Members:

Muhammad Asad (201158)

BEMTS-F-20-A

Sir Umar Farooq

1

INDEX

Chapter 1: Preliminaries

Proposal:

The proposed project involves the development of an autonomous robot utilizing an Arduino or ESP32 microcontroller, L298N motor driver, and custom-designed PCB. The hardware components include a chassis equipped with motors, infrared sensors for line following, and ultrasonic sensors for obstacle detection. The Arduino IDE will be utilized for programming, implementing a PID-based line following algorithm, obstacle avoidance strategies, and integrating these systems seamlessly. To enhance scalability and efficiency, a custom PCB will be designed for neat and compact circuitry. The L298N motor driver will ensure precise control over the robot's movements. Challenges like sensor calibration and real-time decision-making will be addressed through iterative testing and algorithm refinement. The proposal emphasizes the project's significance and expresses excitement about the potential impact, while also providing a budget estimate and timeline for the development stages.

Initial feasibility:

The initial feasibility assessment of the proposed autonomous line follower and obstacle avoider robot project suggests promising prospects. The key components, including Arduino or ESP32 microcontrollers, the L298N motor driver, and sensors, are readily available and commonly used in robotics projects. The use of the Arduino IDE ensures a user-friendly programming environment, facilitating the development of the required algorithms for line following and obstacle avoidance. The integration of a custom-designed PCB demonstrates a commitment to efficient and organized circuitry, promoting scalability and ease of maintenance. Additionally, the project's objectives align with established principles in robotics, making it technically viable. While challenges such as sensor calibration and real-time decision-making exist, the iterative testing and refinement approach outlined in the proposal will likely lead to successful solutions. Considering the availability of resources and the project's alignment with industry-standard components and practices, the initial feasibility suggests a high likelihood of project success. Further detailed analysis and prototyping will be essential to confirm and address any unforeseen challenges.

Specifications of deliverables :

The proposed autonomous line follower and obstacle avoider robot project will involve the creation of a bespoke ATMEL controller-based board, showcasing the utilization of Proteus for simulation and allowing flexibility in choosing any software for PCB design. The incorporation of a dual-channel H-bridge configuration, whether as a custom design or modular implementation, ensures precise control over the robot's movements. For sustained operation, the robot will run on a battery for at least 20 minutes, with an emphasis on using Li-Po/Li-Ion batteries of at least 2000mAh capacity. To facilitate data logging, a Micro SD card module will

be integrated into the design. The identification of red-colored obstacles will be achieved through a combination of IR, ultrasonic, and color sensors, showcasing the project's reliance on I/O, ADC, Timers, Interrupts, and PWM on the embedded side. Additionally, the robot will be designed to transmit data, emphasizing a comprehensive integration of various functionalities to enhance its autonomy and versatility.

Team Roles & Details:

Team Member	Role	Word Count
Muhammad Asad	Software Implementer & Hardware Implementer. Technical Expert Design Incharge	650+700+300+350

Work Breakdown structure:

The team consists of members with diverse backgrounds in robotics, electronics, and programming. Roles were assigned based on individual strengths to foster effective collaboration and project development. One team member was responsible for programming and coding the entire robot, another handled the procurement of parts, report production, and etching. The PCB layout and schematic creation were entrusted to a different team member, while another took charge of etching and component attachment. With support from fellow team members, the soldering process was executed collaboratively, leading to the successful operation of the robot. This strategic allocation of responsibilities allowed the team to leverage each member's expertise, ensuring a cohesive and successful project outcome.

Estimated budget:

\

Components	Quantity	Cost/Item	Total
Battery	1	2700	2700
Arduino uno	1	2100	2100
PCB Board	4	450	1800
L2988	1	500	500
IR Sensor	2	120	240
Ultrasonic Sensor	1	120	120
Voltage Sensor	1	320	320
Current Sensor	1	320	320
SD-card Module	1	250	250
OLED display	1	500	500
Button	1	20	20
Color sensor	1	1250	1250
Print	13	70	910
HCL/H2SO4	1	400	400
Caster Wheel	1	100	100
Jumper Wires	3	140	420
Total	34	11950	11950

Chapter 2 Project Conception:

Objective:

The objective is to develop a fundamental, fully automatic line-following and obstacle-avoiding robot with various features. The primary goal is to create a simple line follower robot using Arduino IDE and ESP32, incorporating key components to serve as the foundation for future automated mobility vehicles.

Abstract:

The Line Follower Robot project establishes the groundwork for future mobility vehicles by integrating the ability to recognize lines, follow them, and adapt to turns. The robot continuously records and displays data from sensors, allowing users to comprehend its behavior and modify features as needed. Infrared sensors enable line recognition, while an ultrasonic sensor facilitates obstacle avoidance. The project employs ESP32, and various components like motors, color

sensor, LCD module, SD card module, voltage sensor, current sensor (ACS-712), and more for comprehensive functionality.

Introduction:

A line follower is a machine capable of following a path, typically represented by a black line on a white background. The robot senses the line, adjusts its course, and corrects movements using sensor feedback. This technology finds applications in industrial and automotive automation, guiding systems, and more.

List of features and operational specification of our project: Features:

- Proteus simulation using Arduino IDE and ESP32.
- Precise line recognition with two infrared sensors.
- Obstacle avoidance with an ultrasonic sensor.
- Two motors on wheels.
- Data logging with SD card and Arduino Uno.
- Motor speed control with a motor driver.
- LCD panel for displaying output voltage, input voltage, and current.

Operation:

The line follower uses optical sensors (IR-LED and photodiode) on its front end to record the line position with robustness and resolution. Two-wheel motion-governing motors implement a steering mechanism. The LCD display indicates the distance covered, and the robot moves in a circle until it detects a line when no black surface is detected.

Components:

- Arduino Mega
- Color sensor
- 16x2 LCD Module
- 1x SD Card Module
- 2x IR Sensors
- 1x Ultrasonic Sensor
- 1x Voltage Sensor
- 1x Current Sensor (ACS-712)
- 1x Robot Chassis
- 2x Motors (3V to 6V)
- Jumper Wires, Header Pins, Resistors
- On/Off Button
- Screws and Nuts
- 1x Motor Driver LM298
- 2x Rechargeable Battery (3.7V, connected to make a 12V battery)

Project Development Process:

- Concept definition after thorough research.
- Programming intelligence burned to Arduino and ESP32 using Arduino IDE.
- Evaluation of software accuracy and electronic component viability using Proteus simulation.
- Practical implementation with hardware.
- PCB board design for a polished and realistic-looking robot.
- Ensuring the stability and reliability of electrical and electronic systems.

Basic block diagrams of the whole system and subcomponents are created to illustrate circuit design components and project planning.

Basic block diagrams of whole system and subcomponents:

Here is a basic block diagram that illustrates the circuit design's key components and provides an overview of the project's planning and design process.

Literature Review:

Ultrasonic sensor:

Ultrasonic transducers and sensors produce or detect ultrasound radiation. The ESP32-based ultrasonic sensor sends signals that bounce off obstacles and return, determining distance based on signal travel time.

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 a 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 :

Ardiono is a widely used SoC microcontroller. It is low cost and highly versatile microcontroller used for various applications which include wireless communication, IoT (Internet of things) devices, home automation, robotics, embedded systems etc.

Robotic Chassis (2 Wheel with DC Motor):

The robotic chassis kit provides a sturdy foundation for the robot, including an acrylic base, gear motors, wheels, ball caster, and other components. It ensures structural integrity and stability during the robot's operation.

Package Contains:

1. 1 x Rubber wires

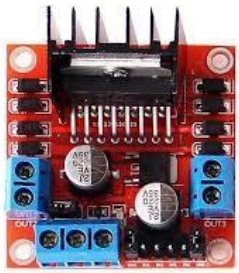
2. 1 x Deceleration motors
3. 1 x Aluminum fasteners
4. 1 x Nylon all-direction wheel
5. 1 x Chassis
6. 1 x Battery box (4 x AA batteries, not included)
7. 1 x Screwdriver.

IR sensor:

Infrared sensors, composed of IR LEDs and photodiodes, are deployed at the robot's front end for accurate line recognition. This sensor combination ensures robustness and high resolution, enabling the robot to precisely follow predefined paths.

L298N Motor Driver

The L298N Motor Driver is a high-power module that controls the direction and speed of DC motors. Integrated with an L298 motor driver IC and a 5V regulator, it plays a pivotal role in governing the robot's motion.



TT Gear Motor:

Controlled by the L298N motor driver, the TT Gear Motor combines a DC motor with a gearbox. This integration allows the robot to operate within an optimal speed range, essential for precise and controlled movements.

SD Card Module

The Micro SD Card Adapter module facilitates easy interfacing with SD cards, providing an SPI interface and a built-in 3.3V voltage regulator. It simplifies data logging and storage, enhancing the robot's capacity for information recording.

16 x 2 LCD Display with I2C module

The 16x2 LCD module, coupled with an I2C interface, simplifies control with minimal pins, offering a clear visual interface for displaying information, including output voltage, current, and other key data.

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 -

ACS712 Current Sensor

Based on the Hall Effect, the ACS712 Current Sensor precisely measures the current flowing through the robot. It provides valuable insights into the robot's power usage, allowing for optimization and efficient energy management.

VOLTAGE SENSOR

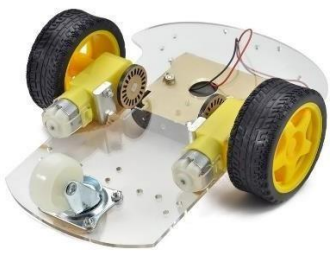
Employing a resistive voltage divider design, the voltage sensor reduces input voltage by a factor of five. This precise monitoring component aids in tracking the robot's power consumption and ensuring stable operation.

Chapter 3 Mechanical Design:

The significance of meticulous mechanical design within the realm of engineering projects cannot be overstated, and its profound impact is particularly pronounced in the context of the line-following robot project. The effectiveness and endurance of the robot, its ability to bear loads, and its overall sustainability are intrinsically linked to the quality of its mechanical design. It is imperative to emphasize that the longevity, load-bearing capacity, and sustainability of the robot hinge upon the precision and ingenuity applied in crafting its mechanical structure. Therefore, it is imperative that the design process minimizes both external and internal constraints on the choice of materials and the overall design framework, ensuring optimal functionality and longevity of the line-following robot.

Mechanism selection and Platform Design:

The assembly of the robot chassis is a seamless and expeditious process, facilitating the swift establishment of a mobile robotics platform. This expeditious construction proves particularly advantageous in situations where time constraints or limited fabrication tools preclude the creation of a bespoke robotic chassis. Notably, robot chassis designs are characterized by their incorporation of multiple pre-drilled holes and slots, strategically positioned to expedite the seamless integration of supplementary components, including sensors. Recognizing these advantages, our project strategically leveraged the efficiency of a commercially available robot chassis, providing a robust foundation. Subsequently, we mounted a meticulously designed PCB, housing all essential circuitry, onto this agile and adaptable platform. This approach not only streamlines the construction process but also optimizes the integration of sophisticated electronic components for enhanced functionality.



Actuators with speciation and datasheet

We employed gear motors with the following specifications as actuators

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

Chapter 4 Software Design:

Controller Selections with features

The selection of the Arduino microcontroller for our line-following robot project was driven by its multifaceted capabilities, aligning seamlessly with the specific requirements of our design. The Ardino's extensive array of GPIO pins proved instrumental in accommodating the diverse sensor array essential for precise line-following and obstacle avoidance. Its robust processing power and built-in functionalities, including timers and interrupts, provided a comprehensive solution for managing real-time tasks crucial for the project's success. Therefore, the Arduino was the optimal choice, embodying a balance between computational power, wireless communication capabilities, and the requisite number of pins for a sophisticated line-following robotic system.

Software Design details & user Requirements
Components

Components that we used it the software design

- Ultrasonic sensor
- SD card module
- IR sensors
- L298 motor driver ● Color sensor
- 16x2 LCD display
- Motors
- Current sensor.
- Voltage sensor

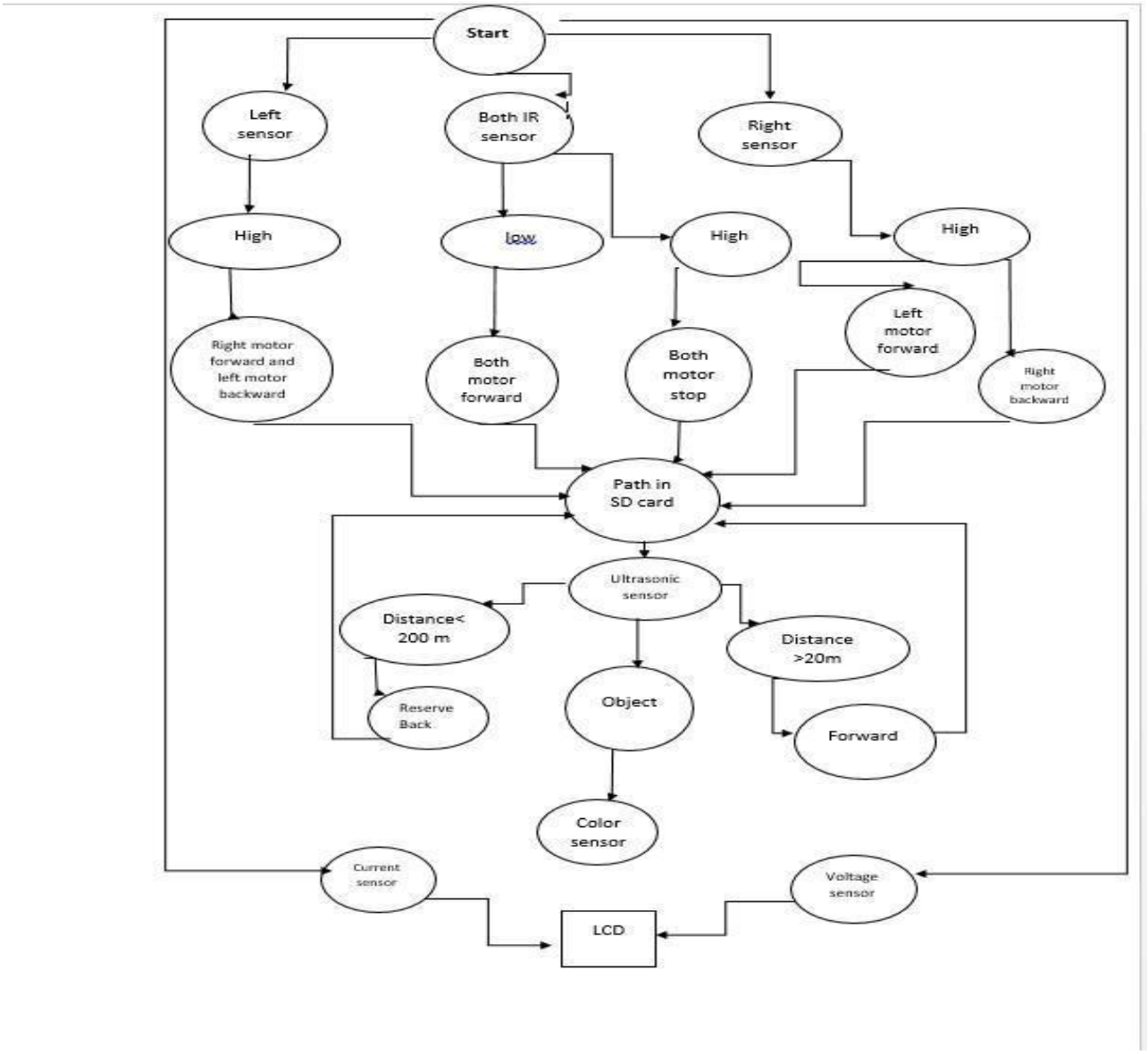
Inputs:

- Ultrasonic Sensor
- Voltage sensor
- Current sensor
- Color sensor

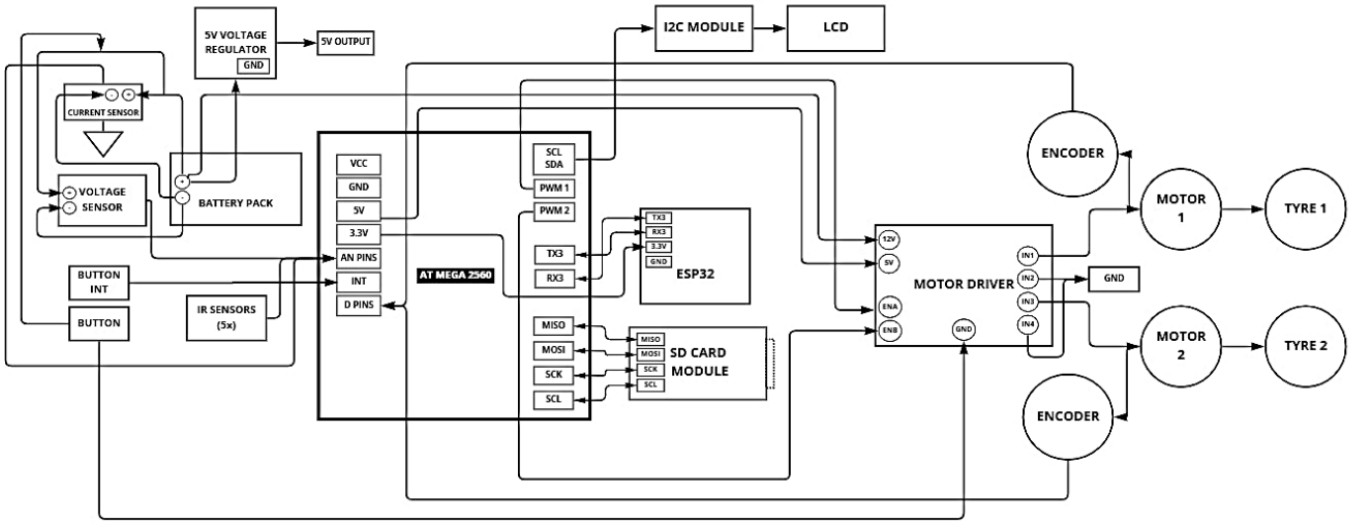
Outputs

- Left Motor
- Right Motor ● SD card
- 16 X 2 LCD

State Machine & System flow diagram



Block diagram:



Chapter 5: Simulations and Final Integration

Integrations and testing all hardware and software component separately

We started by reading each component's data sheet and making a note of the ideal voltage and current for each one. Next, each component was tested independently using a DMM. The majority of the parts were operating as intended. The parts that needed to be replaced were not functional. Following component inspection, we assembled the parts. We were very careful during integration to make sure that no component was shorted or inserted in the incorrect pin, as this could cause the component to burst.

Compiled Arduino code:

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH110X.h>


#define LMS  A4
#define LS   A3
#define CS   A2
#define RS   A1
#define RMS  A0
#define enA  10
```

```
#define enB  11
#define MotorAip1  5
#define MotorAip2  4
#define MotorBip1  3
#define MotorBip2  17

#define i2c_Address  0x3c
#define screen_width  128
#define screen_height  64
#define OLED_RESET     -1

#define Current_sensor A6
#define Voltage_sensor A7

#define right_Encoder  27
#define left_Encoder   26

#define Ultra_Trigger  7
#define Ultra_Echo     6

const int chipSelect = 53;
float current = 0.00;
float voltage = 0.00;
int Obstacle_distance = 0;
int rpm_right = 0;
int rpm_left = 0;
int RPM = 0;

Adafruit_SH1106G display = Adafruit_SH1106G(screen_width,screen_height,&Wire,OLED_RESET);
void setup()
{
  display.begin(i2c_Address,true);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SH110X_WHITE);
```

```
display.setCursor(0, 10);
display.println("Initializing SD card.");
display.display();
delay(1000);
if (!SD.begin(chipSelect))
{
  display.println("Card failed, or not present");
  display.display();
  delay(2000);
}
else
{
  display.println("Card initialized.");
  display.display();
  delay(3000);
}

pinMode(enA, OUTPUT);
pinMode(enB, OUTPUT);
pinMode(LMS, INPUT);
pinMode(LS,  INPUT);
pinMode(CS,  INPUT);
pinMode(RS,  INPUT);
pinMode(RMS, INPUT);
pinMode(MotorAip1,OUTPUT);
pinMode(MotorAip2,OUTPUT);
pinMode(MotorBip1,OUTPUT);
pinMode(MotorBip2,OUTPUT);
analogWrite(enA, 150);
analogWrite(enB, 150);

pinMode(Current_sensor, INPUT);
pinMode(Voltage_sensor, INPUT);
pinMode(Ultra_Echo, INPUT);
pinMode(Ultra_Trigger, OUTPUT);
```

```
  pinMode(right_Encoder, INPUT_PULLUP);
  pinMode(left_Encoder, INPUT_PULLUP);

  display.clearDisplay();
  display.setTextSize(1.8);
  display.setTextColor(SH110X_WHITE);
  display.setCursor(20, 15);
  display.println("Ready to Go..!!!");
  display.display();
}

void loop()
{
   if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==0 && digitalRead(RS)==1
&& digitalRead(RMS)==1)
   forward();
   else if(digitalRead(LMS)==1 && digitalRead(LS)==0 && digitalRead(CS)==0 &&
digitalRead(RS)==0 && digitalRead(RMS)==1)
   forward();
   else if(digitalRead(LMS)==1 && digitalRead(LS)==0 && digitalRead(CS)==1 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
   left();
   else if(digitalRead(LMS)==0 && digitalRead(LS)==1 && digitalRead(CS)==1 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
   left();
   else if(digitalRead(LMS)==0 && digitalRead(LS)==0 && digitalRead(CS)==1 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
   left();
   else if(digitalRead(LMS)==1 && digitalRead(LS)==0 && digitalRead(CS)==0 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
   left();
   else if(digitalRead(LMS)==0 && digitalRead(LS)==0 && digitalRead(CS)==0 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
   left();
   else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==1 &&
digitalRead(RS)==0 && digitalRead(RMS)==1)
   right();
```

```
   else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==1 &&
digitalRead(RS)==1 && digitalRead(RMS)==0)
   right();
   else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==1 &&
digitalRead(RS)==0 && digitalRead(RMS)==0)
   right();
   else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==0 &&
digitalRead(RS)==0 && digitalRead(RMS)==1)
   right();
   else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==0 &&
digitalRead(RS)==0 && digitalRead(RMS)==0)
   right();
   else
   stop();
}

//void  Line_follower()
//{
//   if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==0 && digitalRead(RS)==1
&& digitalRead(RMS)==1)
//   forward();
//   else if(digitalRead(LMS)==1 && digitalRead(LS)==0 && digitalRead(CS)==0 &&
digitalRead(RS)==0 && digitalRead(RMS)==1)
//   forward();
//   else if(digitalRead(LMS)==1 && digitalRead(LS)==0 && digitalRead(CS)==1 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
//   left();
//   else if(digitalRead(LMS)==0 && digitalRead(LS)==1 && digitalRead(CS)==1 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
//   left();
//   else if(digitalRead(LMS)==0 && digitalRead(LS)==0 && digitalRead(CS)==1 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
//   left();
//   else if(digitalRead(LMS)==1 && digitalRead(LS)==0 && digitalRead(CS)==0 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
//   left();
//   else if(digitalRead(LMS)==0 && digitalRead(LS)==0 && digitalRead(CS)==0 &&
digitalRead(RS)==1 && digitalRead(RMS)==1)
```

```
//    left();
//    else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==1 &&
digitalRead(RS)==0 && digitalRead(RMS)==1)
//    right();
//    else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==1 &&
digitalRead(RS)==1 && digitalRead(RMS)==0)
//    right();
//    else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==1 &&
digitalRead(RS)==0 && digitalRead(RMS)==0)
//    right();
//    else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==0 &&
digitalRead(RS)==0 && digitalRead(RMS)==1)
//    right();
//    else if(digitalRead(LMS)==1 && digitalRead(LS)==1 && digitalRead(CS)==0 &&
digitalRead(RS)==0 && digitalRead(RMS)==0)
//    right();
//    else
//    stop();
//}

void forward()
{
//  forward_indication();

   digitalWrite(MotorAip1, HIGH);
   digitalWrite(MotorAip2, LOW);
   digitalWrite(MotorBip1, LOW);
   digitalWrite(MotorBip2, HIGH);
}

void left()
{
//    left_indication();

   digitalWrite(MotorAip1, HIGH);
   digitalWrite(MotorAip2, LOW);
   digitalWrite(MotorBip1, LOW);
```

```
  digitalWrite(MotorBip2, LOW);
}

void right()
{
//   right_indication();

  digitalWrite(MotorAip1, LOW);
  digitalWrite(MotorAip2, LOW);
  digitalWrite(MotorBip1, LOW);
  digitalWrite(MotorBip2, HIGH);
}

void reverse()
{
  digitalWrite(MotorAip1, LOW);
  digitalWrite(MotorAip2, HIGH);
  digitalWrite(MotorBip1, HIGH);
  digitalWrite(MotorBip2, LOW);
}

void stop()
{
//   voltage = Voltage();
//   current = Current();
//   Obstacle_distance = Ultrasonic();
//   display_data();

  digitalWrite(MotorAip1, LOW);
  digitalWrite(MotorAip2, LOW);
  digitalWrite(MotorBip1, LOW);
  digitalWrite(MotorBip2, LOW);
}

//float Current()
```

```
//{
//  unsigned int x=0;
//  float AcsValue=0.0,Samples=0.0,AvgAcs=0.0,AcsValueF=0.0;
//    for (int x = 0; x < 150; x++)
//    {
//      AcsValue = analogRead(Current_sensor);
//      Samples = Samples + AcsValue;
//      delay (3);
//    }
//      AvgAcs=Samples/150.0;
//      AcsValueF = (-2.5 + (AvgAcs * (5.0 / 1024.0)) )/0.185;
//      delay(50);
//      return AcsValueF;
//}
//
//float Voltage()
//{
//    float adc_voltage = 0.0;
//    float in_voltage = 0.0;
//    float R1 = 30000.0;
//    float R2 = 7500.0;
//    float ref_voltage = 5.0;
//    int adc_value = 0;
//
//   adc_value = analogRead(Voltage_sensor);
//   adc_voltage  = (adc_value * ref_voltage) / 1024.0;
//   in_voltage = (adc_voltage / (R2/(R1+R2)))-3.23 ;
//   return in_voltage;
//}
//
//int Ultrasonic()
//{
//  long Duration;
//  int Distance;
//  digitalWrite(Ultra_Trigger, LOW);
```

```
//  delayMicroseconds(2);
//  digitalWrite(Ultra_Trigger, HIGH);
//  delayMicroseconds(10);
//  digitalWrite(Ultra_Trigger, LOW);
//  Duration = pulseIn(Ultra_Echo, HIGH);
//  Distance = Duration * 0.034 / 2;
//  return Distance;
//}
//
//float Encoder_right()
//{
//    unsigned long start_time = 0;
//    unsigned long end_time = 0;
//    int steps=0;
//    float steps_old=0;
//    float temp=0;
//    float rpm=0;
//    start_time=millis();
//    end_time=start_time+100;
// while(millis()<end_time)
// {
//   if(digitalRead(right_Encoder))
//   {
//    steps=steps+1;
//    while(digitalRead(right_Encoder));
//   }
// }
//    temp=steps-steps_old;
//    steps_old=steps;
//    rpm=(temp/20)*10*60;
//    return rpm;
//}
//
//float Encoder_left()
//{
```

```
//    unsigned long start_time = 0;
//    unsigned long end_time = 0;
//    int steps=0;
//    float steps_old=0;
//    float temp=0;
//    float rpm=0;
//    start_time=millis();
//    end_time=start_time+100;
// while(millis()<end_time)
// {
//   if(digitalRead(left_Encoder))
//   {
//    steps=steps+1;
//    while(digitalRead(left_Encoder));
//   }
// }
//    temp=steps-steps_old;
//    steps_old=steps;
//    rpm=(temp/20)*10*60;
//    return rpm;
//}
//int average_rpm(int rpm1, int rpm2)
//{
//  return (rpm1+rpm2)/2;
//}
//
//void SD_card()
//{
//  // make a string for assembling the data to log:
//  String dataString = "";
//
//  // read three sensors and append to the string:
//  for (int analogPin = 0; analogPin < 3; analogPin++) {
//    int sensor = analogRead(analogPin);
//    dataString += String(sensor);
```

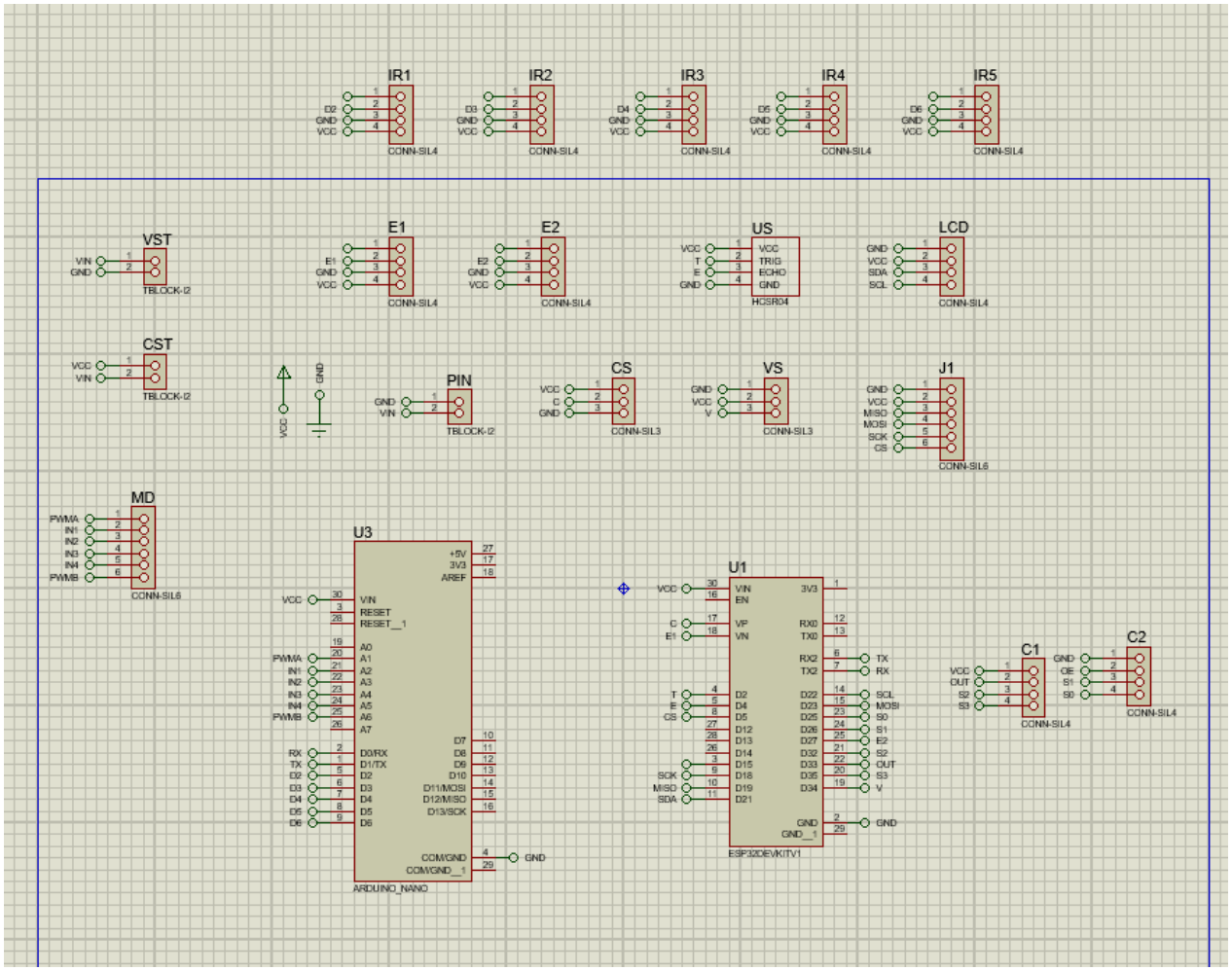
```
//    if (analogPin < 2) {
//      dataString += ",";
//    }
//  }
//
//  // open the file. note that only one file can be open at a time,
//  // so you have to close this one before opening another.
//  File dataFile = SD.open("datalog.txt", FILE_WRITE);
//
//  // if the file is available, write to it:
//  if (dataFile) {
//    dataFile.println(dataString);
//    dataFile.close();
//    // print to the serial port too:
//    Serial.println(dataString);
//  }
//  // if the file isn't open, pop up an error:
//  else {
//    Serial.println("error opening datalog.txt");
//  }
//}
//void forward_indication()
//{
//  display.clearDisplay();
//  display.setCursor(0, 10);
//  display.writeLine(64,5,54,20,SH110X_WHITE);
//  display.writeLine(64,5,74,20,SH110X_WHITE);
//  display.writeLine(64,10,54,25,SH110X_WHITE);
//  display.writeLine(64,10,74,25,SH110X_WHITE);
//  display.writeLine(64,15,54,30,SH110X_WHITE);
//  display.writeLine(64,15,74,30,SH110X_WHITE);
//  display.display();
//}
//void left_indication()
//{
```

```
//  display.clearDisplay();
//  display.setCursor(0, 10);
//  display.writeLine(64,5,49,15,SH110X_WHITE);
//  display.writeLine(64,25,49,15,SH110X_WHITE);
//  display.writeLine(59,5,44,15,SH110X_WHITE);
//  display.writeLine(59,25,44,15,SH110X_WHITE);
//  display.writeLine(54,5,39,15,SH110X_WHITE);
//  display.writeLine(54,25,39,15,SH110X_WHITE);
//  display.display();
//}
//void right_indication()
//{
//  display.clearDisplay();
//  display.setCursor(0, 10);
//  display.writeLine(64,5,79,15,SH110X_WHITE);
//  display.writeLine(64,25,79,15,SH110X_WHITE);
//  display.writeLine(69,5,84,15,SH110X_WHITE);
//  display.writeLine(69,25,84,15,SH110X_WHITE);
//  display.writeLine(74,5,89,15,SH110X_WHITE);
//  display.writeLine(74,25,89,15,SH110X_WHITE);
//  display.display();
//}
//void display_RPM(int value)
//{
//  display.setTextSize(2.2);
//  display.setTextColor(SH110X_WHITE);
//  display.setCursor(35, 40);
//  display.print(value);
//  display.setTextSize(1);
//  display.setCursor(100, 40);
//  display.print("rpm");
//  display.display();
//}
//
//void display_data()
```

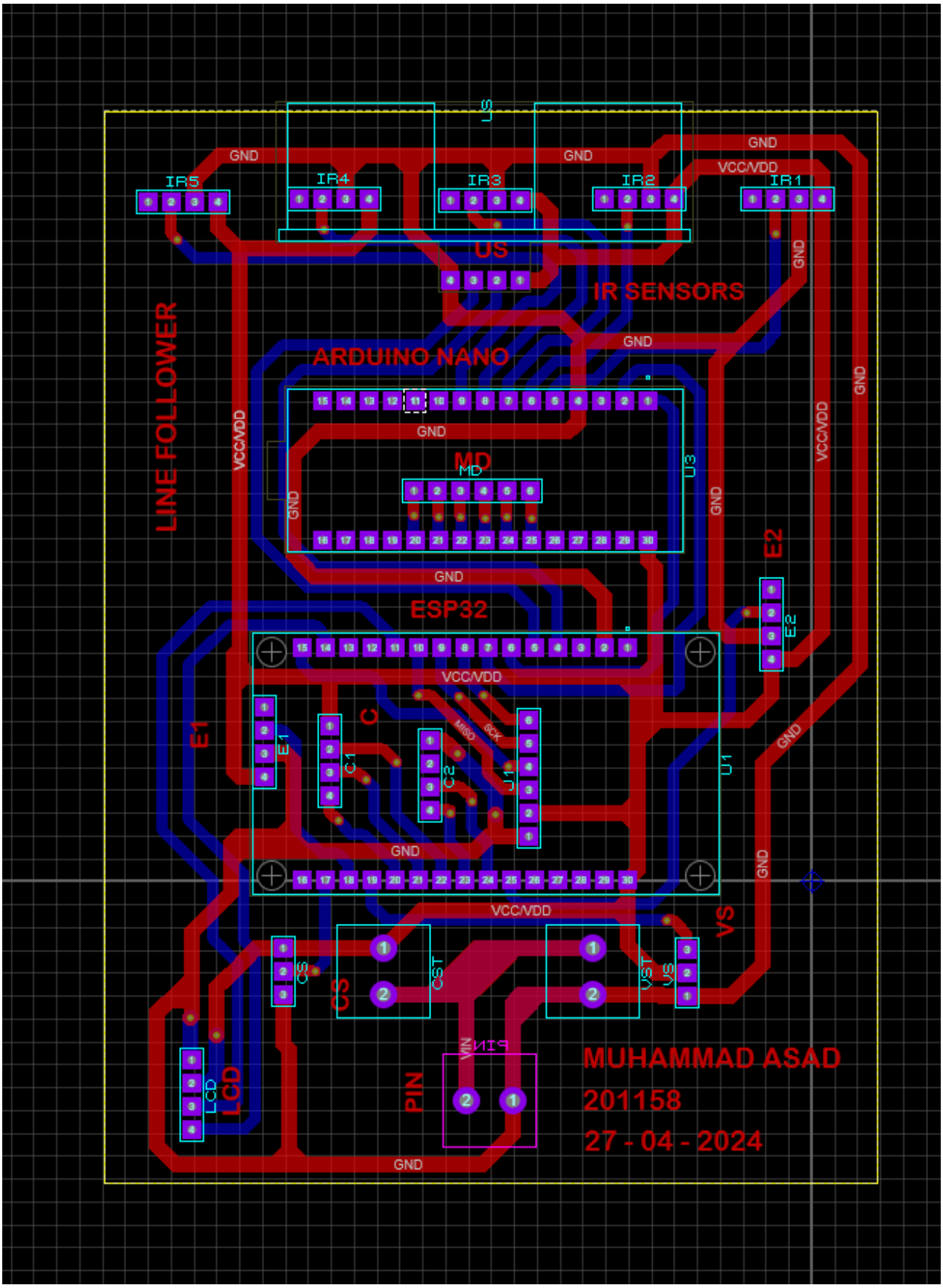
```
//{
//  display.clearDisplay();
//  display.setCursor(10, 20);
//  display.setTextSize(0.5);
//  display.setTextColor(SH110X_WHITE);
//  display.print("Voltage: ");
//  display.print(voltage, 2);
//  display.print(" V");
//  display.setCursor(10, 30);
//  display.print("Current: ");
//  display.print(current, 2);
//  display.print(" mA");
//  display.setCursor(10, 40);
//  display.print("Obstacle: ");
//  display.print(Obstacle_distance, 10);
//  display.print(" cm");
//  display.display();
//}
```

Simulations

We used Proteus ISIS software to conduct our simulations. Many packages of microcontrollers and sensors were needed for our assignment, so we included the libraries first, then merged them all together in our simulation using the block diagram and pinouts that we decided upon. This is the schematic for our Proteus ISIS simulation.



Simulation PCB:



Chapter 6: System Testing

Final testing First stage

Our initial phase involved a meticulous examination of each component in isolation. This scrutiny revealed a few instances where replacements were imperative due to malfunctions. A notable example was the initial motor driver, which encountered a malfunction leading to burnout and necessitating its substitution. Subsequently, we seamlessly integrated each individual element into the overall system, orchestrating a harmonious fusion of components to ensure the coherence and functionality of the entire assembly. This systematic approach allowed us to address and rectify any issues at the component level before proceeding to the comprehensive integration phase, thereby enhancing the overall robustness and reliability of the project.

Second stage

In the subsequent phase of our project, a meticulous testing procedure was undertaken, employing a breadboard to systematically evaluate each individual module. This rigorous testing regimen confirmed the optimal functionality of every module, with each component performing as intended. The real-time display on our LCD served as a comprehensive indicator of the system's health, showcasing accurate values such as voltage, current, encoder readings, and more, while the robot adeptly traversed its designated path. An additional layer of assurance was achieved through the thorough validation of our teacher, ensuring that external factors were seamlessly integrated into our testing framework. This comprehensive testing approach not only validated the integrity of each module but also provided valuable insights into the overall cohesion and performance of our robotic system.

Third stage

In the third phase of our project, we transitioned to testing on a dedicated PCB board after ensuring the individual components operated flawlessly in isolation. The integration process involved meticulous soldering to bring all elements together seamlessly. A comprehensive examination of the PCB board's ports and pins ensued, revealing their impeccable condition. Notably, our sensors consistently delivered accurate values, affirming their reliable performance. The LCD, along with other sensors, demonstrated optimal functionality as anticipated. However, a nuanced challenge emerged during the evaluation – despite the motor driver accurately receiving voltage, a discrepancy arose in its mapping to the motors, indicating a crucial focal point for further investigation and refinement in our pursuit of a fully operational robotic system.

Things that are questionable and get burned again and again

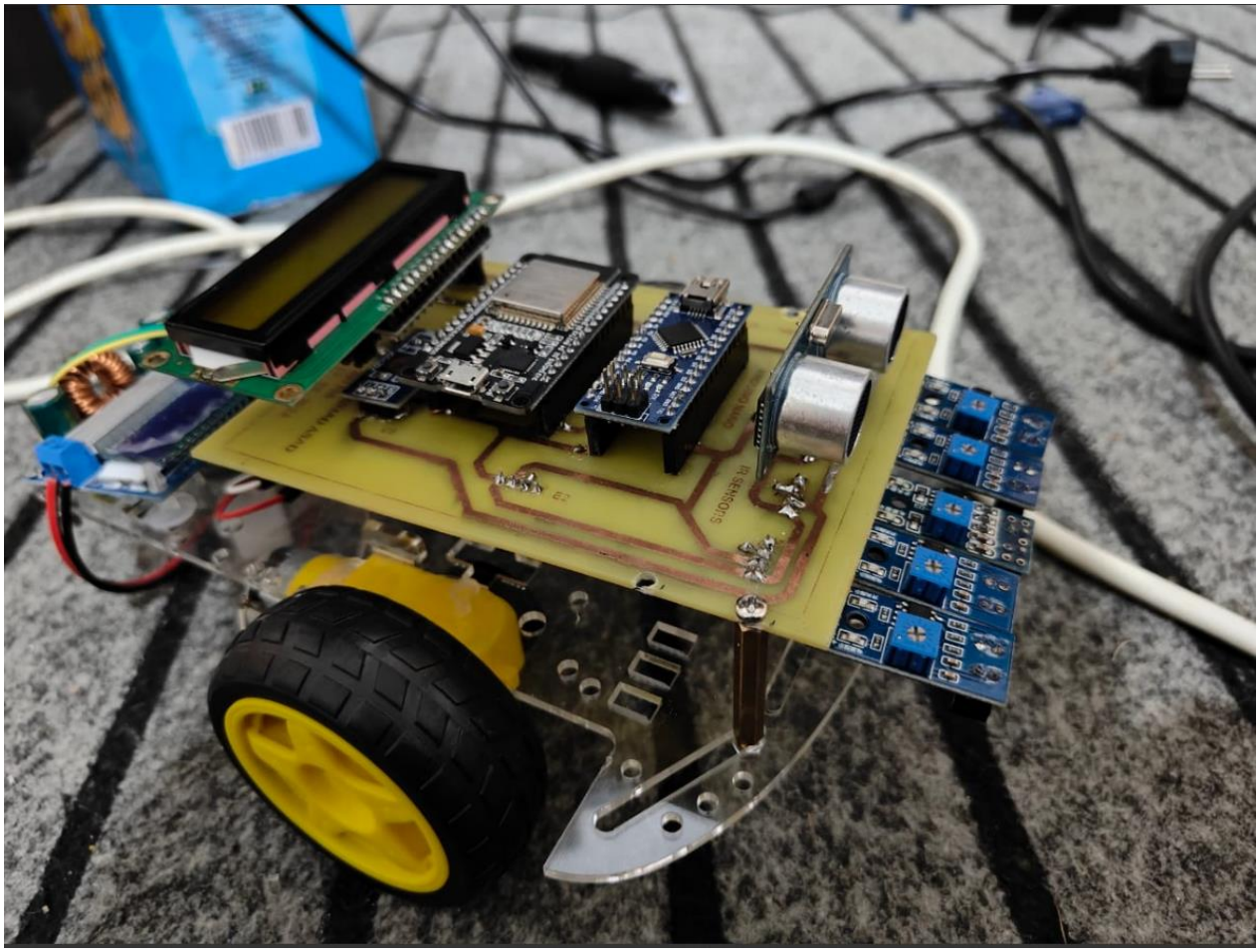
The recurring challenges and persistent issues in our project primarily centered around two critical aspects: the etching process for PCBs and the integration of sensors.

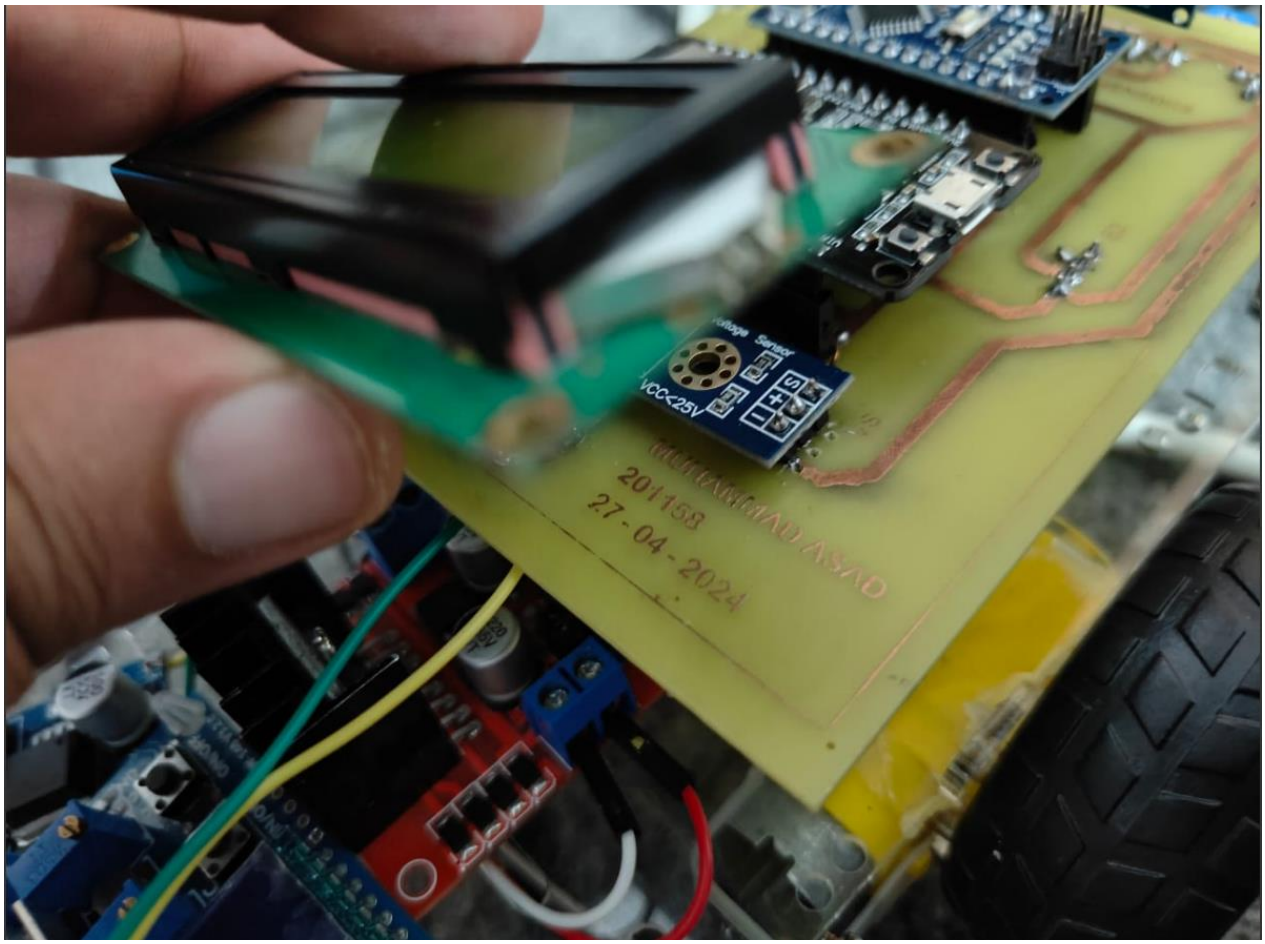
Initially, during the third testing stage, we encountered difficulties with the first PCB, which was not properly etched. This issue prompted us to shift to a second PCB for improved results. Despite overcoming the etching challenge with the second PCB, we faced

complexities while interfacing sensors during integration. The sensors did not seamlessly interact with the PCB, posing a potential setback to the project.

To address these challenges, our team engaged in troubleshooting efforts. For the etching problem, a thorough examination of the etching process was conducted, leading to refinements that ensured the proper execution of the second PCB. Regarding sensor interfacing, careful adjustments and meticulous attention to the connection details were made to ensure seamless integration. These efforts successfully resolved the etching and interfacing challenges, culminating in a well-functioning and optimized final design for our project.

Hardware:





Chapter 7: Project Management

A schematic and PCB for the project were skillfully crafted without specifying an individual. Initially, a recommendation to use Proteus software was made, but due to the substantial number of sensors required, an alternative choice was sought. EasyEDA was selected, and proficiency in the program was acquired through online tutorials, eliminating the steep learning curve. Despite encountering issues in early versions, where component and wire spacing were incorrect and posed a risk of short circuits, efforts rectified the problem, ensuring proper functionality. In summary, significant contributions were made to the project's hardware design and assembly.

The project's code was authored using the Arduino IDE. The coding style prioritized accessibility and clarity, making it comprehensible for individuals with diverse technical backgrounds. Proficiency in working with a variety of sensors, including infrared, ultrasonic, color, voltage, and current, was demonstrated. Their adeptness in seamlessly integrating and assembling hardware further solidified their invaluable contributions to the project's foundational development.

The hardware assembly phase of the project initiated with the procurement of necessary parts. Generous contributions from one team member aligned with the provided schematic. Additional parts were later sourced from specific vendors. The collective efforts of the team were crucial in translating the hardware design from concept to reality.

Risk management that you learned:

It is the process of locating, assessing, and mitigating project risks that could jeopardize the intended results. Throughout the course of a project, project managers are usually in charge of supervising the risk management procedure.
This project had a number of hazards, such as PCB etching, which, if done incorrectly, would have required us to buy a new one from a store, which would have been inconvenient. Drilling and soldering holes on PCBs is another.
We faced significant difficulties with PCB alignment because our PCB was double-layered. We had to make several of the components ourselves because Proteus did not have them.
Throughout the course of a project, project managers are usually in charge of supervising the risk management procedure.
This project had a number of hazards, such as PCB etching, which, if done incorrectly, would have required us to buy a new one from a store, which would have been inconvenient. Drilling and soldering PCB holes is another. We took all the necessary precautions to ensure that there would be no problems when employing acids.
We faced significant difficulties with PCB alignment because our PCB was double-layered. We had to make several of the components ourselves because Proteus did not have them.

Chapter 8 Feedback For Project and Course:

The culmination of this project brought forth a wealth of feedback and insights that greatly enriched our understanding of robotics engineering. Commencing with the meticulous examination of individual components, we learned the importance of thorough pre-integration testing. The discovery that certain components required replacement due to malfunctions, such as the motor driver, underscored the significance of real-world troubleshooting and the need for adaptable problem-solving skills. Moving to the testing phase on a PCB board, the seamless integration of components through meticulous soldering demonstrated the critical role of precision in hardware assembly.

While the majority of components performed optimally, the intricate interplay of sensors and the LCD highlighted the need for careful synchronization to ensure accurate real-time data display. However, a notable challenge emerged during the evaluation phase – the motor driver, although correctly receiving voltage, faced issues in mapping this input to the motors. This specific challenge became a focal point for concentrated efforts in refining the control and functionality of our robotic system.

Overall, this project served not only as a technical endeavor but also as a collective journey of collaborative problem-solving and continuous improvement. It illuminated the interconnected nature of hardware components, underscoring the importance of a holistic approach in designing and testing robotic systems. The experiences gained throughout this project will undoubtedly serve as a solid foundation for future ventures in the dynamic field of robotics and embedded systems.