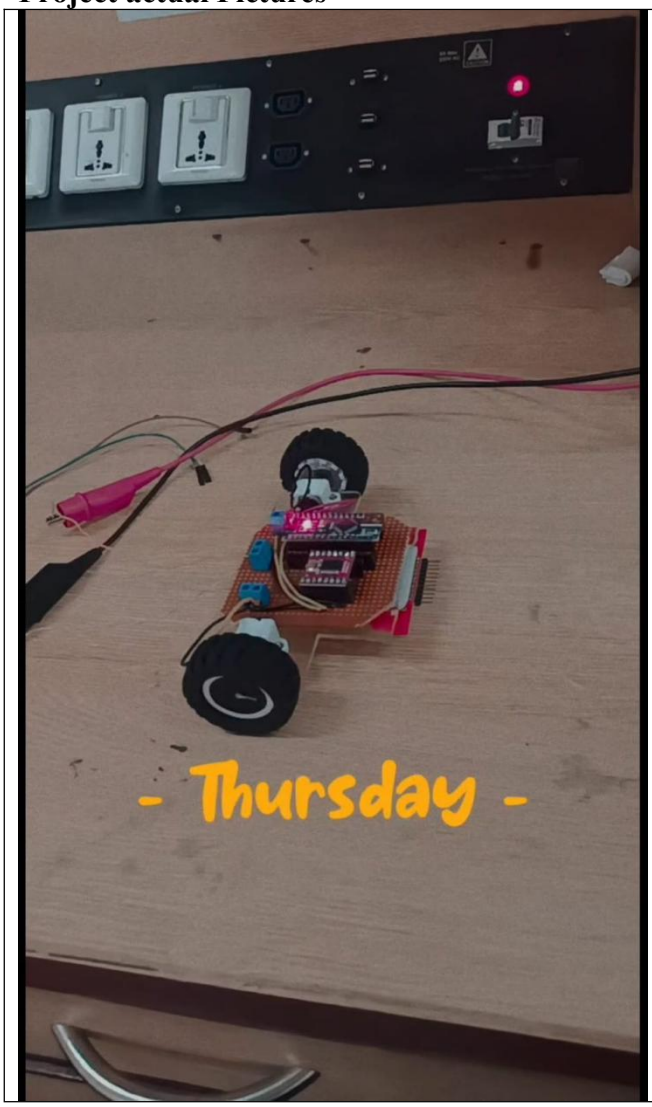
Third stage

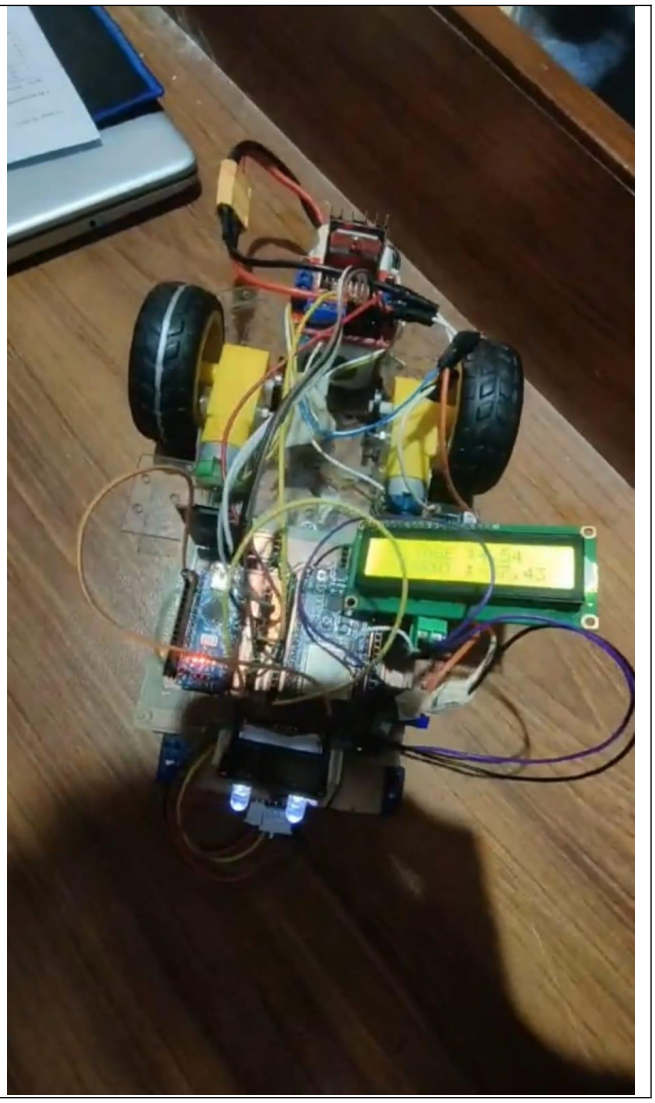
The third stage was testing on pcb board. We integrated the components on pcb while all working properly separately. After soldering we checked all the ports and pins of pcb board they were also fine all our sensors were giving the value properly. LCD was working fine also the other sensors. But our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

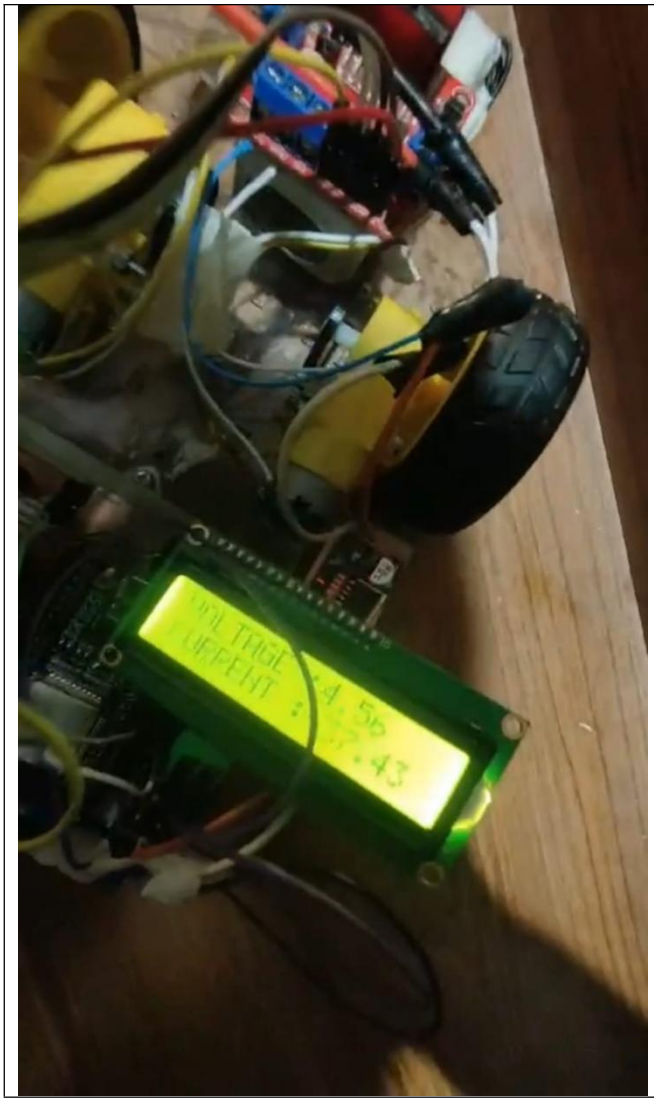
Are motor component being questionable as the speed of motor wasn't being controlled according to a requirement either it was too fast either it was too slow up on troubleshooting the issue we found out that we are connecting the motor drivers enable pains with the analogue pins of the microcontroller rather than connected them to PWM pins upon connecting the enable pens to the PWM pins of my true control speed was solved. Then then the second thing that was questionable was the improper behavior of the LCD display the LCD display was turning on but it didn't display any output data that we want to display on it the light it turns on the screen remain black so on troubleshooting the problem we identified that LCD needs and external 5 volt supply to work properly also using a voltage regulator so we added and different supply to the LCD from the output from the output of voltage regulator.

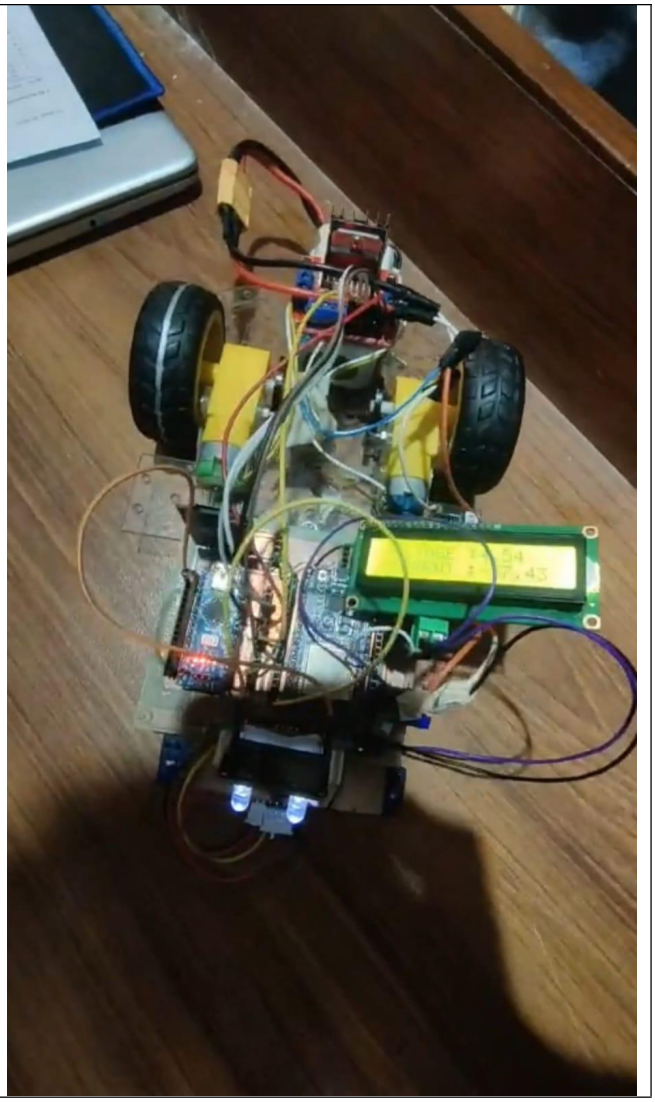
We didn't face any burning of the component as we take our component testing very well before the final design yes but there was a place placement issue of components in a design we did compromised some space that our components required in order to be place properly so to troubleshoot that we have used header pins to make a neat and clean placement of our components. Lastly one major change that we that we made in our design was to use I2C module for connecting LCD as connecting LCD directly with the microcontroller requires a lot of free pins from the controller and we did not have them so we use out to see module to connect a LCD and took place it in a suitable manner.

Project actual Pictures









CHAPTER 7 PROJECT MANAGEMENT

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

- In my capacity as the team lead, I was tasked with overseeing the compilation of a comprehensive program. To achieve this goal, one of my first actions was to categorize each member of the team based on their individual strengths and areas of expertise. This enabled me to assign tasks that would be best suited to each person's skill set, ensuring that everyone was able to contribute in a meaningful way.
-
- One team member, for example, had particular expertise in hardware and PCB design. With this in mind, I designated them to develop a Proteus-based PCB design for our project. Another team member had specialized knowledge in sensor interfacing, so I assigned them with a unique task related to this area.
-

- As for my own role within the project, I focused on consolidating all of the codes written by individuals into one final version. This was a critical step in ensuring that all components of the program worked seamlessly together. We used Nano and ESP32 controllers during testing and debugging procedures to ensure optimal performance.
-
- Through rigorous testing and attention to detail, we were able to achieve our goal of creating a fully functional program that met all requirements and exceeded expectations. Overall, I am proud of what we accomplished as a team and grateful for the opportunity to work with such talented individuals.

Risk management that you learned

Risk management is an essential process that involves identifying, assessing, and preventing potential risks that could negatively affect project outcomes. This crucial task is typically overseen by project managers who ensure that all necessary precautions are taken throughout the project's duration. In a recent project, a number of risks were encountered, including the challenge of dealing with incorrectly etched PCBs which necessitated inconvenient store purchases and difficulties with drilling and soldering PCB holes.

To mitigate any issues when using acids, we took great care to implement all necessary precautionary measures. Furthermore, our double-layered PCB presented a significant challenge in terms of alignment, while many components were not available in Proteus software so we had to create them ourselves. All these challenges required careful management and attention to detail to ensure successful completion of the project.

Through the process of risk management, we were able to identify potential problems early on and take appropriate steps to prevent them from derailing our progress. By being proactive in our approach and addressing issues as they arose, we were able to successfully navigate through the complexities of this project. Our commitment to risk management allowed us to deliver high-quality results while minimizing any negative impacts on our desired outcomes.

CHAPTER 8 FEEDBACK FOR PROJECT AND COURSE

Our task challenged our pre-existing knowledge and skills, while also providing us with valuable insights. Furthermore, it encouraged us to think expansively and critically about all available options. We acquired novel proficiencies such as soldering, which we had previously only studied or observed in demonstrations. In addition to technical expertise, this project helped us develop time-management and financial planning abilities. As individuals, each of us gained a deeper understanding of teamwork dynamics and collaboration strategies. All in all, we can confidently say that the experience was overwhelmingly positive.

CHAPTER 9 GITHUB LINK

https://github.com/engrpakistan/DegreeProject-