REHBAR THE LINE FOLLOWING ROBOT

Group Members:

FAIZAN TAHIR (231778)

BILAL SHAUKAT (231863)

ABDUL REHMAN (231875)

BE MECHATRONICS (SESSION F23-A)

Project Supervisor

Sir. Umer Farooq

Lecturer

DEPARTMENT OF MECHATRONICS ENGINEERING

FACULTY OF ENGINEERING

AIR UNIVERSITY, ISLAMABAD

Table of Content

Chapter 1- Preliminaries

1.1 Proposal

1.2 Initial feasibility

1.3 Comparison table of similar products

1.4 Technical Standards

1.5 Team Roles & Details

1.6 Work Break down structure

1.7 Gantt Chart

1.8 Estimated budgets

Chapter 2-Project Conception

2.1 Introduction

2.2 Literature Review

2.3 List of features and operational specification of your project

2.4 Basic block diagrams of whole system and subcomponents in Draw.io or similar tool

Chapter 3- Mechanical Design

3.1 Platform Design

3.2 Material Selection and choices

3.3 3D CAD design `

Chapter 4- Electronics Design and Sensor Selections

4.1 Component Selection

4.2 Sensors along with specification and features

4.3 Schematic

4.4 PCB

Chapter 5- Software/Firmware Design

5.1 Input Output pinouts

5.2 Controller Selections with features

5.3 Software Design / Code

Chapter 1 – Preliminaries

1.1 Proposal

Project Title:

REHBAR the Line Following Robot

Objectives of this project:

To make such a robot that can:

1 Follow a line no matter its shape.

2 Detect object and can avoid them.

3 Detect color of the object.

4 Detect the changing voltage and current from the power source.

5 Store data into SD card.

6 Display voltage and current onto an OLED.

7 Send and receive data.

Brief Detail:

The robot will follow a black or white line and will record the data of how much it has travelled. This data will then be stored into a SD card and later on will be used by the robot to travel in reverse direction. It will be able to detect the color of the object placed in its path, if it is RED it will avoid it and will follow the line as usual. If the color is GREEN then it will stop. Voltage and current will be measured all the time coming from the battery. This data will be displayed on an OLED. All of the information regarding the path or voltage will then be stored into the cloud.

1.2 Initial Feasibility

At the starting stage this project is quite feasible.

- Robot's components are easily available in the market so it is **technically feasible**.
- Components are low cost so it is also **financially feasible**.
- It's design and integration is not that difficult so it is also **operationally feasible.**

1.3 Comparison table of similar products

Feature	Our Robot	Pololu 3pi+ Line Follower	Tamiya Line Tracer Kit
Microcontroller	Arduino Nano	ATmega328P	None (Analog Circuit)
Sensors (color, Ultrasonic, current, voltage)	Yes	No	No
Line Sensors	5-way IR Array	5 IR sensors	2 Photo-Reflectors
Reverse Feature	Yes	No	No
Object Avoidance	Yes	No	No
Motor Driver	TB6612FNG	Dual H-Bridge	Direct Control
Color Detection	Yes	No	No

1.4 Technical Standards

PCB Size	Less Than 150 x 150 mm
Chassis Size	Approx 300 x 200 mm
Motor Speed	Less Than 100 rpm
Line Width	2 to 3 cm
Motor Driver Operating Voltage	V_{CC} = 5V V_M = 12V

1.5 Team Roles and Details

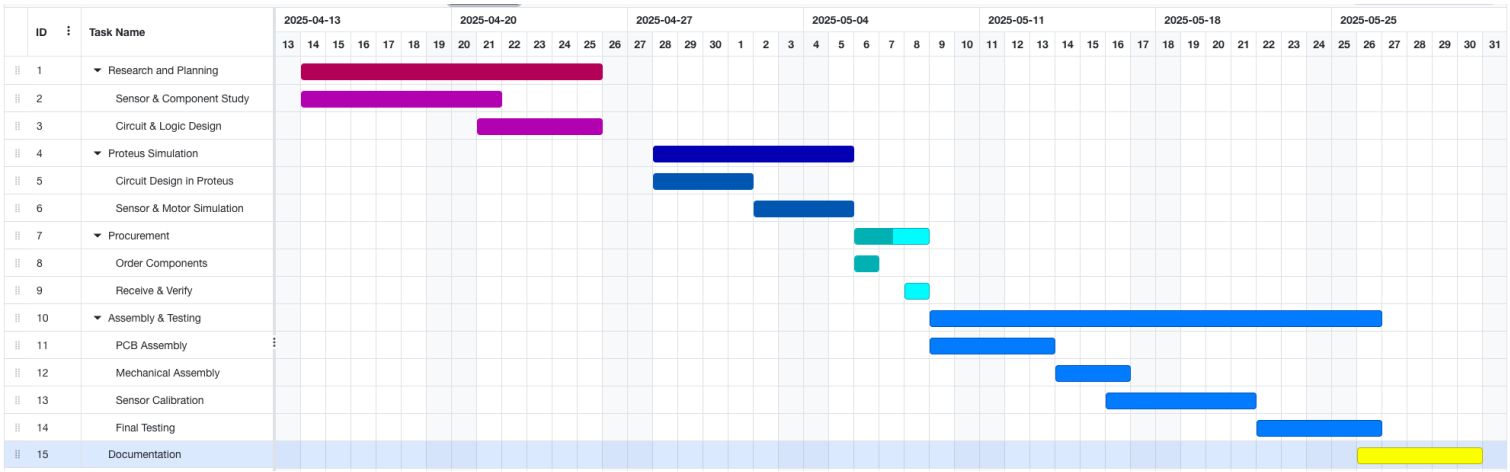
Member Names	Role	Detail
Faizan Tahir	Hardware Lead	Component Analysis Creating a schematic Creating a PCB Soldering the components Checking the functionality of PCB
Abdul Rehman	Integration and tester	Testing components Integrating with NANO Final testing of Robot
Bilal Shaukat	Coding	Simulation in Proteus Writing a test code for Proteus Writing final code for Robot

1.6 Work Break down Structure

Our Work will be done in the following manner:

- First of all, Faizan will do the market and component research and will tell Bilal of these components.
- Bilal will then create a simulation to check these components and operation with microcontroller and write a test code.
- Then we will buy these components and Abdul Rehman will test them on breadboard with specified pin numbers in the simulation.
- Abdul Rehman will then tell Faizan and he will create a Schematic and PCB for our Robot.
- Faizan will solder the components and our PCB will be ready.
- Bilal will write a fully functional code for the robot and Faizan and Abdul Rehman will do the final testing of the robot.

1.7 Gantt Chart



1.8 Estimated Budget

Components	Model	Quantity	Cost
Motor Driver	TB6612FNG	2	1200
ESP32	Esp 32 NODEMCU 38pin (CH9102x)	1	1300
Arduino	Arduino Nano	1	1000
OLED Display	SSD1106	1	750
Voltage Sensor	(0-25) Voltage Sensor	1	50
Current Sensor	ACS712	1	375
SD Card Module	NA	1	150
Header Pins	All Types	8 of each type	15*16
Ultrasonic Sensor	SR04	1	240
Motors with Wheels	DC motors	2	900
IR Sensor	TCRT5000	1	500
Encoders / RPM Sensor	H206 Tachometer	2	260
Battery	NA	6	2000
Jumper Wires	All Types	3 packs	480
Acrylic Body	NA	1	500
Total: 9795			

Chapter 2 – Project Conception

2.1 Introduction

REHBAR is a line following robot which will follow a black line (also white line if wanted). It has been integrated with different sensors that will work together to create a beautiful environment for the robot.

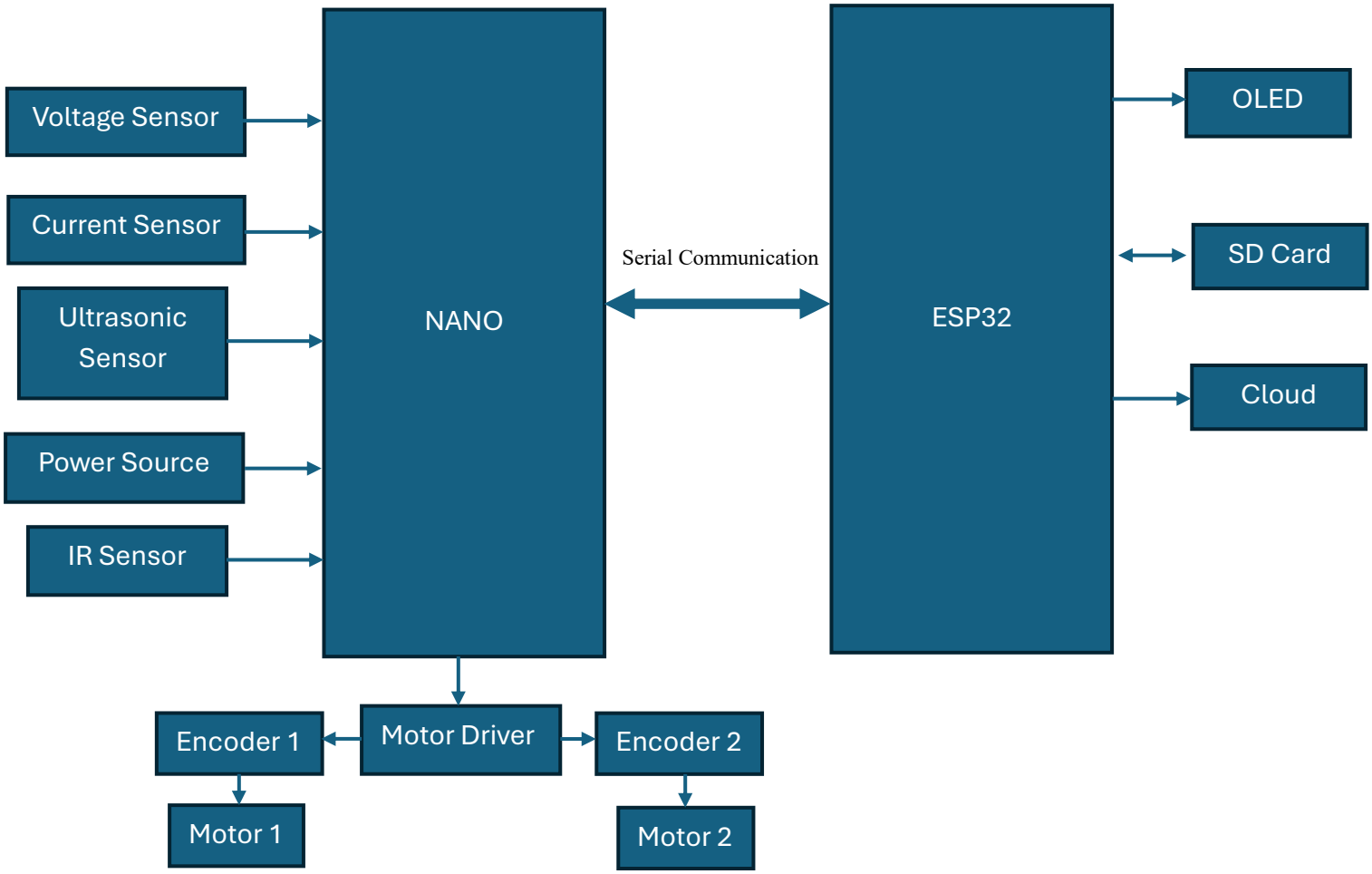
2.2 Literature Review

Topic	Research Paper	Citations
Line Follower	**"Autonomous Line Following Robot Using IR Sensors and Arduino"**	Sahu, A. Nema, A. Negi, A. Jaiswal, and U. S. Kurmi, "Line Following Robot Using Arduino," International Journal of Creative Research Thoughts (IJCRT), vol. 12, no. 6, Jun. 2024. [Online]. Available: https://www.ijcrt.org/papers/IJCRT2406111.pdf
Line Following Algorithm	**"Comparative Study of Line Following Algorithms for Robot Navigation"**	G. Dewantoro, J. Mansuri, and F. D. Setiaji, "Comparative Study of Computer Vision Based Line Followers Using Raspberry Pi and Jetson Nano," Journal of Robotics and Control (JRC), vol. 2, no. 2, pp. 54–60, 2021. [Online]. Available: https://www.researchgate.net/publication/357545530
Designing	**"Design and Implementation of a Smart Line Follower Robot for Industrial Applications"**	R. Farkh and K. Aljaloud, "Vision Navigation Based PID Control for Line Tracking Robot," Intelligent Automation & Soft Computing, vol. 35, no. 1, pp. 901–911, Jun. 2022. doi: 10.32604/iasc.2023.027614.

2.3 List of features and operational specification of your project

Feature	Specification
Line Detection	5-way IR Sensor Array
Chassis	Lightweight acrylic/plastic
Motor Driver	TB6612FNG (dual H-bridge)
Microcontroller	Arduino Nano
Power Supply	7.4V Li-ion (regulated to 5V)
Speed Control	PWM-based control
Reaction Time	< 300 ms
Max Speed	20 cm/s
Weight	Less Than 750g

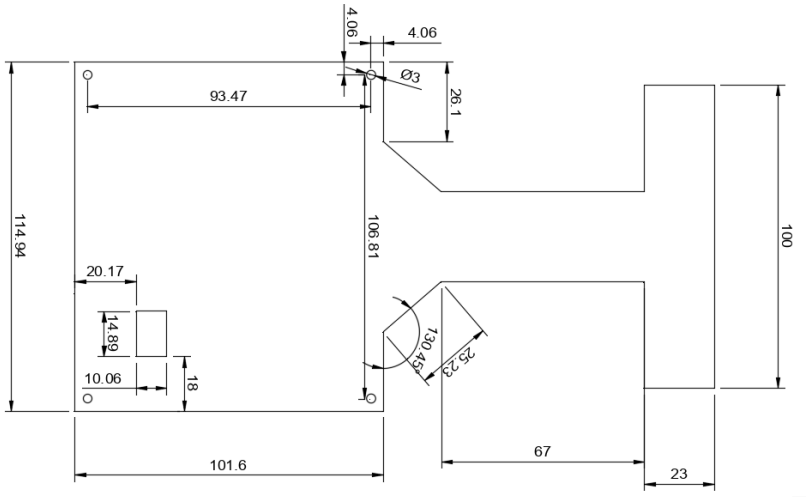
2.4 Block Diagram



Chapter 3 – Mechanical Design

3.1 Platform Design

We are considering making a T-shaped platform that can withstand the weight of PCB and all the other sensors un top of it. The platform will look something like this:

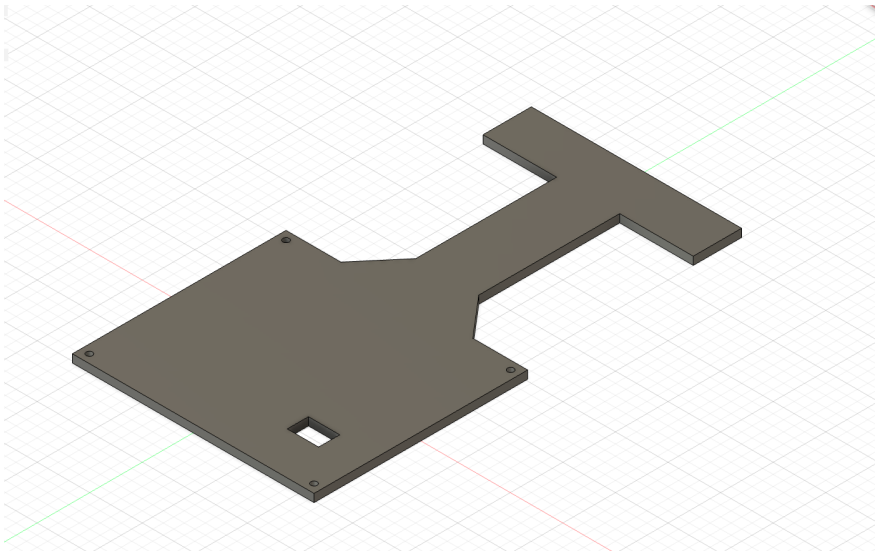


3.2 Material Selection and Choices

The platform will be made of acrylic sheet due to its strength to weight ratio and it is easy to build as it requires laser cutting. Acrylic is transparent, looks good, is light weight and will fir electronic components very nicely.

3.3 CAD Model

CAD model is designed in Fusion Software. It is given below:



Chapter 4 – Electronic Design

4.1 Component Selection

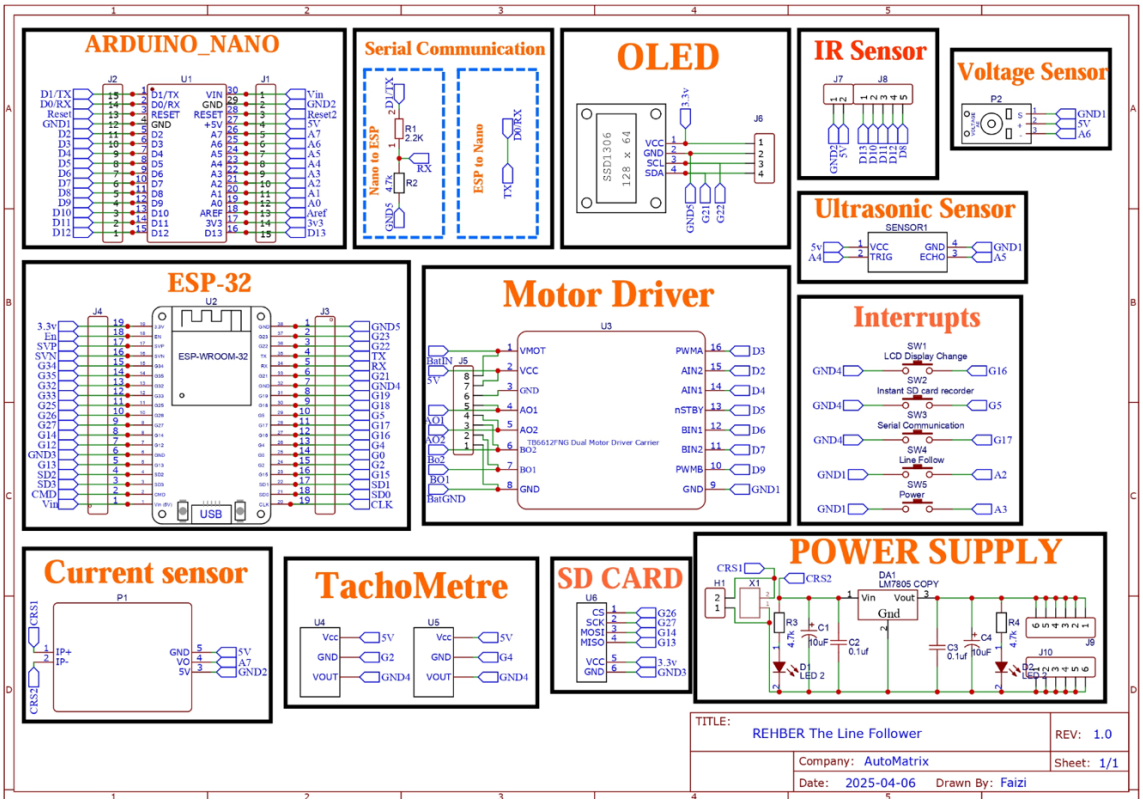
Component	Purpose
ESP32 (38 Pin, CH9102x)	Main controller for data processing and wireless communication; supports Wi-Fi and Bluetooth.
Arduino Nano	Acts as a secondary microcontroller to interface sensors and pass data to ESP32 via UART.
Motor Driver (TB6612FNG)	Dual H-bridge driver used to control the speed and direction of two DC motors.
DC Motors with Wheels	Provide the robot with locomotion; controlled via PWM signals.
TCRT5000 IR Sensors	Line detection sensors that help the robot follow a path based on contrast.
Ultrasonic Sensor (HC-SR04)	Used for obstacle detection by measuring distance using ultrasonic waves.
OLED Display (SSD1306)	Displays real-time data such as sensor values or battery status.
Voltage Sensor (0–25V)	Measures battery or supply voltage and sends analog voltage level to microcontroller.
Current Sensor (ACS712)	Measures current consumed by the system to monitor motor load or power draw.
H206 Tachometer (Encoder)	Used to detect RPM of DC motors, useful for speed control and feedback.

4.2 Sensors with Specifications

Sensor	Model	Specifications	Features
IR Sensor	TCRT5000	3.3V–5V, ~2–12mm range	Reflective IR for line tracking; fast response time
Ultrasonic Sensor	HC-SR04	5V, 2cm–400cm range, ~15° beam angle	Accurate distance measurement using ultrasonic waves
Voltage Sensor	0–25V Analog	Input: 0–25V, Output: 0–5V scaled	Measures supply/battery voltage; simple 2-pin analog output
Current Sensor	ACS712	Rated: 5A / 20A / 30A (your choice), Analog output	Hall-effect based; high accuracy current sensing
RPM Sensor	H 206 Tachometer	Output: digital pulses, Speed: 0–6000 RPM	Measures wheel rotation using IR reflection

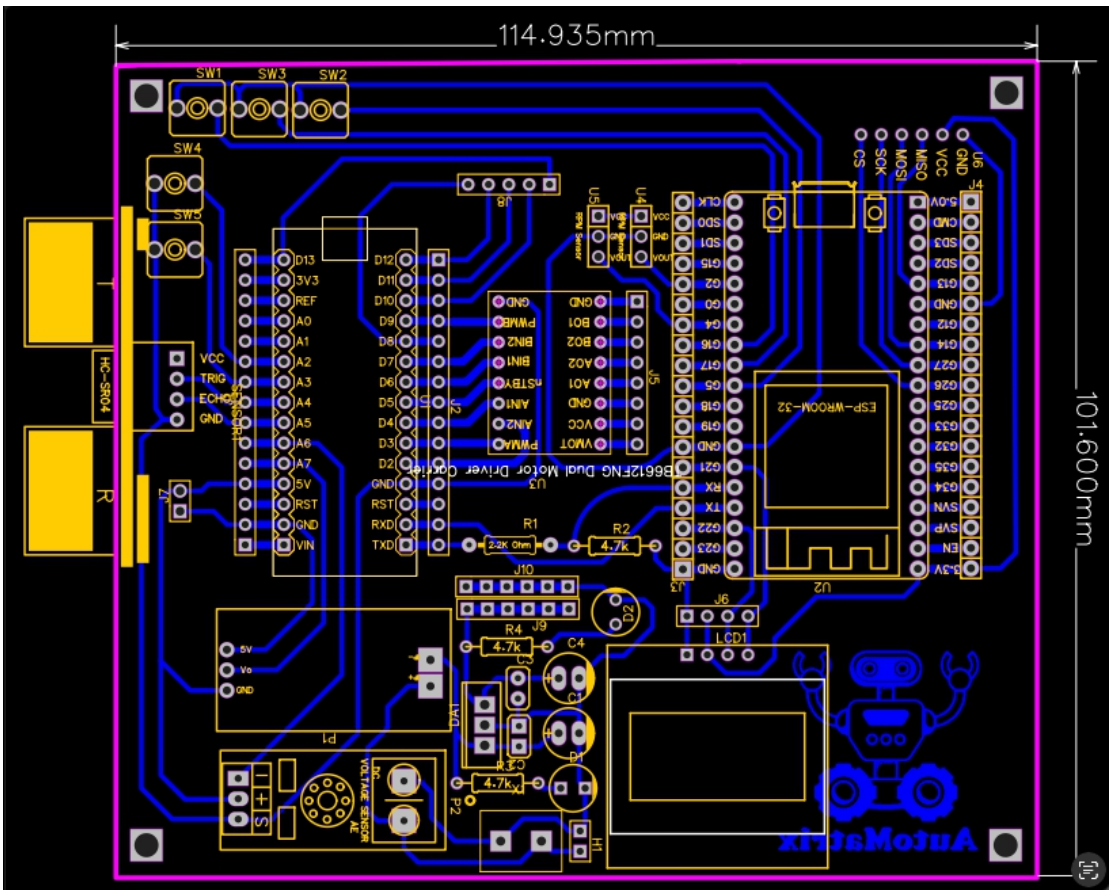
4.3 Schematic

Schematic is built in Easy EDA. Here it is the schematic of our LFR:



4.4 PCB

Here is the PCB:



CHAPTER 5 – Software Design

5.1 Input Output pinouts

NANO Pinout:

Component	Function	Arduino Nano Pin
Motor Driver (TB6612FNG)		
Standby Pin	STBY	D5
Left Motor PWM	PWMA	D3 (PWM)
Left Motor AIN1	Direction Control	D4
Left Motor AIN2	Direction Control	D2
Right Motor PWM	PWMB	D9 (PWM)
Right Motor BIN1	Direction Control	D6
Right Motor BIN2	Direction Control	D7
IR Sensors (5-way)	Line Following Inputs	D13, D12, D11, D10, D8
Ultrasonic Sensor	Trigger	A4
	Echo	A5
Voltage Sensor	Analog Voltage Reading	A6
Current Sensor	Analog Current Reading	A7
Serial Communication	Debug via Serial Monitor	TX (D1), RX (D0)
Power Supply	5V / GND	5V, GND

ESP Pinout:

Component	Function	ESP32 Pin
OLED		
	SCL	G22
	SDA	G21
SD Card Module		
	CS	G26
	SCK	G27
	MOSI	13
	MISO	14
RPM Sensor		
	V_{OUT1}	2
	V_{OUT2}	4

5.2 Controller Features

NANO:

- Due to its **compact size,** it fits easily into small robotics or embedded projects.
- It is **very easy to use** very beginner-friendly with the Arduino IDE.
- It has **sufficient I/O Pins** that are enough GPIOs for controlling motors and reading IR sensors.
- It has **analog Inputs (A6, A7)** that are useful for voltage and current sensing.

ESP:

- It has **dual core 32-bit** MCU, faster processing, suitable for real-time tasks like hand-tracking.
- **Wi-Fi & Bluetooth Built-In** which are essential for wireless communication or IoT control.
- **More GPIOs and Peripherals** which are better expansion capability for sensors, displays, etc.

5.3 Software Design / Code

NANO Code:

Pins Initialized:

```
#define BASE_SPEED 105

// Motor A pins
int PWM_A = 3;
int AIN1 = 4;
int AIN2 = 2;

// Motor B pins
int PWM_B = 9;
int BIN1 = 6;
int BIN2 = 7;

// STBY pin
int STBY = 5;

// IR sensor pins
int sensors[] = {8, 12, 11, 10, 13};

// Ultrasonic sensor pins
int trigPin = A4;
int echoPin = A5;

// Voltage and current sensor pins
int voltagePin = A6;
int currentPin = A7;

```

IR Sensor Functions:

```
// Movement functions
void moveForward() {
  analogWrite(PWM_A, BASE_SPEED);
  analogWrite(PWM_B, BASE_SPEED);
  digitalWrite(AIN1, HIGH);
  digitalWrite(AIN2, LOW);
  digitalWrite(BIN1, HIGH);
  digitalWrite(BIN2, LOW);
}
```

```
void turnSlightLeft() {
  analogWrite(PWM_A, BASE_SPEED / 2);
  analogWrite(PWM_B, BASE_SPEED);
  digitalWrite(AIN1, HIGH);
  digitalWrite(AIN2, LOW);
  digitalWrite(BIN1, HIGH);
  digitalWrite(BIN2, LOW);
}
```

```
void turnSlightRight() {
  analogWrite(PWM_A, BASE_SPEED);
  analogWrite(PWM_B, BASE_SPEED / 2);
  digitalWrite(AIN1, HIGH);
  digitalWrite(AIN2, LOW);
  digitalWrite(BIN1, HIGH);
  digitalWrite(BIN2, LOW);
}
```

```
void sharpTurnLeft() {
  analogWrite(PWM_A, BASE_SPEED / 2);
  analogWrite(PWM_B, BASE_SPEED);
  digitalWrite(AIN1, HIGH);
  digitalWrite(AIN2, LOW);
  digitalWrite(BIN1, HIGH);
  digitalWrite(BIN2, LOW);
}
```

```
void sharpTurnRight() {
  analogWrite(PWM_A, BASE_SPEED);
  analogWrite(PWM_B, BASE_SPEED / 2);
  digitalWrite(AIN1, HIGH);
  digitalWrite(AIN2, LOW);
  digitalWrite(BIN1, HIGH);
  digitalWrite(BIN2, LOW);
}
```

```
void stopMotors() {
  analogWrite(PWM_A, 0);
  analogWrite(PWM_B, 0);
  digitalWrite(AIN1, LOW);
  digitalWrite(AIN2, LOW);
  digitalWrite(BIN1, LOW);
  digitalWrite(BIN2, LOW);
}
```

Ultrasonic Sensor:

Obstacle Detection

```
156  // Obstacle detection
157  long getDistance() {
158    digitalWrite(trigPin, LOW);
159    delayMicroseconds(2);
160    digitalWrite(trigPin, HIGH);
161    delayMicroseconds(10);
162    digitalWrite(trigPin, LOW);
163    return pulseIn(echoPin, HIGH) * 0.034 / 2;
164  }
```

Obstacle Avoidance

```
166  // Avoid obstacle
167  void avoidObstacle() {
168    stopMotors();
169    delay(200);
170
171    // Move away from line
172    analogWrite(PWM_A, BASE_SPEED);
173    analogWrite(PWM_B, BASE_SPEED);
174    digitalWrite(AIN1, LOW);
175    digitalWrite(AIN2, HIGH);
176    digitalWrite(BIN1, HIGH);
177    digitalWrite(BIN2, LOW);
178    delay(500);
179
180    stopMotors();
181    delay(300);
```

```
183    // Move back to line
184    analogWrite(PWM_A, BASE_SPEED);
185    analogWrite(PWM_B, BASE_SPEED);
186    digitalWrite(AIN1, HIGH);
187    digitalWrite(AIN2, LOW);
188    digitalWrite(BIN1, LOW);
189    digitalWrite(BIN2, HIGH);
190    delay(500);
191
192    stopMotors();
193    delay(300);
194  }
```

Voltage and Current Sensor:

```
196  // Voltage & Current Monitoring
197  void checkVoltageAndCurrent() {
198    int voltageRaw = analogRead(voltagePin);
199    int currentRaw = analogRead(currentPin);
200
201    float voltage = voltageRaw * (5.0 / 1023.0) * (11.0); // Assuming 10k:1k divider
202    float current = (currentRaw - 512) * (5.0 / 1023.0) / 0.185; // For ACS712 5A sensor
203
204    Serial.print("Voltage: ");
205    Serial.print(voltage, 2);
206    Serial.print(" V, Current: ");
207    Serial.print(current, 2);
208    Serial.println(" A");
209  }
```

Void Setup:

```
27  void setup() {
28    Serial.begin(9600);
29
30    // Motor pins
31    pinMode(PWM_A, OUTPUT);
32    pinMode(AIN1, OUTPUT);
33    pinMode(AIN2, OUTPUT);
34    pinMode(PWM_B, OUTPUT);
35    pinMode(BIN1, OUTPUT);
36    pinMode(BIN2, OUTPUT);
37    pinMode(STBY, OUTPUT);
38    digitalWrite(STBY, HIGH); // enable motor driver
39
40    // IR sensors
41    for (int i = 0; i < 5; i++) {
42      pinMode(sensors[i], INPUT);
43    }
44
45    // Ultrasonic pins
46    pinMode(trigPin, OUTPUT);
47    pinMode(echoPin, INPUT);
48
49    stopMotors();
50  }
```

Void Loop:

```
52  void loop() {
53    checkVoltageAndCurrent();
54
55    long distance = getDistance();
56
57    if (distance < 10) {
58      avoidObstacle();
59      return;
60    }
61
62    int sensorValues[5];
63    for (int i = 0; i < 5; i++) {
64      sensorValues[i] = !digitalRead(sensors[i]);
65    }
66
67    int s1 = sensorValues[0];
68    int s2 = sensorValues[1];
69    int s3 = sensorValues[2];
70    int s4 = sensorValues[3];
71    int s5 = sensorValues[4];
```

```
73    // Line following logic
74    if (s1 && s2 && s3 && s4 && s5) {
75      moveForward();
76    } else if (s3 && !s2 && !s4) {
77      moveForward();
78    } else if (s2 && !s3 && !s4) {
79      turnSlightLeft();
80    } else if (s4 && !s3 && !s2) {
81      turnSlightRight();
82    } else if (s1 && !s2 && !s3 && !s4 && !s5) {
83      sharpTurnLeft();
84    } else if (s5 && !s2 && !s3 && !s4 && !s1) {
85      sharpTurnRight();
86    } else if (s2 && s3 && !s4 && !s1 && !s5) {
87      turnSlightLeft();
88    } else if (s3 && s4 && !s2 && !s1 && !s5) {
89      turnSlightRight();
90    } else if (s1 && s2 && !s3 && !s4 && !s5) {
91      sharpTurnLeft();
92    } else if (s4 && s5 && !s1 && !s2 && !s3) {
93      sharpTurnRight();
94    } else if (s2 && s3 && s4) {
95      moveForward();
96    } else {
97      stopMotors();
98    }
```

ESP Code:

Pins Initialed:

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <U8g2lib.h>

// SD card SPI pins
#define SCK_PIN   27
#define MISO_PIN  13
#define MOSI_PIN  14
#define CS_PIN    26

// OLED I2C pins
#define OLED_SCL 22
#define OLED_SDA 21

// Tachometer input pins
#define TACHO1_PIN 4
#define TACHO2_PIN 2
```

OLED Code:

```
// OLED object (no reset pin)
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE, OLED_SCL, OLED_SDA);
```

```
// Initialize OLED
u8g2.begin();
u8g2.setFont(u8g2_font_ncenB08_tr);
u8g2.clearBuffer();
u8g2.drawStr(0, 15, "Starting...");
u8g2.sendBuffer();
```

Tacometer Code:

```
// Tachometer pulse counters
volatile unsigned long pulseCount1 = 0;
volatile unsigned long pulseCount2 = 0;

unsigned long lastRPMCalcTime = 0;
unsigned long rpm1 = 0;
unsigned long rpm2 = 0;
```

```
31  void IRAM_ATTR countPulse1() {
32    pulseCount1++;
33  }
34
35  void IRAM_ATTR countPulse2() {
36    pulseCount2++;
37  }
```

```
50  // Setup tachometer pins with interrupts
51  pinMode(TACHO1_PIN, INPUT_PULLUP);
52  pinMode(TACHO2_PIN, INPUT_PULLUP);
53  attachInterrupt(digitalPinToInterrupt(TACHO1_PIN), countPulse1, RISING);
54  attachInterrupt(digitalPinToInterrupt(TACHO2_PIN), countPulse2, RISING);
55
```

SD Card Code:

```
56  // Initialize SD card
57  SPI.begin(SCK_PIN, MISO_PIN, MOSI_PIN, CS_PIN);
58  SPI.setFrequency(1000000);
59
60  Serial.println("Initializing SD card...");
61  if (!SD.begin(CS_PIN)) {
62    Serial.println("SD Init Failed");
63    u8g2.clearBuffer();
64    u8g2.drawStr(0, 15, "SD Init Failed!");
65    u8g2.sendBuffer();
66    return;
67  }
68
69  Serial.println("SD Init OK");
70  u8g2.clearBuffer();
71  u8g2.drawStr(0, 15, "SD Init OK");
72  u8g2.sendBuffer();
73  }
74
```

Void Loop:

```
void loop() {
  // Calculate RPM every second
  if (millis() - lastRPMCalcTime >= 1000) {
    noInterrupts();
    rpm1 = pulseCount1 * 60;  // 1 pulse = 1 rotation
    rpm2 = pulseCount2 * 60;
    pulseCount1 = 0;
    pulseCount2 = 0;
    interrupts();
    lastRPMCalcTime = millis();

    // Print to serial
    Serial.printf("Tacho1 RPM: %lu\tTacho2 RPM: %lu\n", rpm1, rpm2);

    // Update OLED
    u8g2.clearBuffer();
    char buf1[24], buf2[24];
    sprintf(buf1, "Tacho1 RPM: %lu", rpm1);
    sprintf(buf2, "Tacho2 RPM: %lu", rpm2);
    u8g2.drawStr(0, 15, buf1);
    u8g2.drawStr(0, 35, buf2);
    u8g2.sendBuffer();
  }

  delay(100); // Light delay for responsiveness
}
```

Conclusion

- The Line Following Robot (LFR) project was a successful combination of mechanical and electronic systems working together. We used important components like the ESP32, Arduino Nano, motor driver, IR sensors, OLED display, and other sensors to build a robot that can follow a black line on a surface and check its voltage and current levels.
- The robot's body was made using a lightweight and stable design, and the electronic parts were chosen carefully to make sure everything works smoothly. The sensors help the robot detect the line and avoid obstacles, while the display shows useful information.
- This project helped us learn how to bring together hardware and software to solve a real problem. In the future, we can add more features like wireless control, better obstacle avoidance, or automatic speed control.
- Overall, this project gave us hands-on experience in building and programming a real robotic system.