# LINE FOLLOWER ROBOT

*Group Members:*

**ANAB SHAZIL (221682)**

**M. ABDULLAH KHAN (221748)**

**BE MECHATRONICS (Session Spring 2023)**

**Project Supervisor**

**Eng. Umer Farooq**

**Lecturer**

**And**

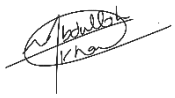
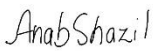
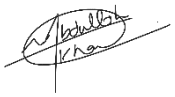
**Engr. Sadia Saeed**

**Lab Engineer**

**DEPARTMENT OF MECHATRONICS ENGINEERING**

**FACULTY OF ENGINEERING**

**AIR UNIVERSITY, ISLAMABAD**

| Team Member Name | Role | Word count that each has written in assigned colour code in report | Signature |
|---|---|---|---|
| Team Lead | Muhammad Abdullah Khan | 830 |  |
| Project Manager | Anab Shazil | 3000 |  |
| Technical | Muhammad Abdullah Khan | - |  |
| Integrator and Tester | Anab/Abdullah | - |  |

| Name | Roll No. | Colour Code |
|---|---|---|
| Anab Shazil | 221682 | Yellow |
| Muhamad Abdullah Khan | 221748 | Red |

# INDEX

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

7.2 Comment individually about success /failure of your project

7.3 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

# CHAPTER 1 PRELIMINARIES

## 1.1 Proposal:

The objective was to construct a line follower robot according to the guidelines provided Line follower robot is one kind of autonomous robot which follows a line until that line exists. Generally, the line is drawn on the floor. It can be either black or white. The line can also be normal visible color or invisible magnetic field or electric field. The robot follows the line by using Infra-Red Ray (IR) sensors. There are five IR sensors which makes it an IR sensor array. These sensors read the line and send that reading to Arduino and then control the robot movement. In this paper, the authors will explain about the robot design, implementation, coding, testing, problems they faced and their solutions.

## 1.2 Initial Feasibility:

The initial phase of the project was divided in to two parts first we understood the turtle robot simulation and understood it.

For the part two of this phase we designed a basic line follower design using 3 IR sensors 2 motors LCD and L3298 motor driver this basic simulation helps us to understand the core working of the line follower robot we won't be adding complete simulation specifications for the scope of this report as that was only used as a medium to develop a better understanding

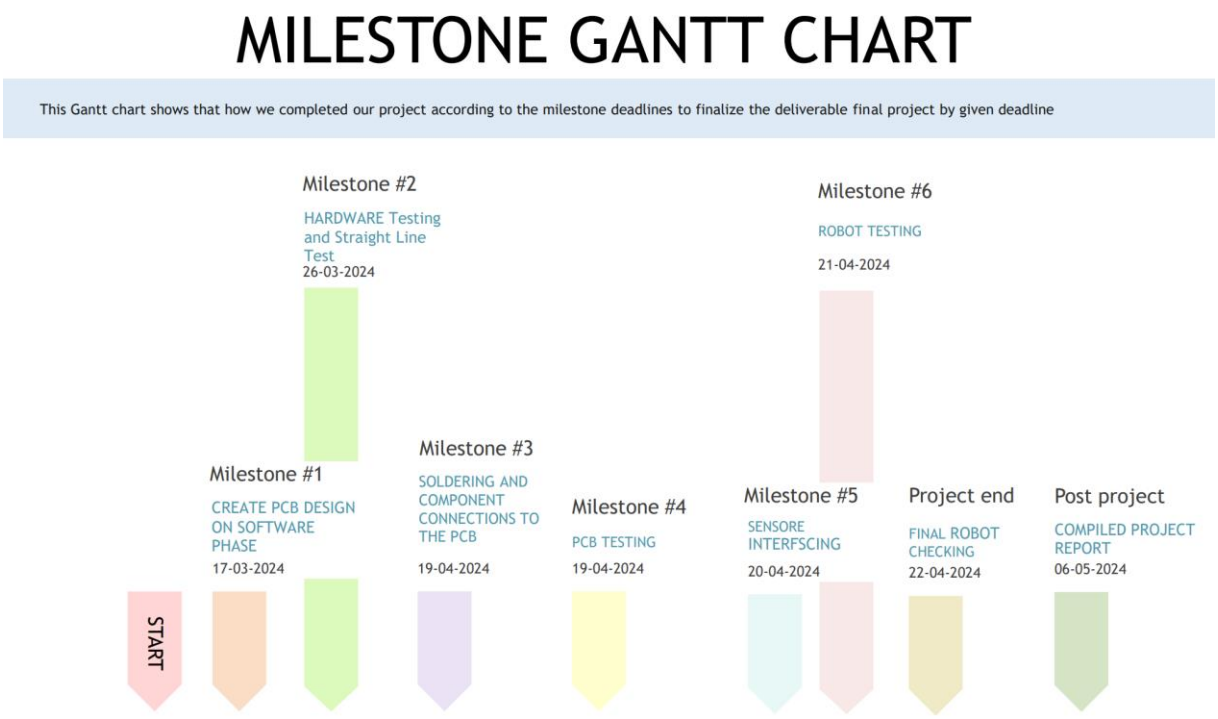
## 1.3 Specifications of Deliverables:

1. You must design custom ATMEL controller-based board by yourself for this project.
2. Simulation should be done on Proteus and also design PCB in the same software with linked schematic
3. Robotic Base of any suitable structure can be bought from market (Acrylic with 2 or 4 motors)
4. Dual channel H bridges can be used as modules or can be customized and designed by you
5. It should operate on battery for at least 20 Min (preferably you can use Li-Po/Li-Ion batteries with at least 2000mah capacity or you can use your own power supplies for powering the bot).
6. Micro SD card module must be used to store the information for logging.
7. IR, Ultrasonic, Colour Sensors can be used to detect red colored Obstacle.
8. On Embedded side we must use I/O, ADC, Timers, Interrupts, PWM in REHBER
9. Brain. Robot should send data wirelessly to on a remote computer using wireless module.
10. A report describing your effort Block diagram with pins /Algorithms/State machine, Schematics, component list, Bill of Material, Limitation, future works, references (Sample will be given) should be submitted on GCR along with individual video presentation that should include the video of your project following the line at the end of the presentation.

## 1.4 Team role:

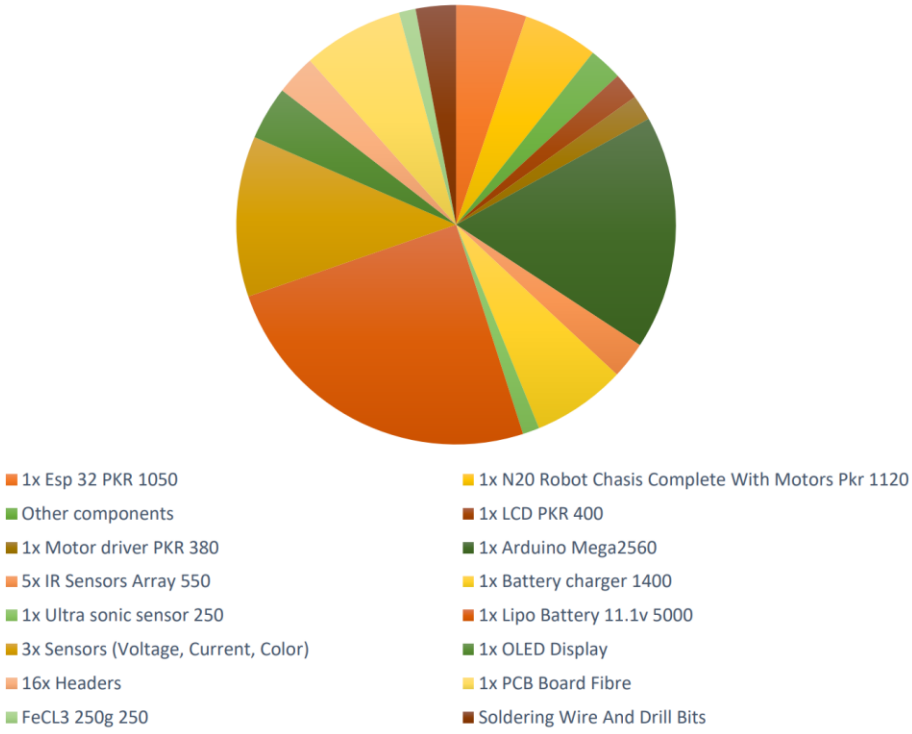
| Name | Roll number | Role |
|---|---|---|
| Anab Shazil | 221682 | Report/Software |
| Muhammad Abdullah Khan | 221748 | Hardware/Code |

## 1.5 Gantt Chart:




## 1.6 Estimated budget:

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




# CHAPTER 2
# PROJECT CONCEPTION

## 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 manuvering 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 [1].

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.

## 2.2 List of features and operational specification of our project

### Features:

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

- 5 IR sensors are used to detect white line with great precision thus the angles on path such as a 90 turn and 45 turn is also 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 and control the robot via Bluetooth and Wi-Fi connectivity
- Finally, the project will also be displaying outputs as LCD panel is used for this purpose

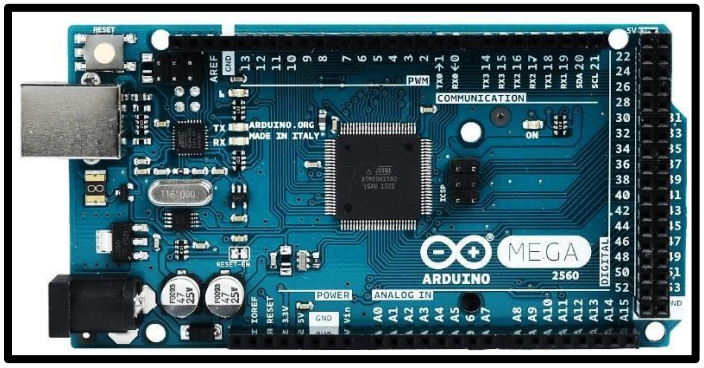
### Operation:

The basic operations of line follower are as follows:

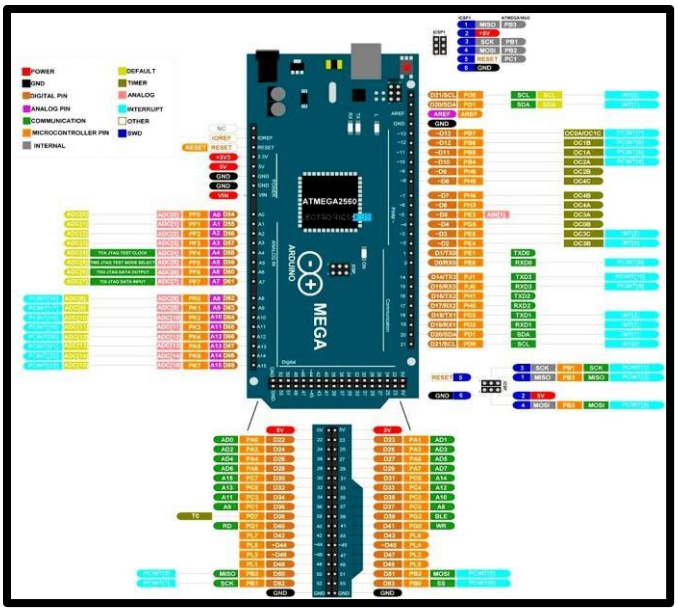
- Capture line position with optical sensors mounted at front end of the robot. For this a combination of IR-LED and Photodiode called an optical sensor has been used.
- This make sensing process of high resolution and high robustness.
- Steer robot requires steering mechanism for tracking. Two motors 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:

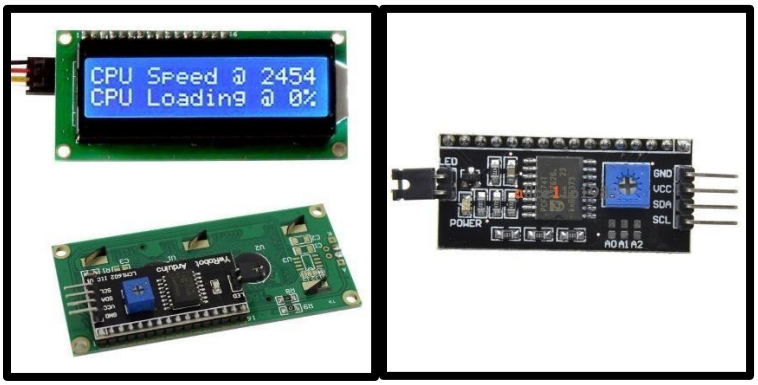
- 1x ATMEGA 2560



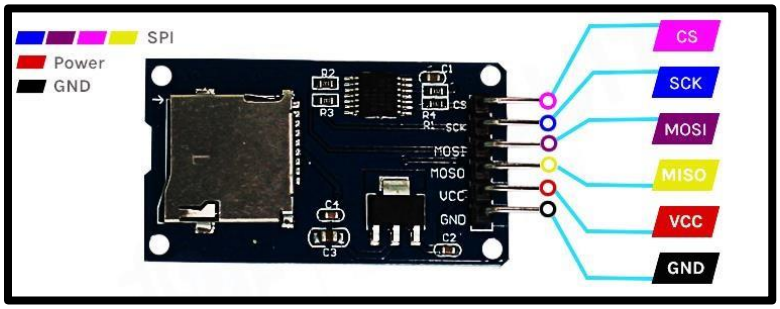

- Pin Configuration:




- 1x ESP 32
- 1x LCD

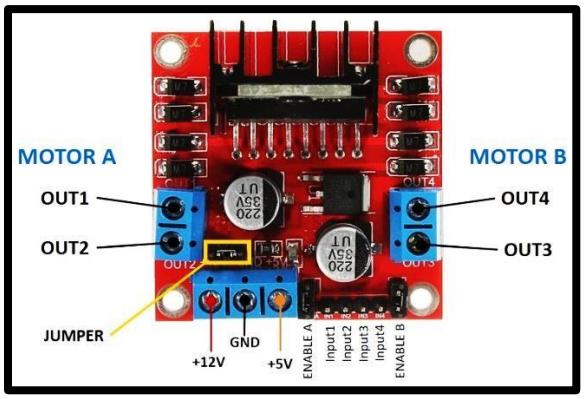



- 1x SD Card Module

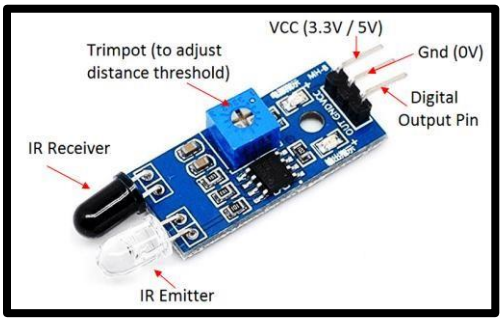



- 1x Motor Driver LM298

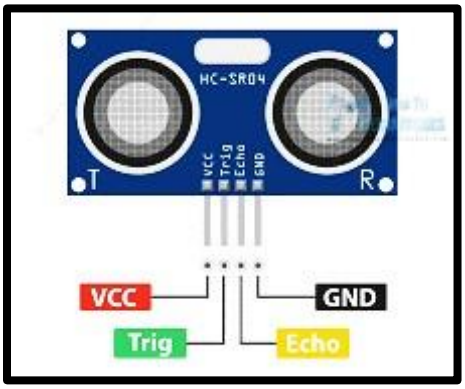
- 5x IR Sensors




1x Ultrasonic Sensor
1x Encoder

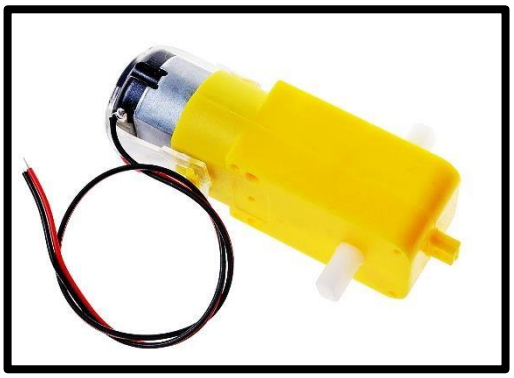



1x Robot Chassis

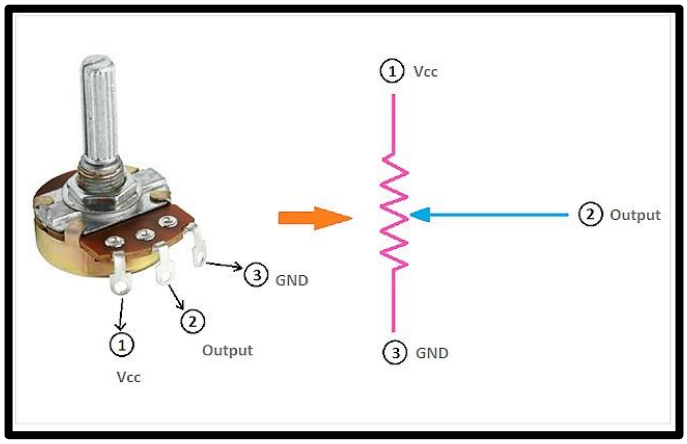



Screws and Nuts

- 2x Motors 3V to 6V



- Potentiometer



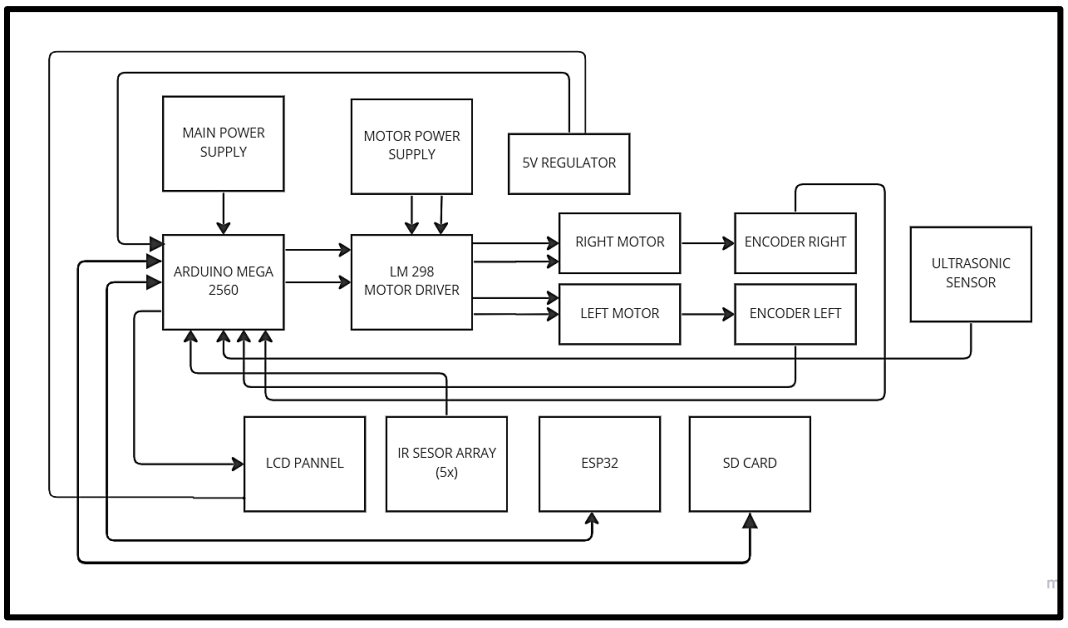

- Jumper Wires
- Header Pins
- Resistors Capacitors
- On/Off Button
- Battery

## 2.3 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.

## 2.4 Block diagrams:

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




## 2.5 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.

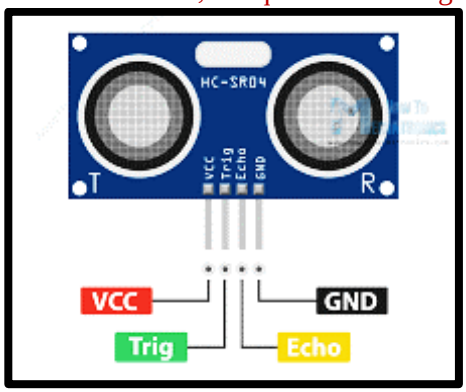



*Figure 7*

**Pin Configuration:**

*Table 3*

| 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 mega 2560:**

The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

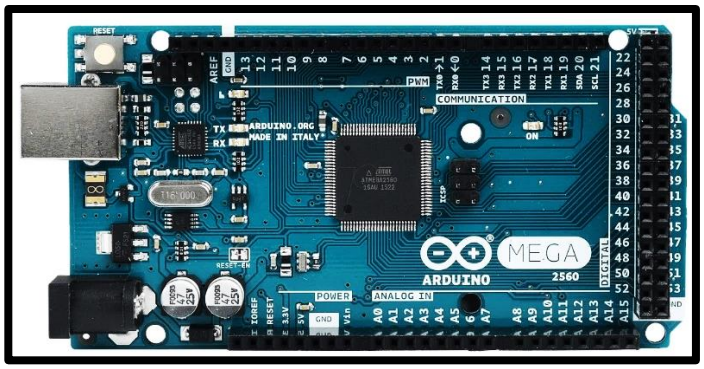



*Figure 8*

## Pin Configuration:

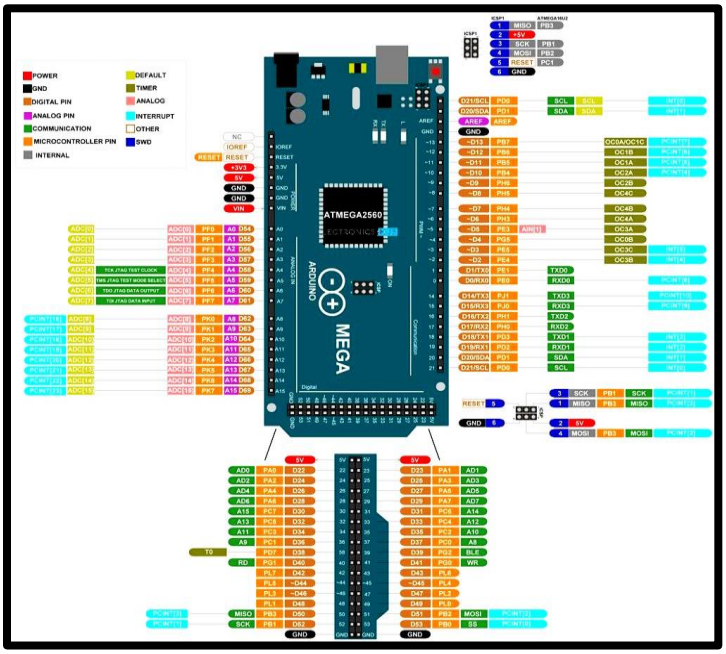



*Figure 9*

## Robotic Chassis (2 Wheel with DC Motor):

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

## Package Contains:

1. 1 x Rubber wires

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

*Figure 10*

## Ir sensor:

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.

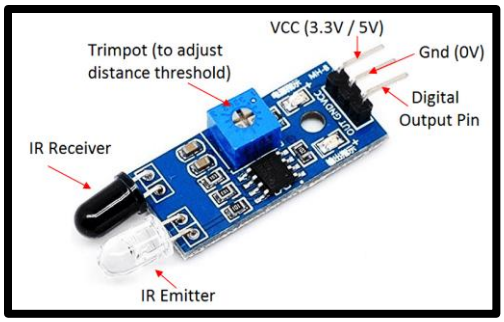



*Figure 11*

## 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.

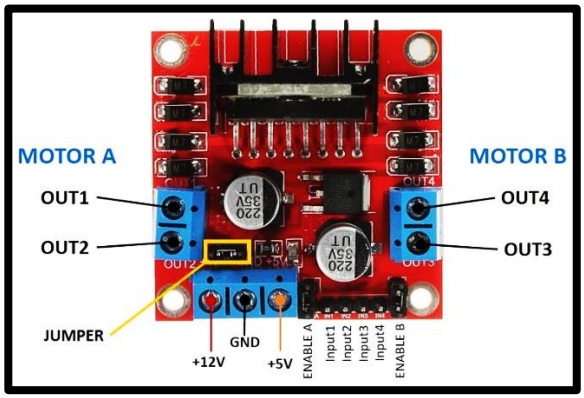



*Figure 12*

## 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.

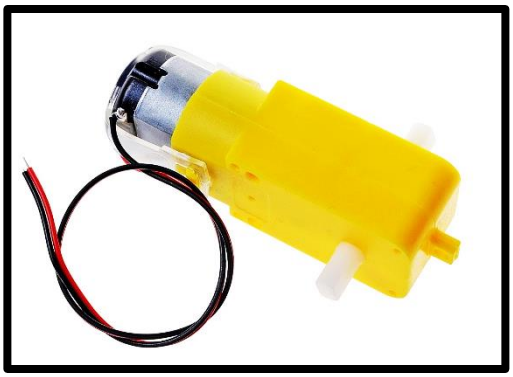


*Figure 13*

**Lm393 IR sensor**

LM393 Motor Speed Measuring Sensor Module For Arduino, the major goal is to check the rate of an electric motor. The module can be used in association with a microcontroller for motor speed detection, pulse count, position limit, etc. LM393 Motor Speed Measuring Sensor Module widely used in motor speed detection, pulse count, the position limit, etc. The DO output interface can be directly connected to a micro-controller IO port, if there is a block detection sensor, such as the speed of the motor encoder can detect.This is a thorough LM393 Motor Speed Measuring Sensor Module For Arduino.

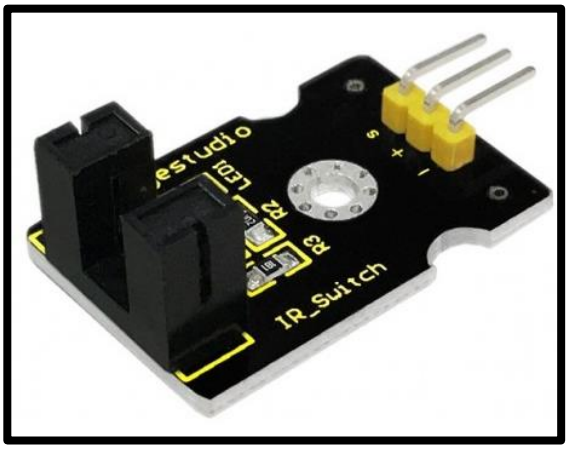



*Figure 14*

**Rotary incremental encoder**

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

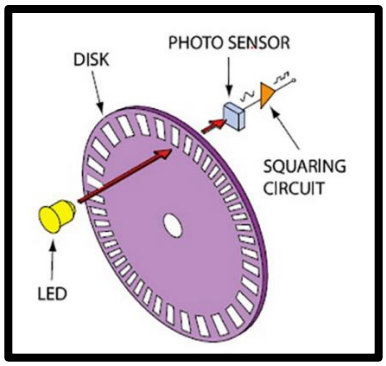



*Figure 15*

**SD Card Module**

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.

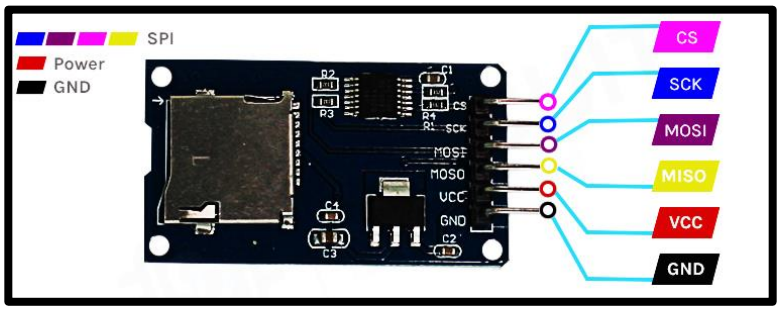



*Figure 16*

**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.

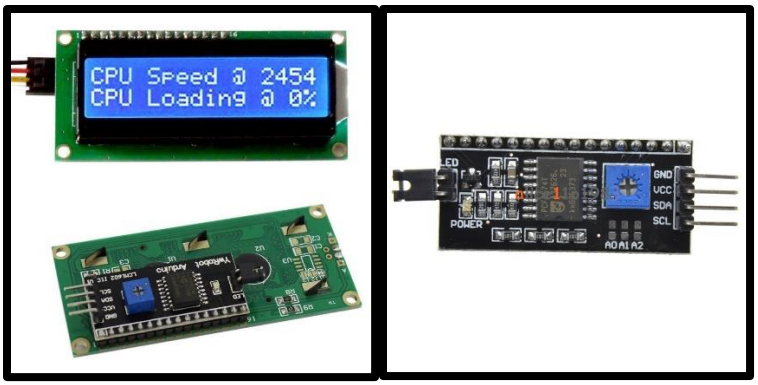



*Figure 17*

**LCD Pin Configuration:**

*Table 4*

| 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:**

*Table 5*

| 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.

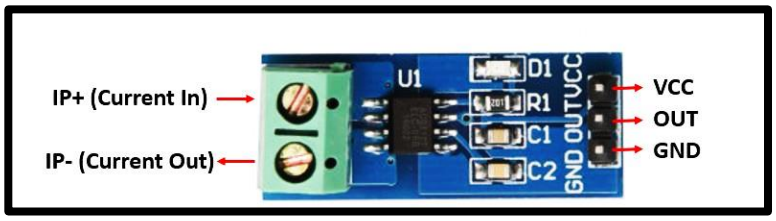



*Figure 18*

## VOLTAGE SENSOR

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

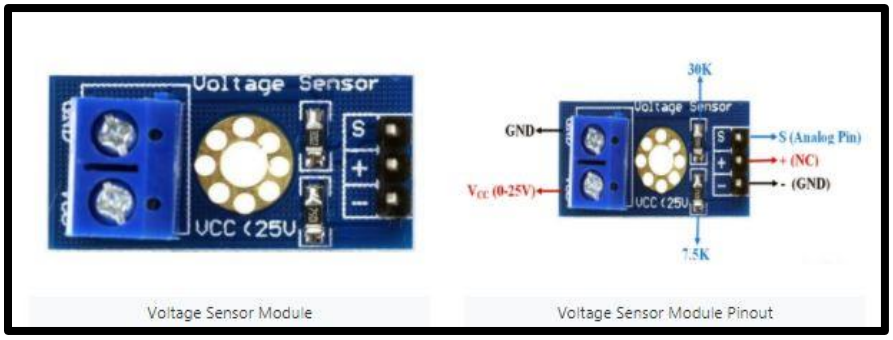



*Figure 19*

**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.

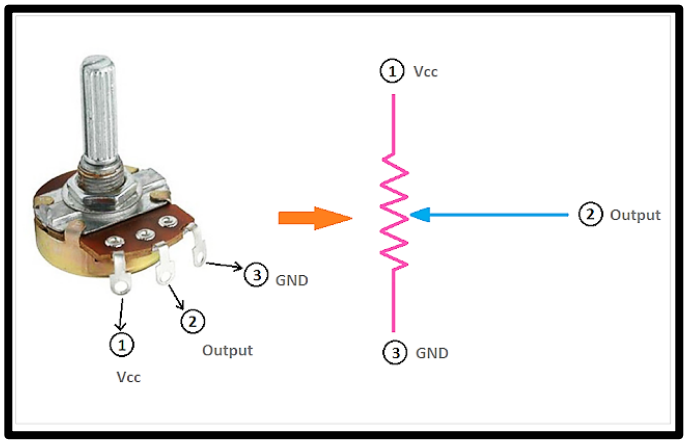



*Figure 20*

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.

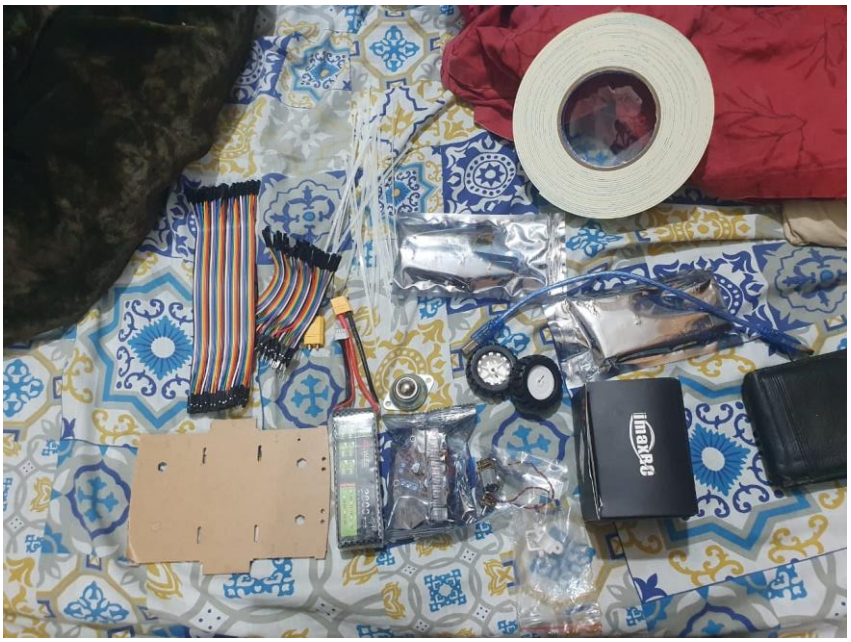
GitHub - MertArduino/Arduino-Line-Following-Robot
How to make and programme the fastest line following robot - Quora

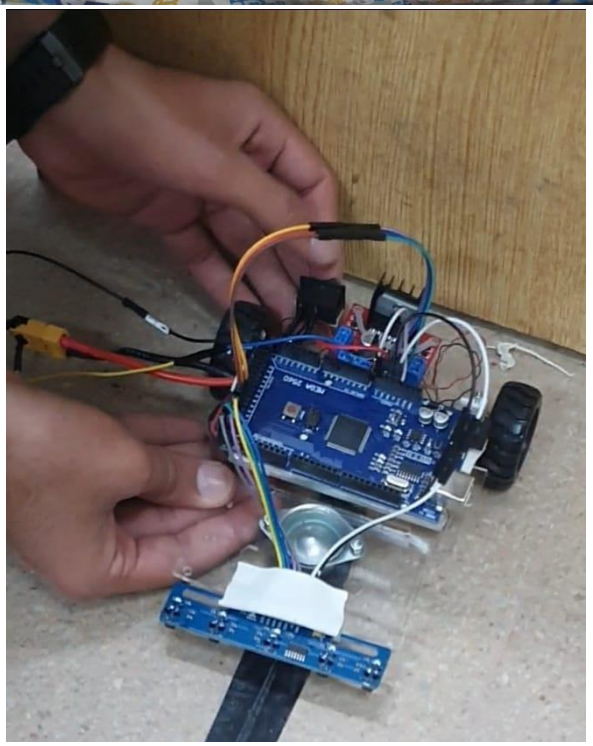
# CHAPTER 3 MECHANICAL DESIGN

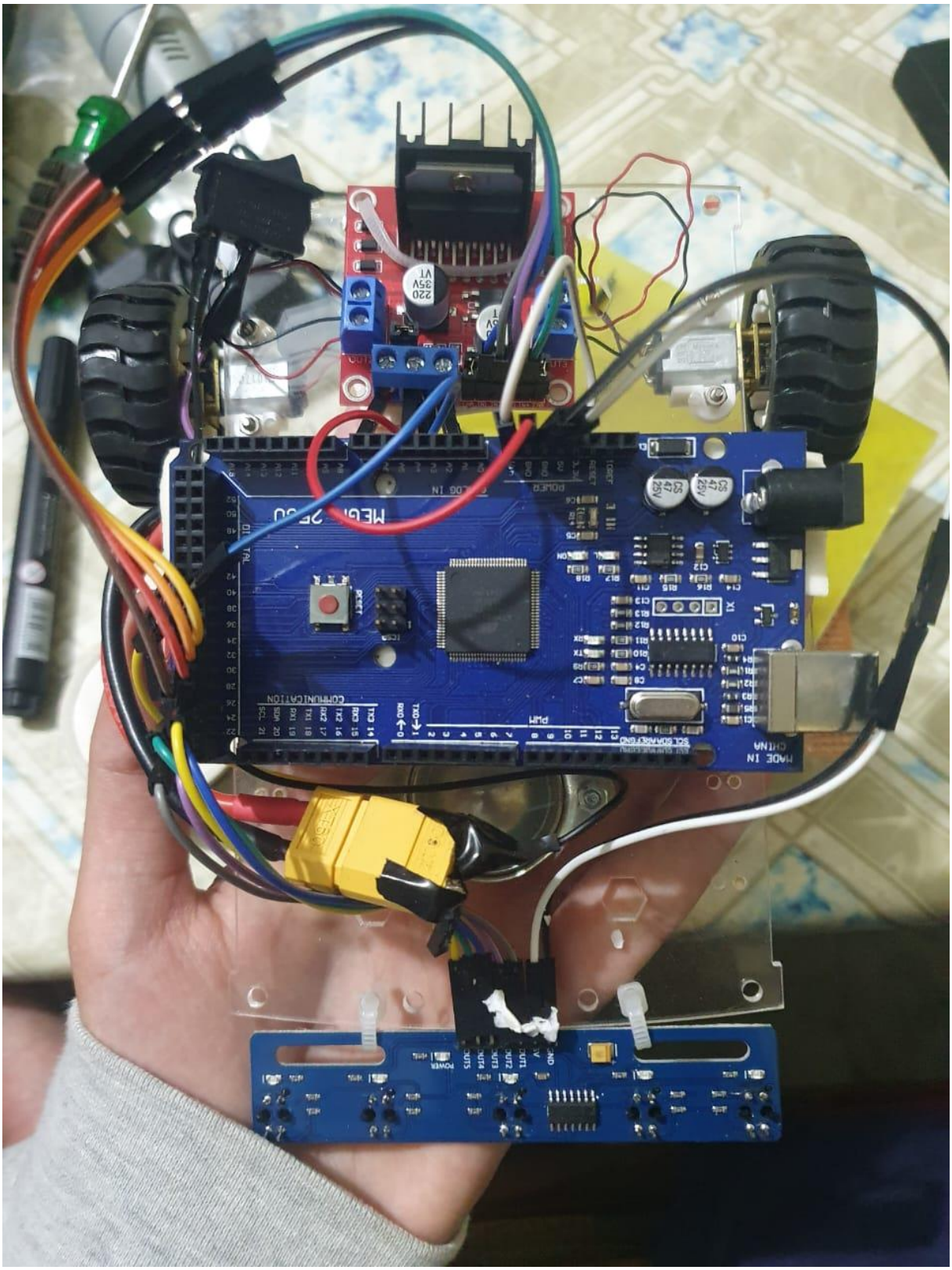
Mechanical design plays very important rule in any project of engineering. Our line following robot is an engineering project that can be affected by a lame or bad model of mechanical design. We can say that the mechanical design is responsible for the sustainability, weight lifting, long lasting life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

## 3.1. Mechanism selection and Platform Design

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






220
35V
COMMUNICATION
PWM
MADE IN CHINA

## 3.2 Actuators with speciation and datasheet

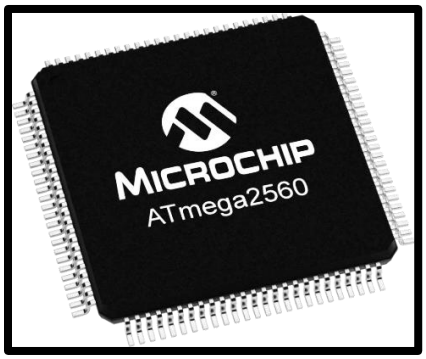
We used Gear motors as actuators whose specs are as follows

| | |
|---|---|
| **Size** | **21 x 14.7 cm approx** |
| **Voltage** | **3-6 V** |
| **Gear Motor Reduction 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/FIRMWARE DESIGN

## 4.1 Controller Selections with features

Atmega2560, commonly found in the Arduino Mega 2560 as its main microcontroller. It's an AVR RISC-based microcontroller that executes powerful instructions in a single clock cycle. This allows it to strike a fine balance between power consumption and processing speed.




Para metrics:

| Name | Value |
|---|---|
| Program Memory Type | Flash |
| Program Memory Size (KB) | 256 |
| CPU Speed (MIPS/DMIPS) | 16 |
| SRAM (KB) | 8,192 |
| Data EEPROM/HEF (bytes) | 4,096 |
| Digital Communication Peripherals | 4-UART, 5-SPI, 1-I2C |
| Capture/Compare/PWM Peripherals | 4 Input Capture, 4 CCP, 16PWM |
| Timers | 2 x 8-bit, 4 x 16-bit |
| Number of Comparators | 1 |
| Temperature Range (°C) | -40 to 85 |
| Operating Voltage Range (V) | 1.8 to 5.5 |
| Pin Count | 100 |

**Advantages:**

- Low power consumption with fast start-up
- Easier to use, with 8-bit microcontroller being less complex than 32/64 bit versions
- QTouch Suite allows for ease of exploring, developing and debugging own touch applications
- Patented Adjacent Key Suppression technology allows for unambiguous detection of key events

**Disadvantages:**

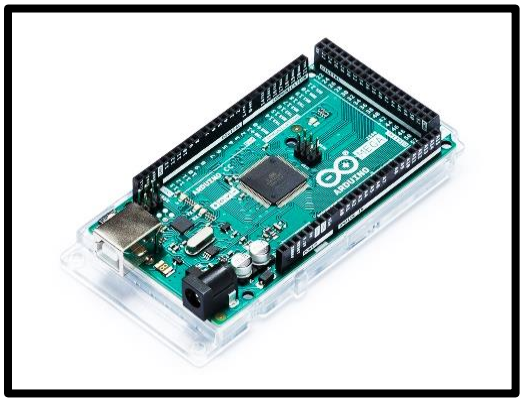
- Limited amount of flash memory write cycles restricts the number of times images can be flashed when programmed to pc
- Naturally lacks incremental performance compared to higher bit microcontrollers

ATmega2560 belongs in an umbrella of microcontrollers; ATmega640/1280/1281/2560/2561. It does share common configurations such as the EEPROM and RAM but still consists of differences as shown below:

*Table 7*

| Device | Flash | General Purpose I/O pins | 16-bit resolution PWM channels | Serial USARTs | ADC Channels |
|---|---|---|---|---|---|
| ATmega640 | 64KB | 86 | 12 | 4 | 16 |
| ATmega1280 | 128KB | 86 | 12 | 4 | 16 |
| ATmega1281 | 128KB | 54 | 6 | 2 | 8 |
| ATmega2560 | 256KB | 86 | 12 | 4 | 16 |
| ATmega2561 | 256KB | 54 | 6 | 2 | 8 |

**Arduino Mega 2560**



Known for its capabilities in handling more complex projects, the Arduino Mega 2560 gives your projects plenty of room and opportunities. It's recommended for 3D printers and robotics projects with its 54 digital I/O pins, 16 analog inputs, and a large space.

Features/Specifications

- Operating voltage: 5V
- Input voltage (recommended): 7-12V
- Input voltage (limits): 6-20V
- Digital I/O pins: 54 (of which 14 provide PWM output)
- Analog input pins: 16
- DC current per I/O pin: 40mA
- DC current for 3.3V pin: 50mA
- Flash Memory: 256 KB, 8KB used by bootloader
- SRAM: 8 KB
- EEPROM: 4 KB
- Clock Speed: 16 MHz

Arduino Mega 2560 is an all-around good option. There's no denying that this board brings performance with the running of an ATmega2560, but it provides a substantial number of I/O pins and program space as well.

If you're currently an Arduino Uno user, it's perhaps time to step up and take a look at the Arduino Mega 2560 for that extra "juice" of performance and I/O pins.

## 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

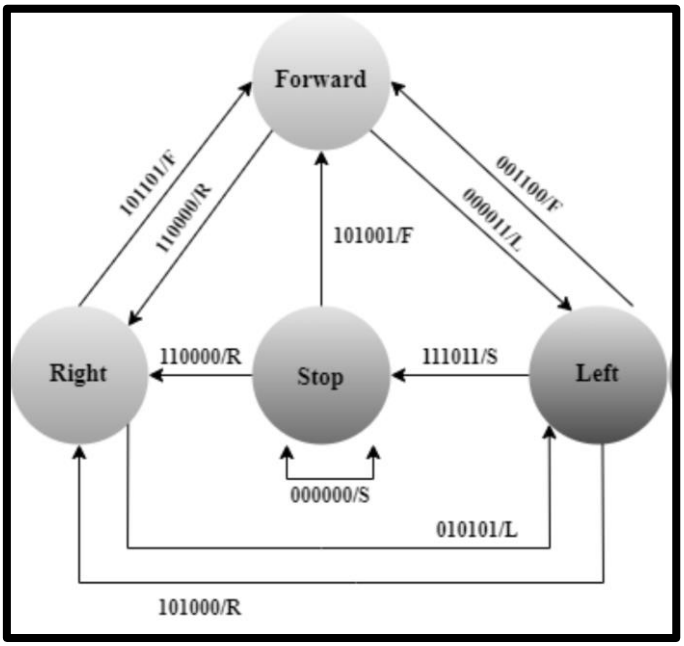
**Inputs**

- 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**

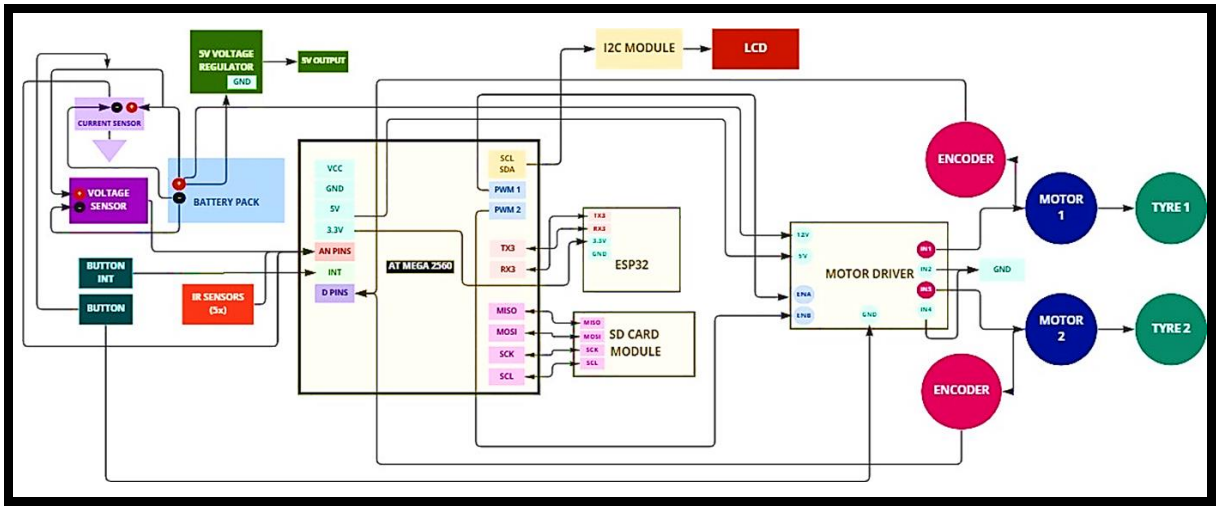
- Left Motor
- Right Motor
- SD card
- 16 X 4 LCD

## 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



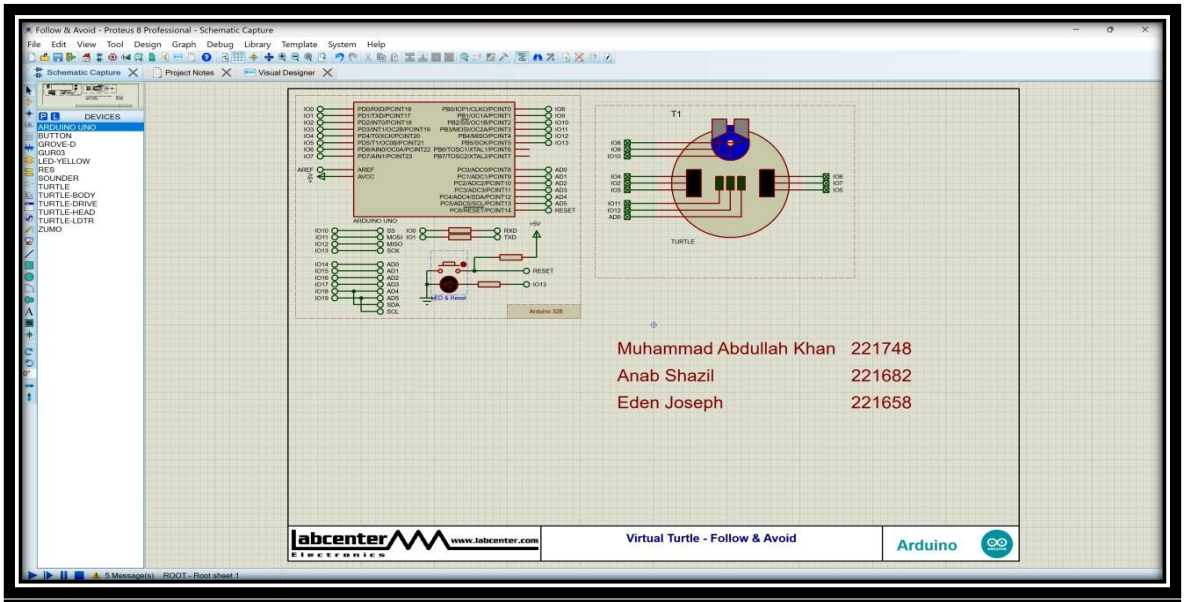

# CHAPTER 5

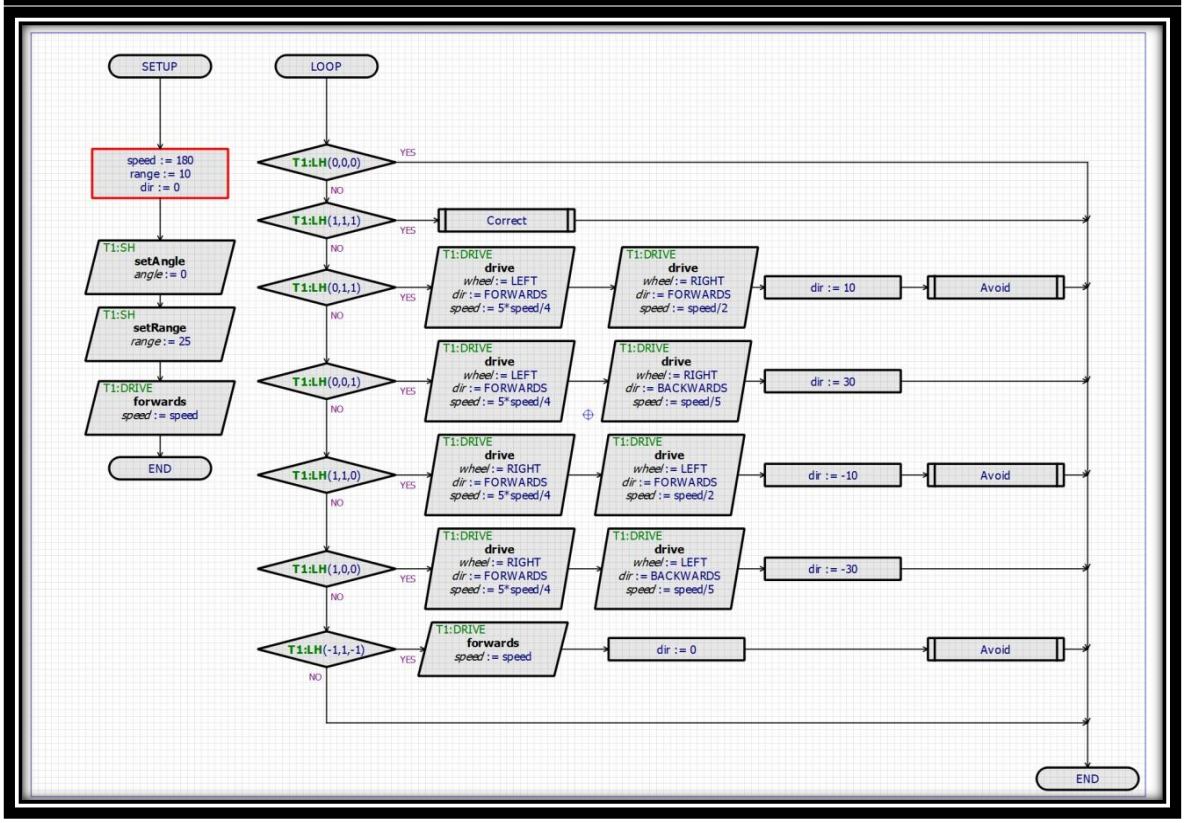
# SIMULATIONS AND FINAL INTEGRATIONS

## 5.1 Integrations and testing all hardware and software component separately

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

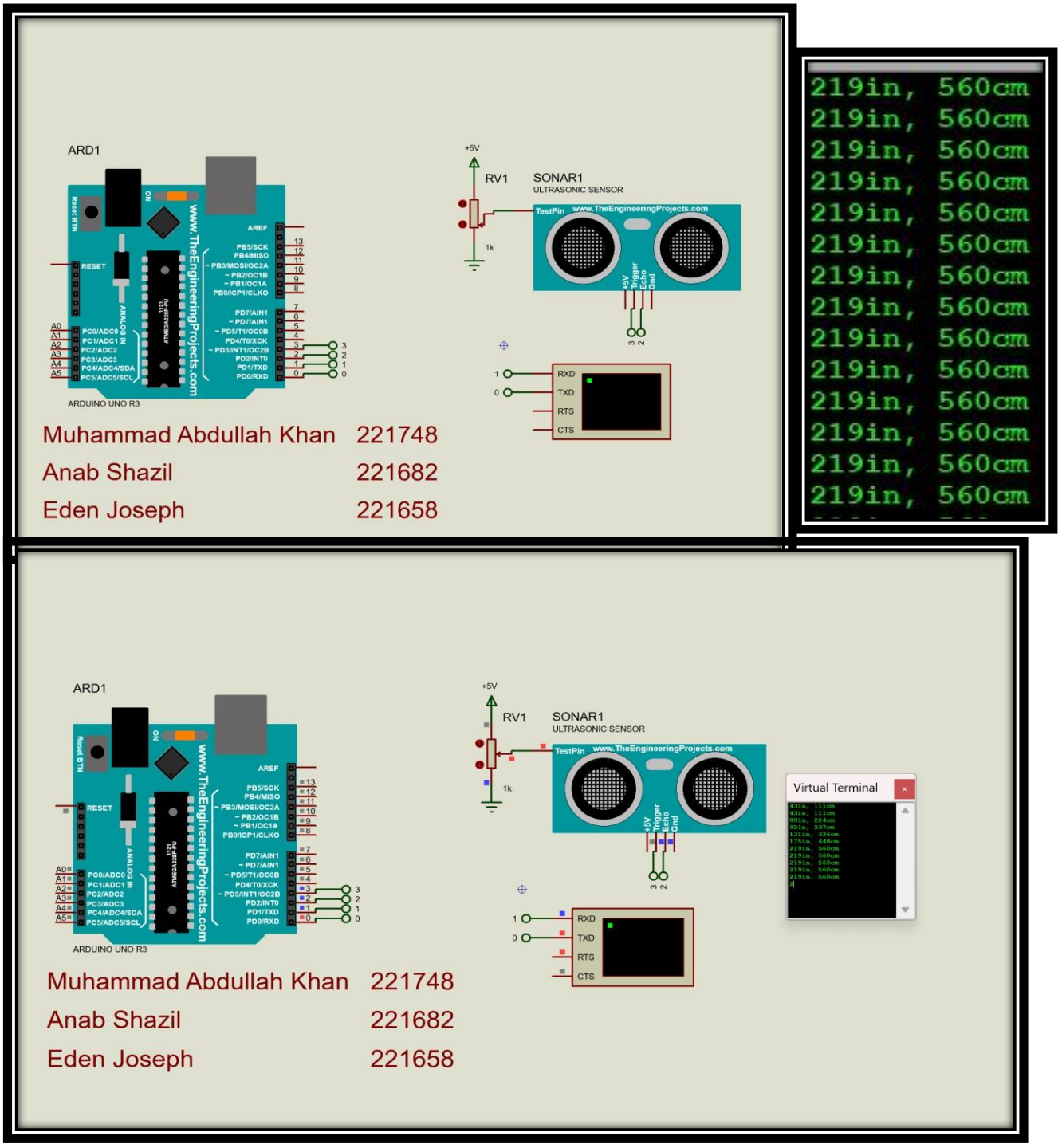
# Simulation


Muhammad Abdullah Khan 221748
Anab Shazil 221682
Eden Joseph 221658
Virtual Turtle - Follow & Avoid


SETUP
LOOP
speed := 180
range := 10
dir := 0
T1:SH setAngle angle := 0
T1:SH setRange range := 25
T1:DRIVE forwards speed := speed
END
T1:LH(0,0,0)
T1:LH(1,1,1)
Correct
T1:LH(0,1,1)
T1:LH(0,0,1)
T1:LH(1,1,0)
T1:LH(1,0,0)
T1:LH(-1,1,-1)
T1:DRIVE drive wheel := LEFT dir := FORWARDS speed := 5*speed/4
T1:DRIVE drive wheel := RIGHT dir := FORWARDS speed := speed/2
T1:DRIVE drive wheel := RIGHT dir := BACKWARDS speed := speed/5
T1:DRIVE drive wheel := LEFT dir := BACKWARDS speed := speed/5
T1:DRIVE forwards speed := speed
dir := 10
dir := 30
dir := -10
dir := -30
dir := 0
Avoid
YES
NO
END

After this, our task was to test each component individually to get an idea on how each component works

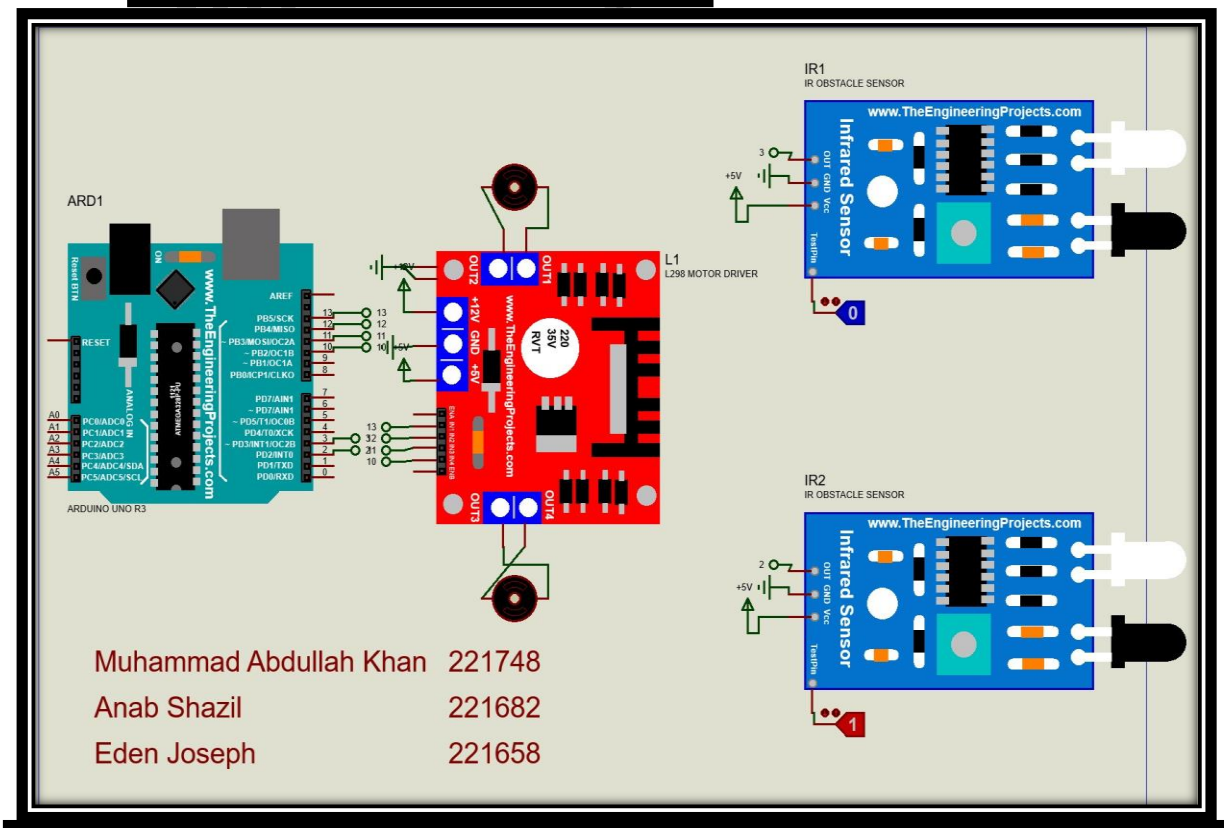
## Ultrasonic Sensor:

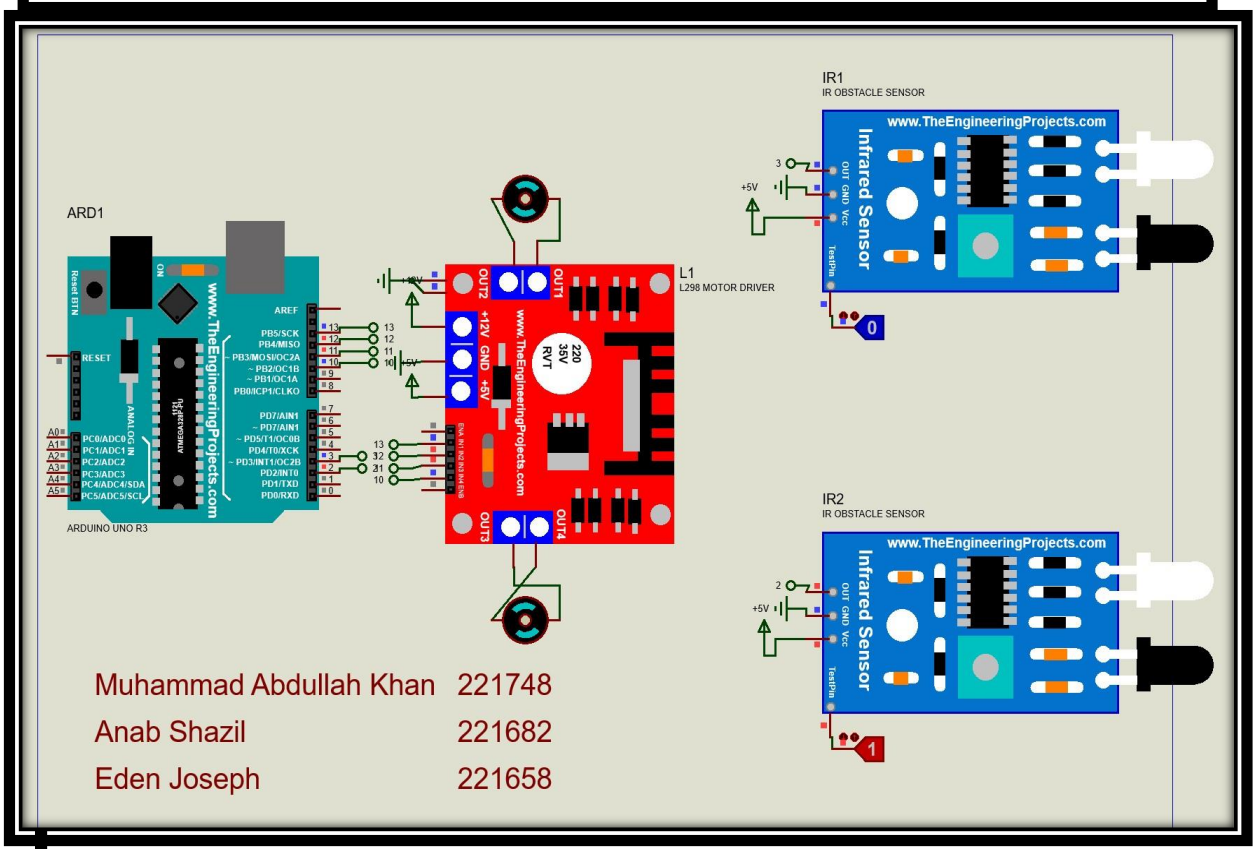


## Arduino IDE Code To Test Ultrasonic Sensor:

```
const int echoPin = 2; // Echo Pin of Ultrasonic Sensor
const int pingPin = 3; // Trigger Pin of Ultrasonic Sensor

void setup()
{
  Serial.begin(9600); // Starting Serial Communication
pinMode(pingPin, OUTPUT); // initialising pin 3 as output
pinMode(echoPin, INPUT); // initialising pin 2 as input
} void
loop()
{   long duration,
inches, cm;
     digitalWrite(pingPin,
LOW);
delayMicroseconds(2);
     digitalWrite(pingPin,
HIGH);
delayMicroseconds(10);
     digitalWrite(pingPin,
LOW);
     duration = pulseIn(echoPin, HIGH); // using pulsin function to determine
total time   inches = microsecondsToInches(duration); // calling method   cm =
microsecondsToCentimeters(duration); // calling method
  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();

delay(100);
} long microsecondsToInches(long microseconds) // method to covert microsec to
inches
{  return microseconds / 74
/ 2;
} long microsecondsToCentimeters(long microseconds) // method to covert
microsec to centimeters
{    return microseconds / 29
/ 2;
}
```

[redacted]



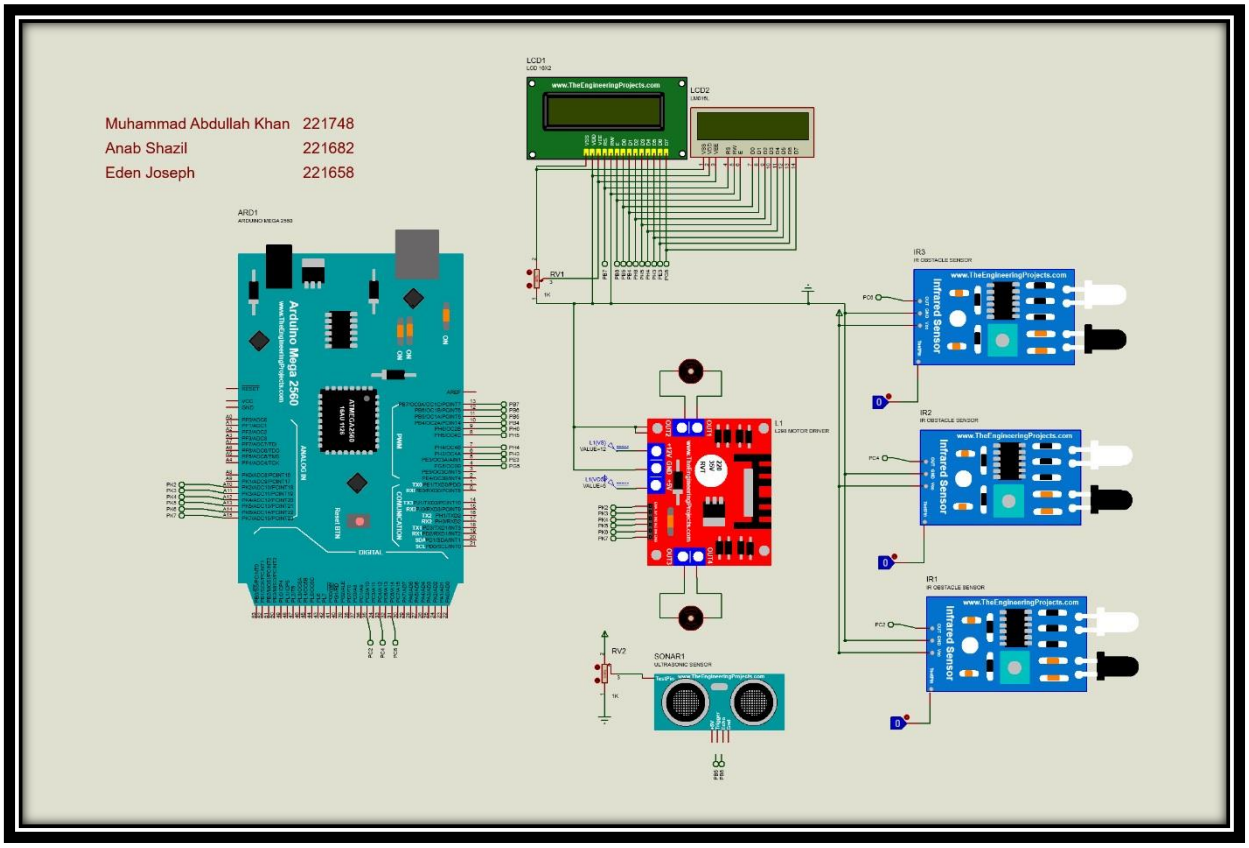



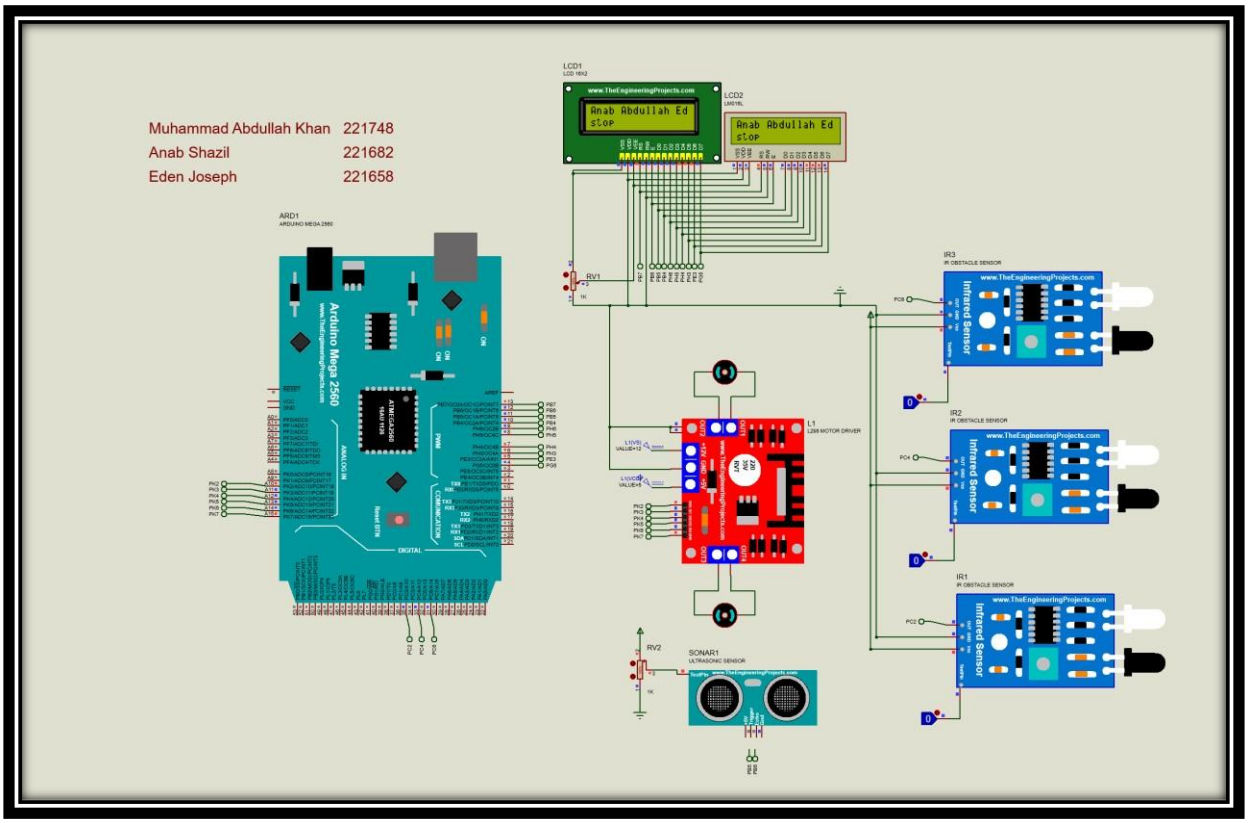

[redacted]

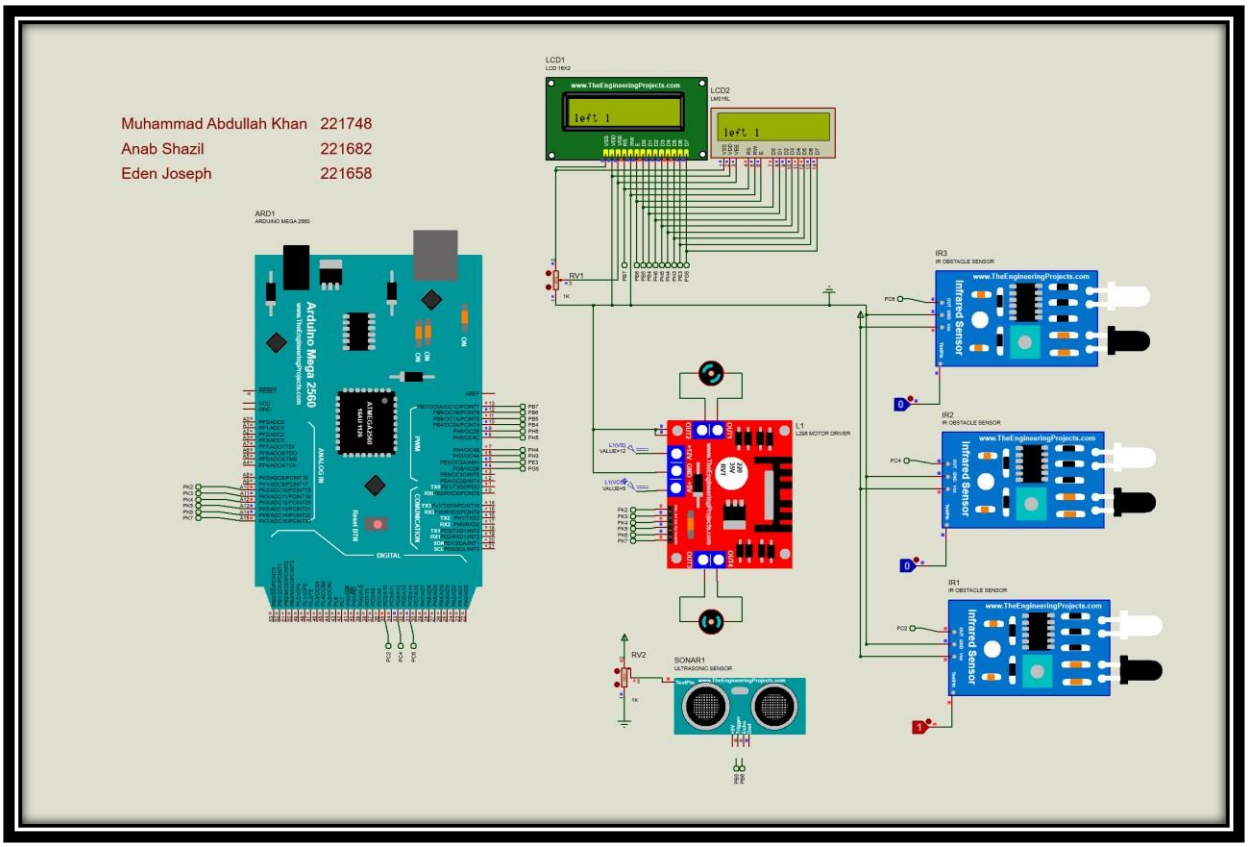
**Arduino IDE Code To Test IR Sensors And Motor Drivers:**

```
void setup() {
pinMode(2, INPUT);
pinMode(3, INPUT);
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);
pinMode(12, OUTPUT);
pinMode(13, OUTPUT);
}

void loop() {
  int v = digitalRead(2);
int s = digitalRead(3);
if(v == 1 and s == 1){
digitalWrite(13, 1);
digitalWrite(12, 0);
digitalWrite(11, 1);
digitalWrite(10, 0);
  }
  if(v == 1 and s == 0){
digitalWrite(13, 0);     digitalWrite(12,
1);     digitalWrite(11, 1);
digitalWrite(10, 0);
  }
  if(v == 0 and s == 1){
digitalWrite(13, 1);     digitalWrite(12,
0);     digitalWrite(11, 0);
digitalWrite(10, 1);
  }
  if(v == 0 and s == 0){
digitalWrite(13, 0);     digitalWrite(12,
1);     digitalWrite(11, 0);
digitalWrite(10, 1);
  }
}
```

## Complete LFR Simulation:






## Arduino IDE Code To Test LFR Simulation:

```
#include <LiquidCrystal.h>

const int rs = 13, en = 12, d0 = 11, d1 = 10, d2 = 9, d3 = 8, d4 = 7, d5 = 6, d6 = 5, d7 = 4;

LiquidCrystal lcd(rs, en, d0, d1, d2, d3, d4, d5, d6, d7);

int S_A = A10;  // speed motor a

int M_A1 = A11; // motor a = +

int M_A2 = A12; // motor a = -

int M_B1 = A13; // motor b = -

int M_B2 = A14; // motor b = +

int S_B = A15;  // speed motor b


int R_S = 31; // sensor R
int S_S = 33; // sensor S int
L_S = 35; // sensor L
```

```
  if ((digitalRead(L_S) == 1) &&
(digitalRead(S_S) == 1)

&& (digitalRead(R_S) ==

0)) {     turnLeft();

lcd.setCursor(0, 1);

lcd.print("left 1");

delay(1000);     lcd.clear();

  }

  if ((digitalRead(L_S) == 1) &&
(digitalRead(S_S) == 0)
&& (digitalRead(R_S) ==
0)) {     turnLeft();
lcd.setCursor(0, 1);
lcd.print("left 2");
delay(1000);     lcd.clear();
```

```
void forword() {
  digitalWrite(M_A1, LOW);
digitalWrite(M_A2, HIGH);
digitalWrite(M_B1, HIGH);
digitalWrite(M_B2, LOW);
}
void turnRight() {
digitalWrite(M_A1, LOW);
digitalWrite(M_A2, LOW);
digitalWrite(M_B1, HIGH);
digitalWrite(M_B2, LOW);
}
void turnLeft() {
  digitalWrite(M_A1, LOW);
digitalWrite(M_A2, HIGH);
digitalWrite(M_B1, LOW);
digitalWrite(M_B2, LOW);
}
void Stopit() {
  digitalWrite(M_A1, LOW);
digitalWrite(M_A2, LOW);
digitalWrite(M_B1, LOW);
digitalWrite(M_B2, LOW);
}
void setup() {
Serial.begin(9600);
lcd.begin(16, 2);
  lcd.print("Anab Abdullah Eden");


  pinMode(M_B1, OUTPUT);
```

```
  }
  if ((digitalRead(L_S) == 0) &&
(digitalRead(S_S) == 1)
&& (digitalRead(R_S) ==
1)) {     turnRight();
lcd.setCursor(0, 1);
lcd.print("right 1");
delay(1000);     lcd.clear();
  }
  if ((digitalRead(L_S) == 0) &&
(digitalRead(S_S) == 0)
&& (digitalRead(R_S) ==
1)) {     turnRight();
lcd.setCursor(0, 1);
lcd.print("right 2");
delay(1000);     lcd.clear();
  }
  if ((digitalRead(L_S) == 1) &&
(digitalRead(S_S) == 1) &&
(digitalRead(R_S) == 1)) {
    Stopit();
    lcd.setCursor(0, 1);
lcd.print("stop");
delay(1000);
lcd.clear();
  }
  if ((digitalRead(L_S) == 0) &&
(digitalRead(S_S) == 0) &&
(digitalRead(R_S) == 0)) {
    Stopit();
    lcd.setCursor(0, 1);
```

```
  pinMode(M_B2, OUTPUT);
pinMode(M_A1, OUTPUT);
pinMode(M_A2, OUTPUT);
pinMode(S_B, OUTPUT);   pinMode(S_A,
OUTPUT);


  pinMode(L_S, INPUT);
pinMode(S_S, INPUT);
pinMode(R_S, INPUT);


  analogWrite(S_A, 1000);
analogWrite(S_B, 1000);
}
void loop() {
  if ((digitalRead(L_S) == 0) &&
(digitalRead(S_S) == 1)
&& (digitalRead(R_S) ==
0)) {     forword();
lcd.setCursor(0, 1);
lcd.print("Forward");
delay(1000);     lcd.clear();
  }
```

```
    lcd.print("stop");
delay(1000);
lcd.clear();
  }
}
```

## 5.2 Compiled Arduino code:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

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


File myFile;     //MAKING OBJECT FOR SD CARD

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

int IR[5]={49,47,45,43,41};   //IR array sensor pins
int sensor[5]={};                     //IR sensor data storing array

const int trigPin = 3;            // For ultrasonic sensor
const int echoPin = 2;
long duration;
int distance;

#define ANALOG_IN_PIN_V A7   //Pin for voltage sensor
#define ANALOG_IN_PIN_C A6   //Pin for current sensor

float adc_voltage = 0.0;   // Floats for ADC voltage & Input voltage
float in_voltage = 0.0;
float R1 = 30000.0;   // Floats for resistor values in divider (in ohms)
float R2 = 7500.0;
float ref_voltage = 5.0;   // Float for Reference Voltage
int adc_value = 0;    // Integer for ADC value

// Define color sensor pins
#define S0 33
#define S1 35
#define S2 37
#define S3 39
#define sensorOut 31

// Define motor pins
int motor1A = 48;
int motor1B = 22;
int motor2A = 46;
int motor2B = 24;
```

```
// Calibration Values
int redMin = 93; // Red minimum value
int redMax = 192; // Red maximum value
int greenMin = 107; // Green minimum value
int greenMax = 255; // Green maximum value
int blueMin = 87; // Blue minimum value
int blueMax = 209; // Blue maximum value

int redPW = 0;       // Variables for Color Pulse Width Measurements
int greenPW = 0;
int bluePW = 0;

int redValue;       // Variables for final Color values
int greenValue;
int blueValue;
//-------------------------------------------------------------------------
-----------------------------------------------------------------


void setup() {
  // put your setup code here, to run once:

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  display.clearDisplay();

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  // Set color sensor pins as outputs
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  // Set motor pins as outputs
  pinMode(motor1A, OUTPUT);
  pinMode(motor1B, OUTPUT);
  pinMode(motor2A, OUTPUT);
  pinMode(motor2B, OUTPUT);


  pinMode(sensorOut, INPUT);   // Set Color Sensor output as input
  pinMode(echoPin, INPUT);    // Sets the echoPin as an Input

  digitalWrite(S0,HIGH);    // Set Frequency scaling to 20% for color sensor
  digitalWrite(S1,LOW);

  // Setup Serial Monitor
```

```
  Serial.begin(9600);

  for(int i=0;i<=4;i++)
  pinMode(IR[i],INPUT);      //decalaring ir sensor pins as inputs

  SD.begin(53);

}
//--------------------------------------------------------------------------
----------------------------------------------------------------------



void loop() {
  // put your main code here, to run repeatedly:
   myFile = SD.open("full.txt", FILE_WRITE);
  // write to it:
  myFile.println("-------------------------------------------------");
  int dis = ULTRA();
  float Amp=Current();
  float bat=Vol();

  Serial.println(Amp);
  Serial.println(bat);
  Serial.println(dis);

   if(dis<=7)
  {
  stop();
  colr();
  }

  for(int i=0;i<=4;i++)
  {
  sensor[i] = digitalRead(IR[i]);  //Storing data in of array
  Serial.print(sensor[i]);
  myFile.print(sensor[i]);

  }
  myFile.println("_-");

if( ( sensor[0]==1 && sensor[1]==1 && sensor[2]==0 && sensor[3]==0 &&
sensor[4]==0 ) || (sensor[0]==1 && sensor[1]==1 && sensor[2]==1 &&
sensor[3]==0 && sensor[4]==0  ) || (sensor[0]==1 && sensor[1]==1 &&
sensor[2]==1 && sensor[3]==1 && sensor[4]==0  )||(sensor[0]==1 &&
sensor[1]==1 && sensor[2]==1 && sensor[3]==0 && sensor[4]==1  ))
{
  digitalWrite(motor1A,HIGH);
  digitalWrite(motor2A,LOW);   //RIGHT
```

```
  digitalWrite(motor1B,LOW);
  digitalWrite(motor2B,LOW);
  myFile.println("Turn Data = Right");
}
if((sensor[0]==0 && sensor[1]==0 && sensor[2]==0 && sensor[3]==1 &&
sensor[4]==1 )||(sensor[0]==0 && sensor[1]==0 && sensor[2]==1 &&
sensor[3]==1 && sensor[4]==1  )||(sensor[0]==0 && sensor[1]==1 &&
sensor[2]==1 && sensor[3]==1 && sensor[4]==1  )||(sensor[0]==1 &&
sensor[1]==0 && sensor[2]==1 && sensor[3]==1 && sensor[4]==1  ))
{
  digitalWrite(motor1A,LOW);
  digitalWrite(motor2A,LOW);   //LEFT
  digitalWrite(motor1B,HIGH);
  digitalWrite(motor2B,LOW);
  myFile.println("Turn Data = Left");
}
if((sensor[0]==1 && sensor[1]==0 && sensor[2]==0 && sensor[3]==0 &&
sensor[4]==1 )||(sensor[0]==1 && sensor[1]==1 && sensor[2]==0 &&
sensor[3]==1 && sensor[4]==1  )||(sensor[0]==1 && sensor[1]==1 &&
sensor[2]==1 && sensor[3]==1 && sensor[4]==1))
{
  digitalWrite(motor1A,HIGH);
  digitalWrite(motor2A,LOW);   //FORWARD
  digitalWrite(motor1B,HIGH);
  digitalWrite(motor2B,LOW);
  myFile.println("Turn Data = Forward");
}
if((sensor[0]==0 && sensor[1]==0 && sensor[2]==0 && sensor[3]==0 &&
sensor[4]==0))
{
  digitalWrite(motor1A,LOW);
  digitalWrite(motor2A,LOW);   //STOP
  digitalWrite(motor1B,LOW);
  digitalWrite(motor2B,LOW);
  myFile.println("Turn Data = Stop");
}
   myFile.println("Total Current = ");
   myFile.print(Amp);

 myFile.println("Total Voltage = ");
 myFile.print(bat);


  myFile.println("No Obstacle");
  myFile.println("-------------------------------------------");
  myFile.close();

  display.clearDisplay();
```

```
  //Voltage sensor
  display.setCursor(0, 15);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.print("Voltage = ");
  display.print(bat);
  display.print(" V");
  display.display();

  //Current sensor
  display.setCursor(0, 25);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.print("Current = ");
  display.print(Amp);
  display.print(" A");
  display.display();

  //Obstacle
  display.setCursor(0, 35);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.print("No Obstacle !");
  display.display();



  Serial.println("done.");


}
//--------------------------------------------------------------------------
----------------------------------
int ULTRA()
{
 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
  return distance;
}
```

```
//--------------------------------------------------------------------------------
---------------------------------
float Current()
{
  int adc = analogRead(ANALOG_IN_PIN_C);
  float voltage = adc * 5 / 1023.0;
  float current = (voltage - 2.5) / 0.100;
  //if (current < 0.16)
  //current = 0;
  return current;

}
//--------------------------------------------------------------------------------
---------------------------------
float Vol()
{
   // Read the Analog Input
   adc_value = analogRead(ANALOG_IN_PIN_V);

   // Determine voltage at ADC input
   adc_voltage  = (adc_value * ref_voltage) / 1024.0;

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

}


void colr()
{
  for(int i=0;i<=5;i++)

 {
  // Read Red value
   redPW = getRedPW();
  // Map to value from 0-255
  redValue = map(redPW, redMin,redMax,255,0);
  // Delay to stabilize sensor
  delay(200);

  // Read Green value
  greenPW = getGreenPW();
  // Map to value from 0-255
  greenValue = map(greenPW, greenMin,greenMax,255,0);
  // Delay to stabilize sensor
  delay(200);

  // Read Blue value
```

```
  bluePW = getBluePW();
  // Map to value from 0-255
  blueValue = map(bluePW, blueMin,blueMax,255,0);
  // Delay to stabilize sensor
  delay(200);

  // Print output to Serial Monitor
  Serial.print("Red = ");
  Serial.print(redValue);
  Serial.print(" - Green = ");
  Serial.print(greenValue);
  Serial.print(" - Blue = ");
  Serial.println(blueValue);

 if(redValue <= 50 && greenValue <=50 && blueValue <=50 && redValue >= 10 &&
greenValue >=10 && blueValue >=10  )
  {
  myFile.println("Black Obstacle");
  myFile.println("Stopped!");
  myFile.println("-----------------------------------------");
  myFile.close();

  display.clearDisplay();
  //Black obstacle
  display.setCursor(27, 10);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.print("Black Obstacle");
  display.setCursor(40, 25);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("Stopped!");
  display.display();


    while(1)
    { stop(); }
  }

 if(redValue>greenValue && redValue > blueValue && greenValue<=163 &&
greenValue>=76&& blueValue<=155 && blueValue>=59)
  {
  myFile.println("Red Obstacle");
  myFile.println("Returning to start");
  display.clearDisplay();
  //Red obstacle
  display.setCursor(27, 10);
  display.setTextSize(1);
  display.setTextColor(WHITE);
```

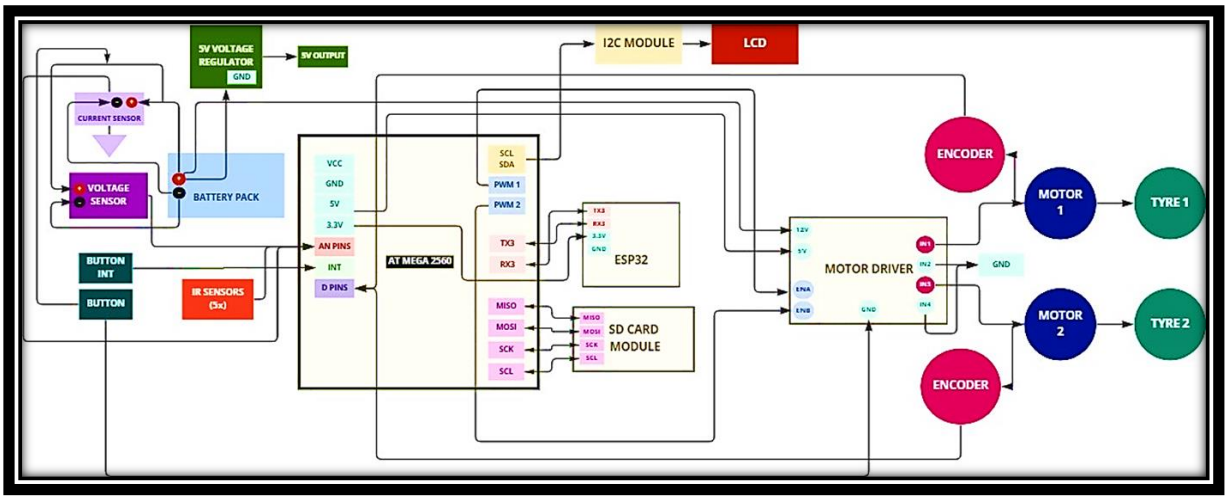
```
  display.print("Red Obstacle");
  display.setCursor(10, 25);
  display.setTextSize(1.5);
  display.setTextColor(WHITE);
  display.println("Returning to Start");
  display.display();
    red();
  }
}
}
//--------------------------------------------------------------------------
----------------------------------
int getRedPW() {                          // Function to read Read Pulse
Widths
  // 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;

}
//--------------------------------------------------------------------------
----------------------------------
int getGreenPW() {                        // Function to read Green Pulse
Widths
  // 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
  return PW;

}
//--------------------------------------------------------------------------
----------------------------------
int getBluePW() {                       // Function to read Blue Pulse Widths
  // 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;

}
//---------------------------------------------------------------------------
-----------------------------------
void stop()
{
  digitalWrite(motor1A,LOW);
  digitalWrite(motor2A,LOW);   //STOP
  digitalWrite(motor1B,LOW);
  digitalWrite(motor2B,LOW);

}
//---------------------------------------------------------------------------
-----------------------------------
void red()
{

  // need work
while(1){
  myFile.println("Reached!");
  myFile.println("-------------------------------------------");
  myFile.close();

  display.clearDisplay();
  //Reached at starting point
  display.setCursor(40, 25);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("Reached!");
  display.display();
}
}
```

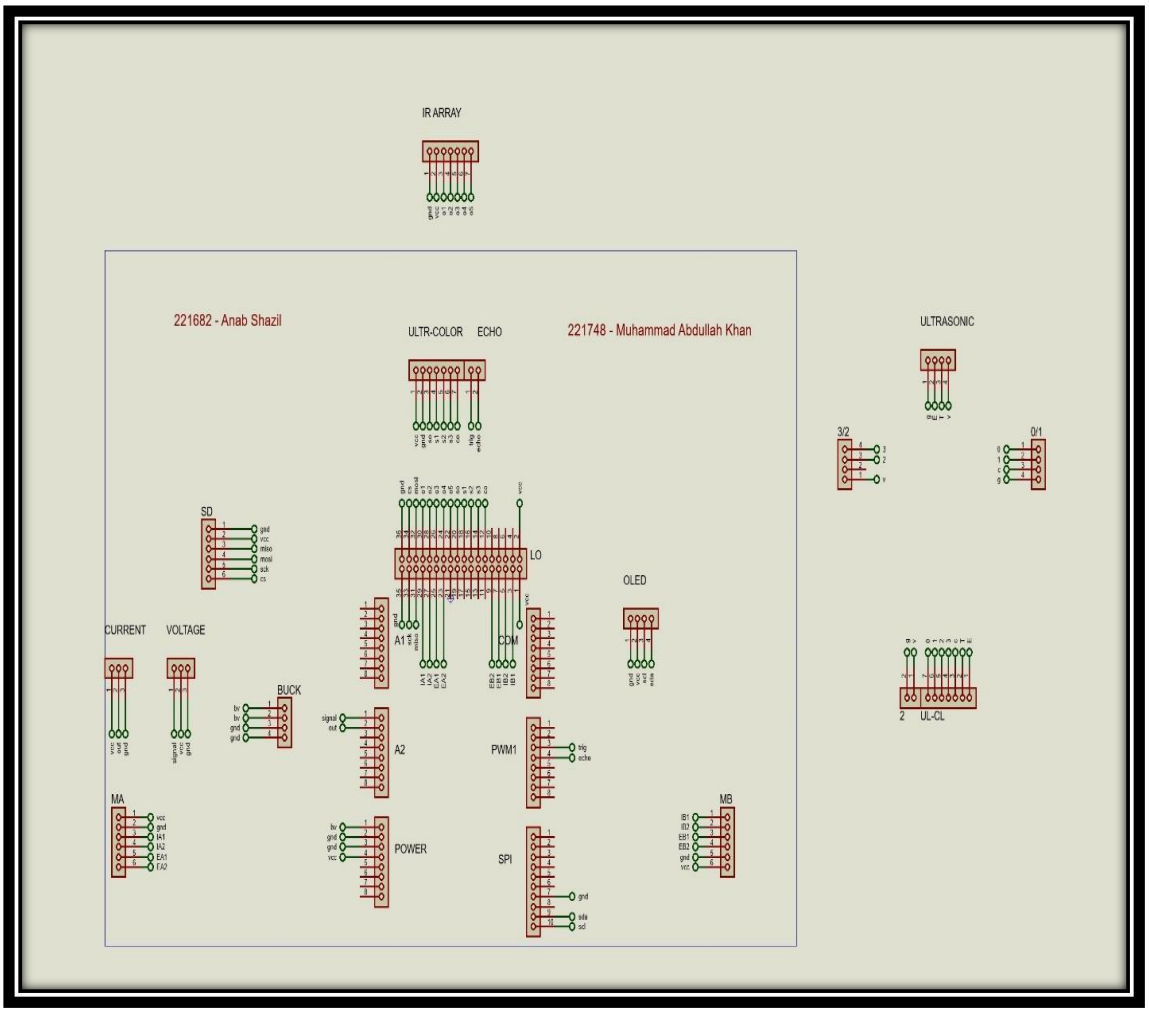
# Chapter 6

# System making and testing phase

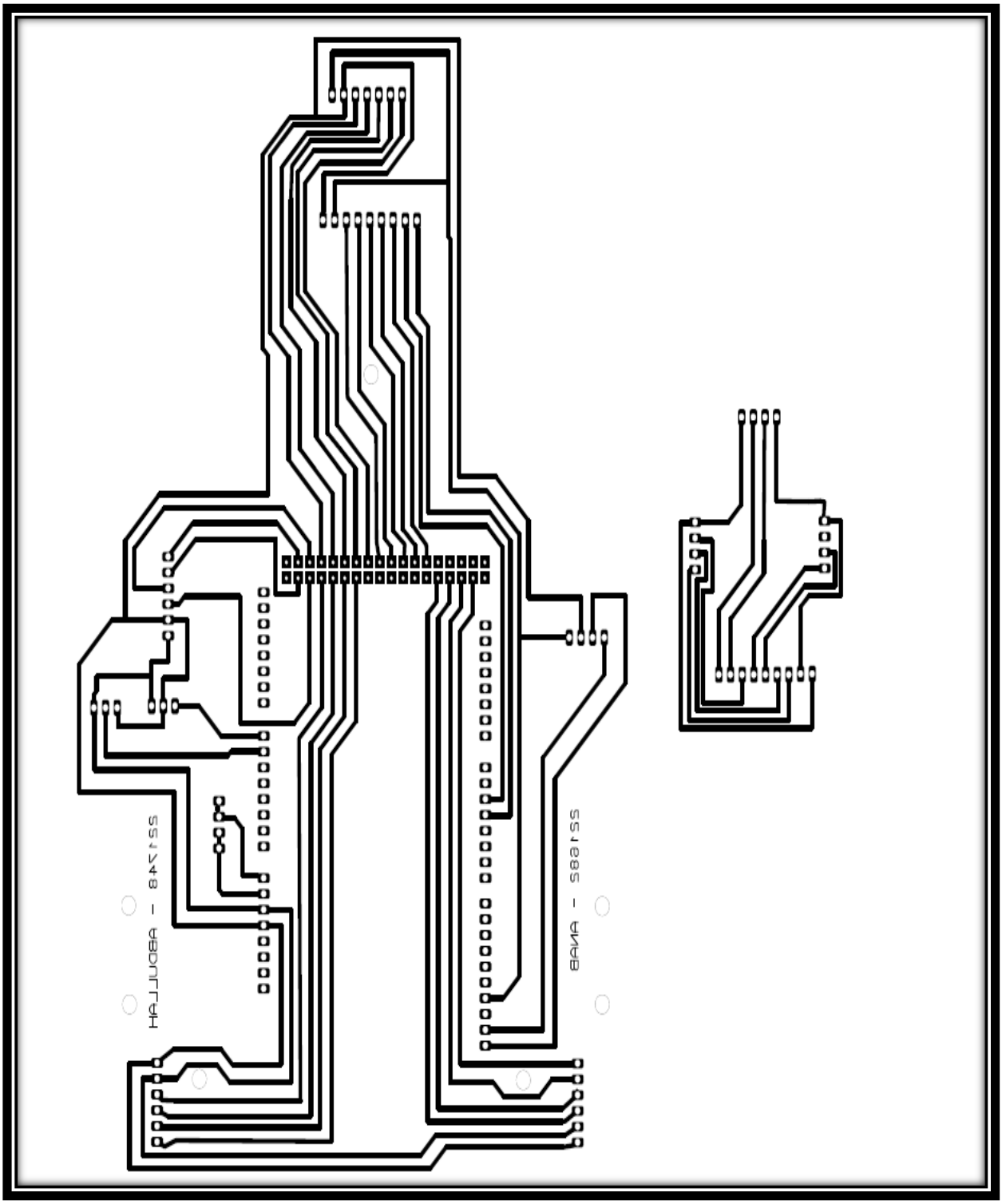
## 6.1 Block diagram:



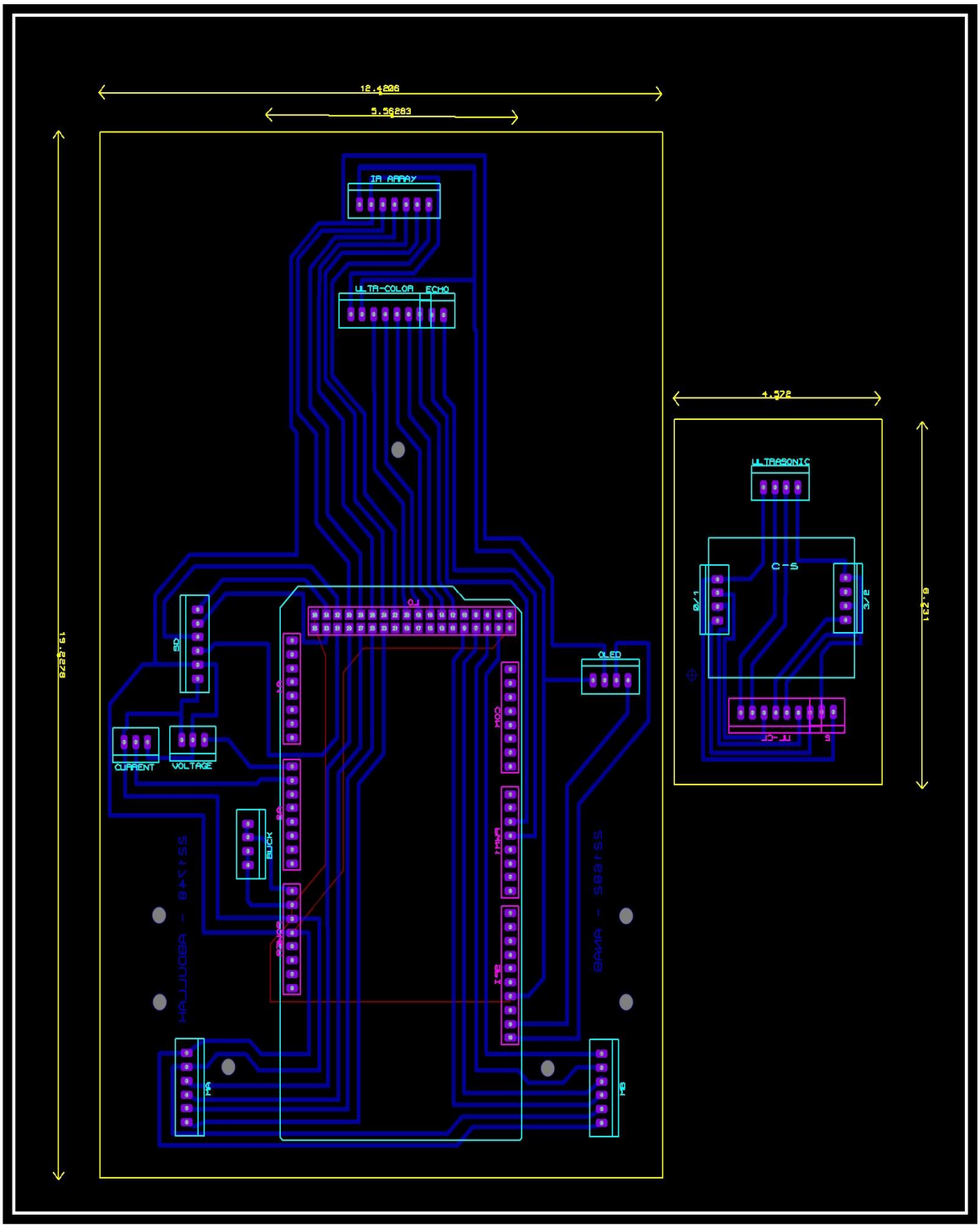

## 6.2 Simulations, print PCB and 3D desighn:



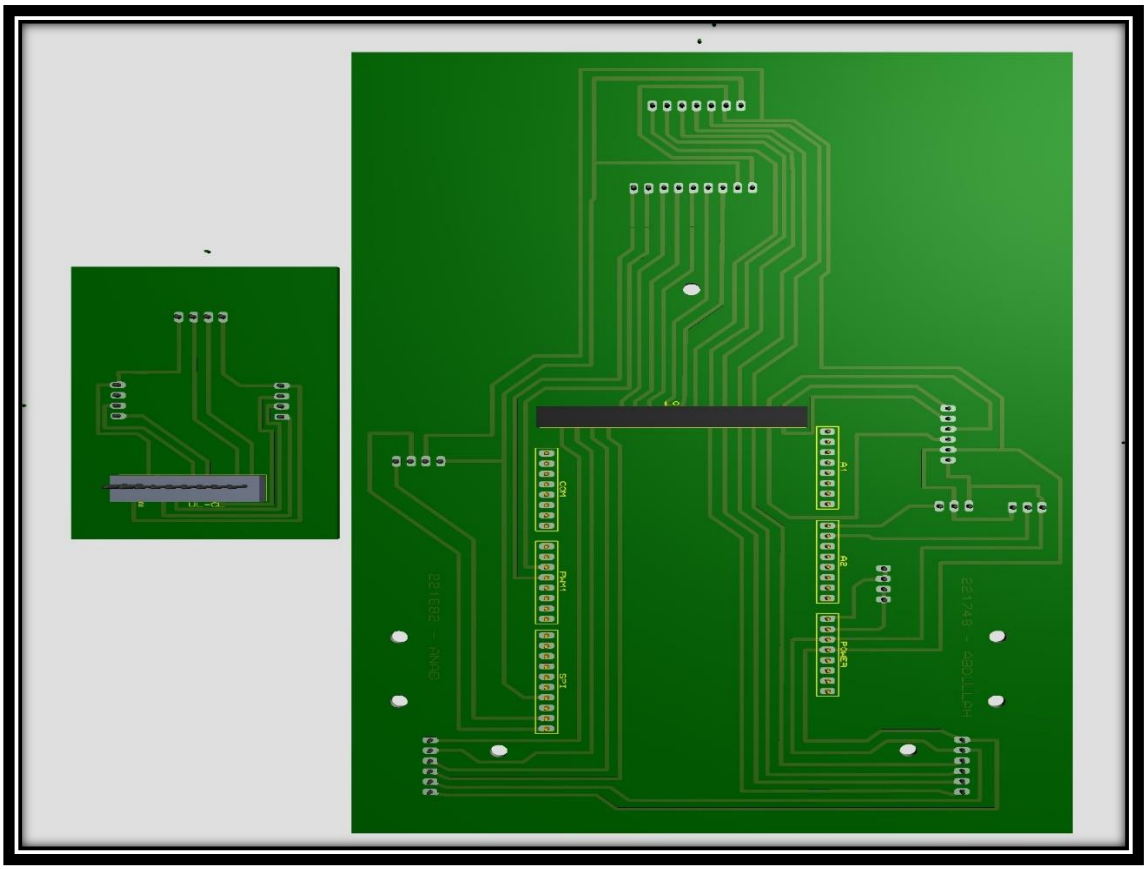
## Simulation print:

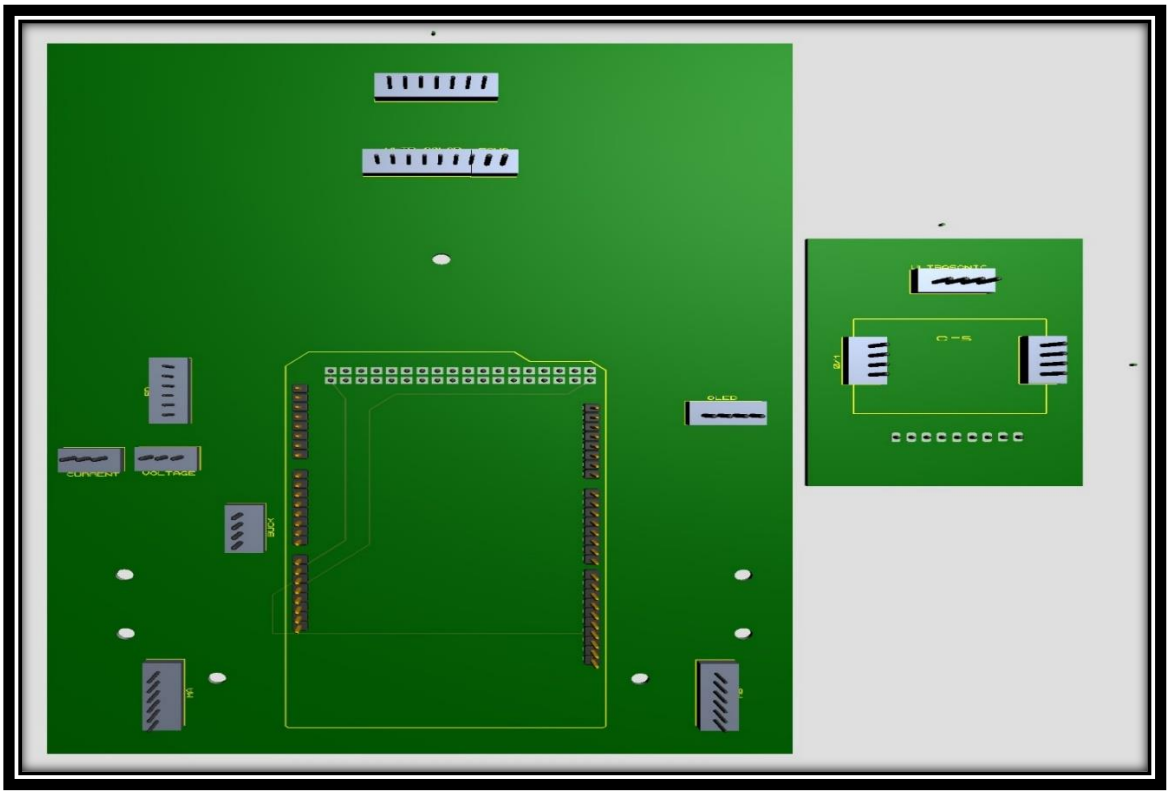


## PCB:

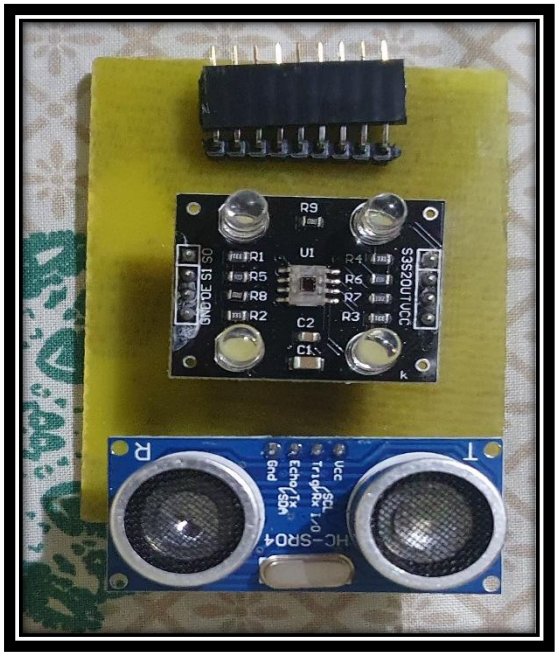

IR ARRAY
ULTRA-COLOR ECHO
ULTRASONIC
OLED
CURRENT
VOLTAGE
BUCK

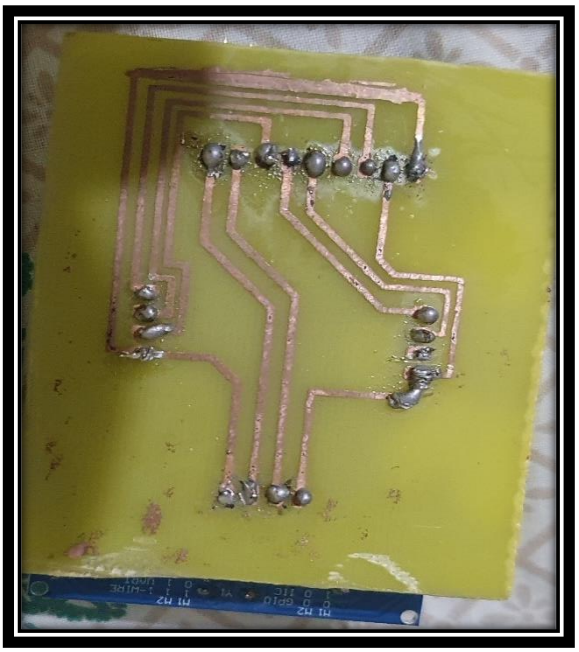
## PCB 3D design:





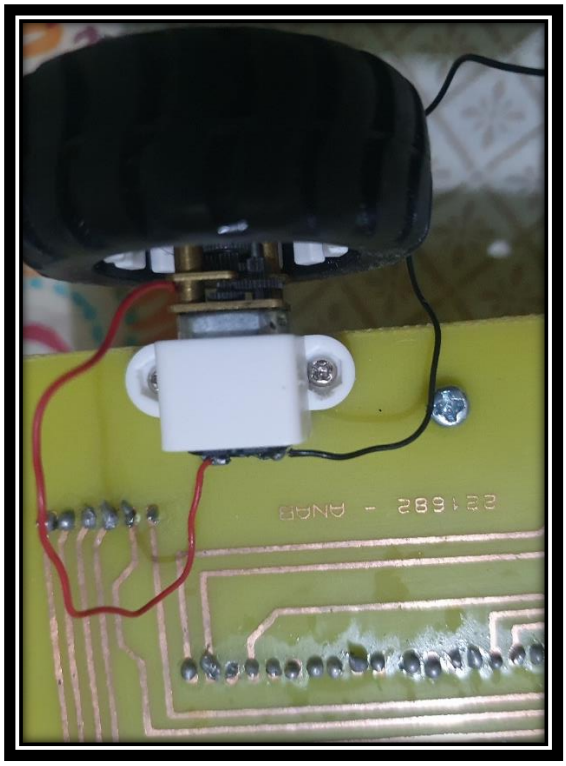
## 6.3 Component functions and real pictures

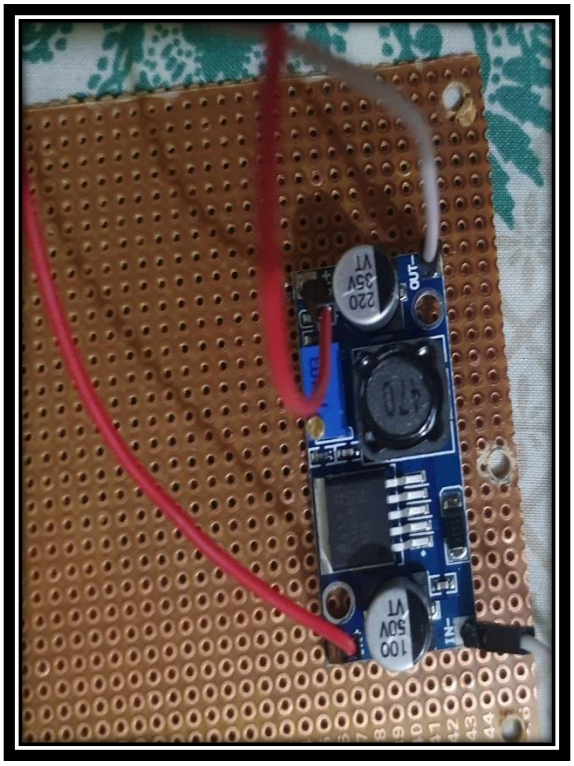


**Ultrasonic sensor:**

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.

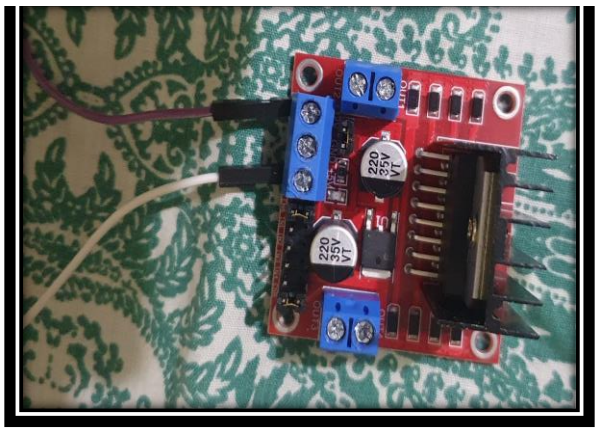




## Motor and buck converter

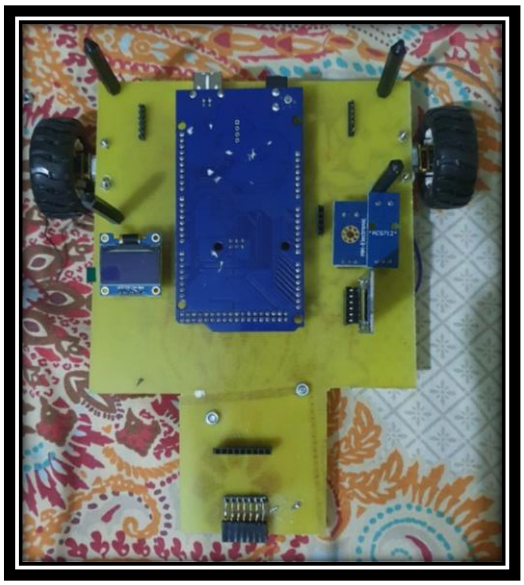

## Battery

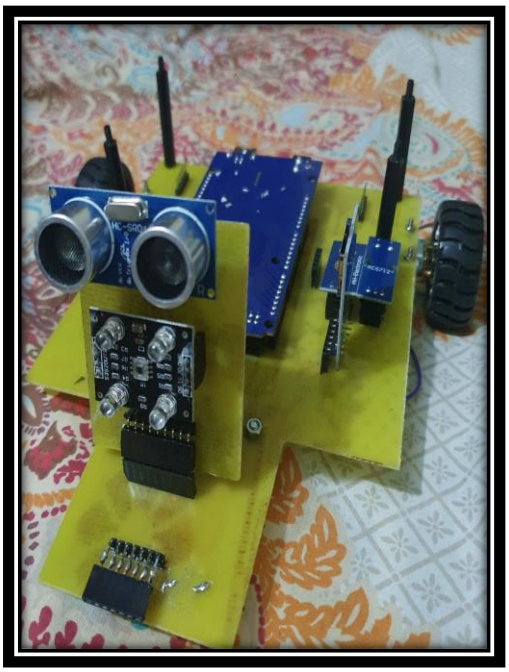
**Lm393 IR sensor:** LM393 Motor Speed Measuring Sensor Module For Arduino, the major goal is to check the rate of an electric motor. The module can be used in association with a microcontroller for motor speed detection, pulse count, position limit, etc. LM393 Motor Speed Measuring Sensor Module widely used in motor speed detection, pulse count, the position limit, etc. The DO output interface can be directly connected to a micro-controller IO port, if there is a block detection sensor, such as the speed of the motor encoder can detect.This is a thorough LM393 Motor Speed Measuring Sensor Module For Arduino.

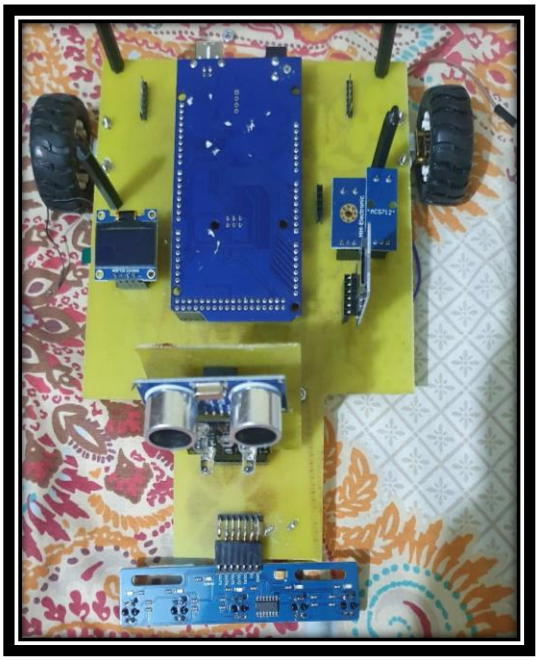


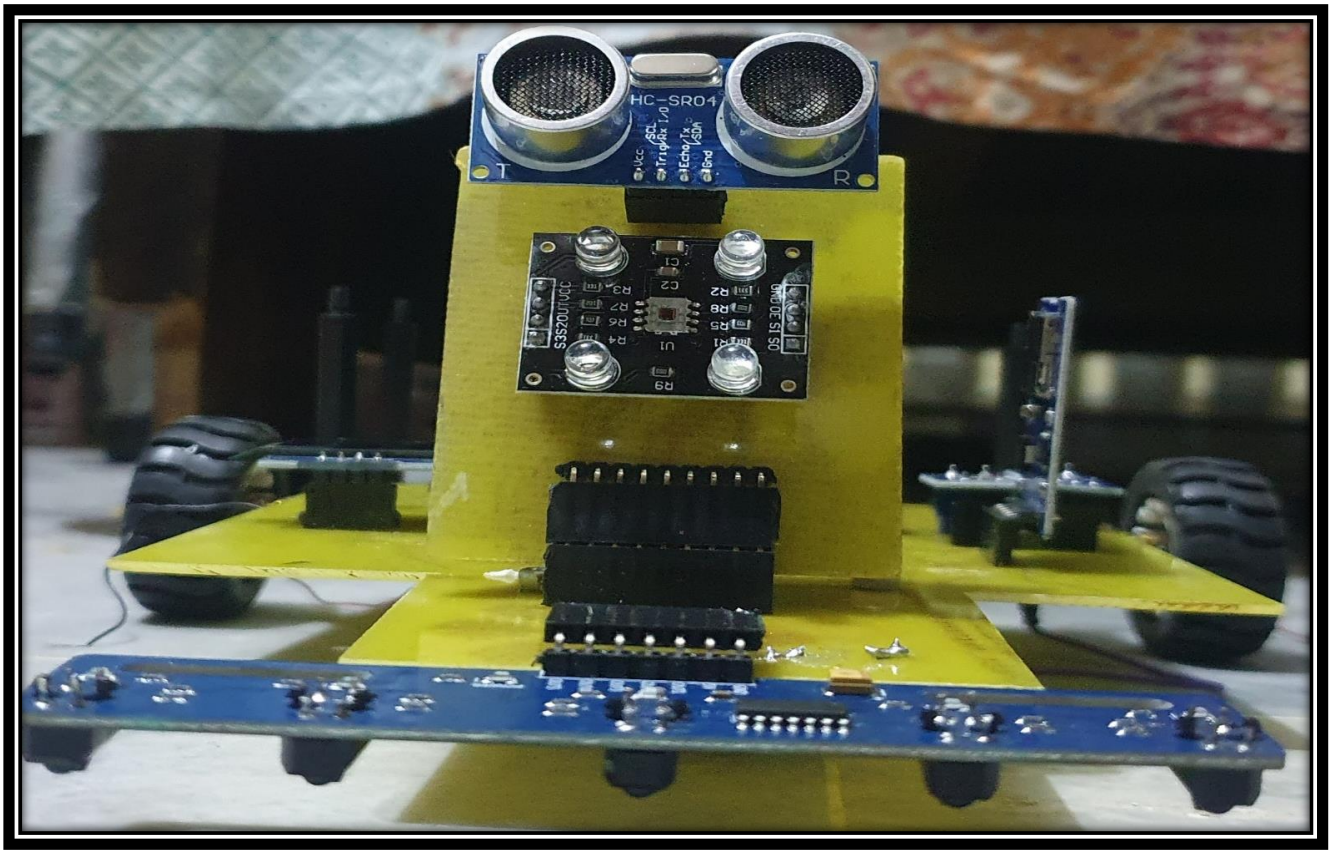

**L298N Motor Driver**

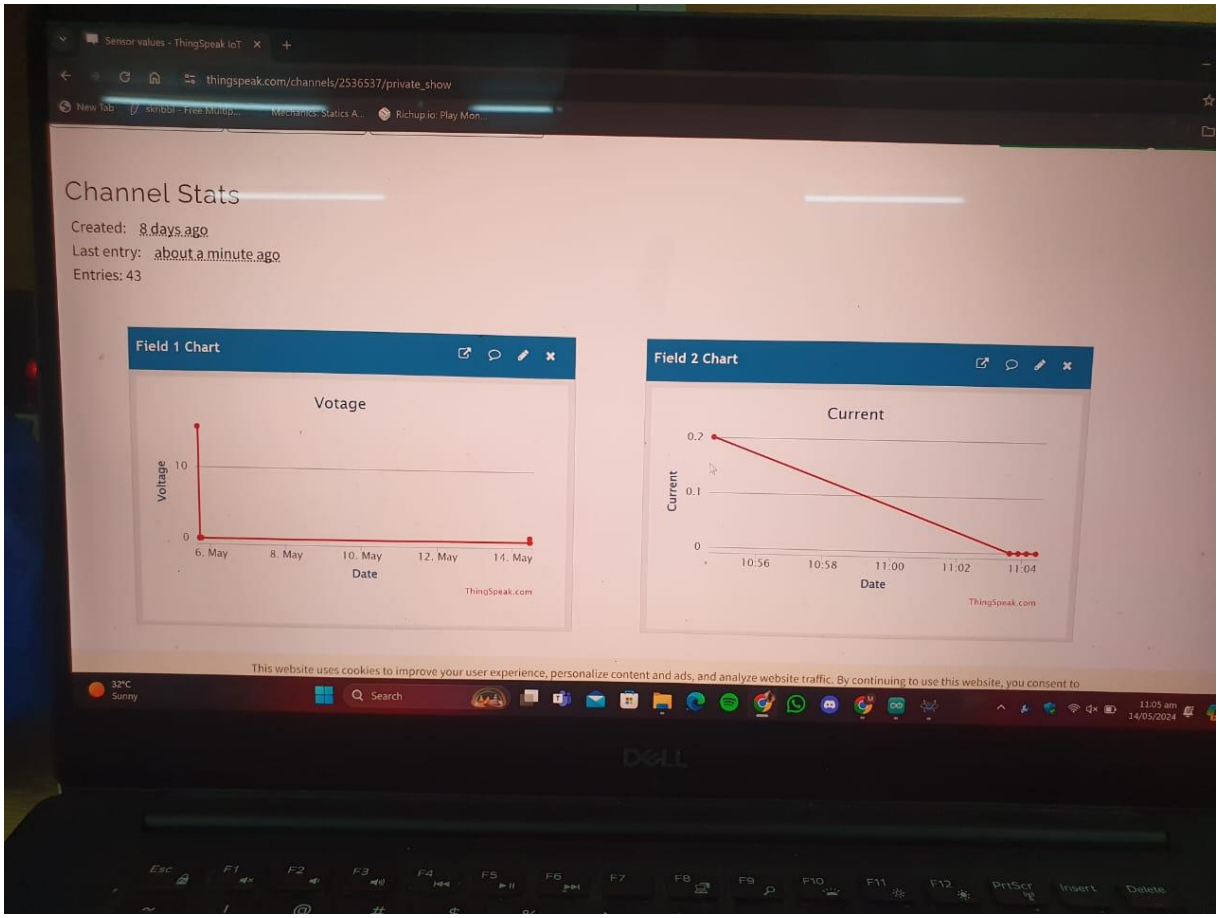
This **L298N Motor Driver Module** is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. **L298N Module** can control up to 4 DC motors, or 2 DC motors with directional and speed control.








HC-SR04
T
R




Voltage and Current Values Data on Webserver using ESP32

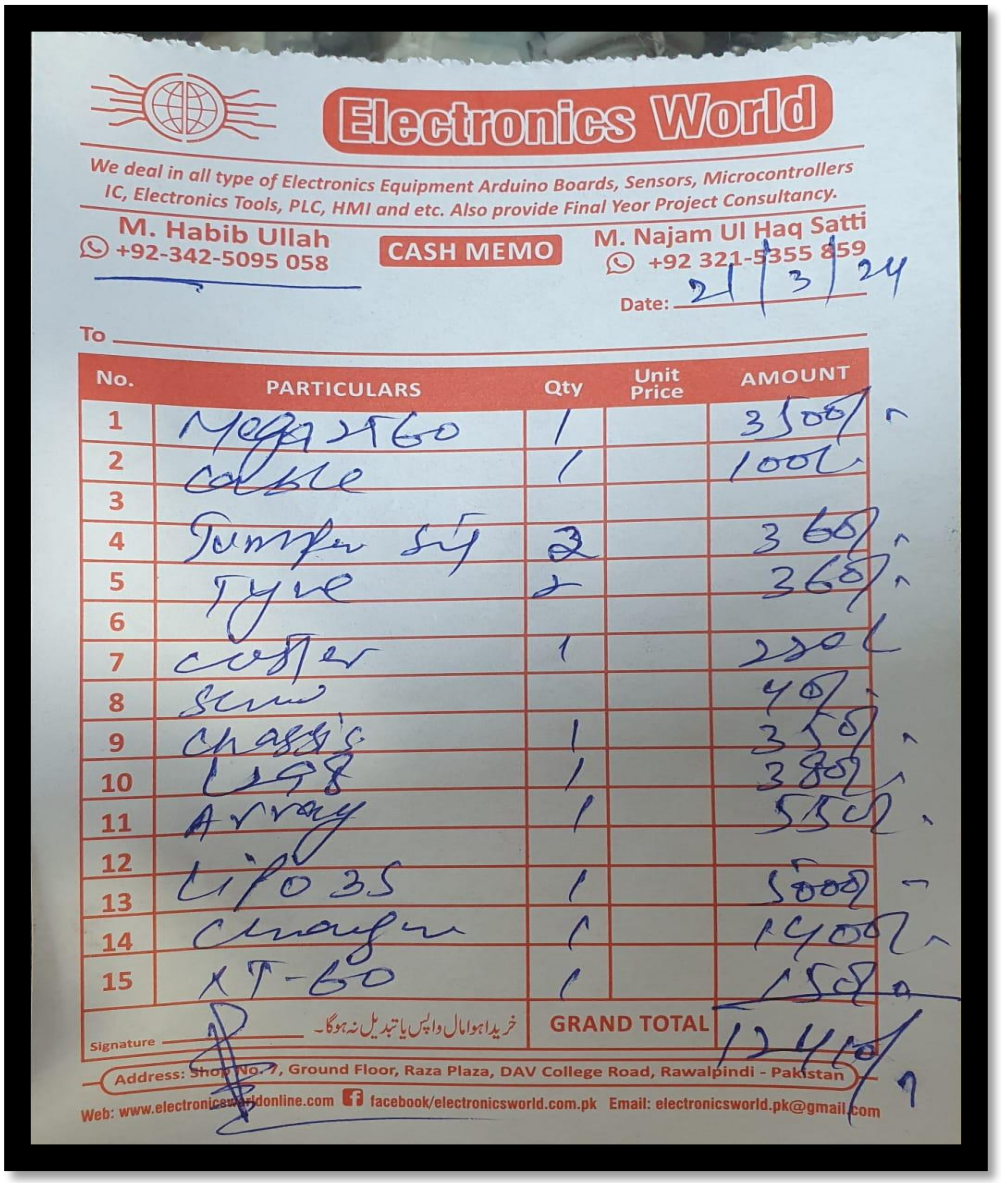
# CHAPTER 7

# PROJECT MANAGEMENT

## 7.1 Comment individually about success failure of your project:

| | |
|---|---|
| **Anab Shazil** | The project was a success |
| **M. Abdullah Khan** | The project was a success |

## 7.2 Bill of material list:



Electronics World

We deal in all type of Electronics Equipment Arduino Boards, Sensors, Microcontrollers IC, Electronics Tools, PLC, HMI and etc. Also provide Final Yeor Project Consultancy.

M. Habib Ullah
+92-342-5095 058

CASH MEMO

M. Najam Ul Haq Satti
+92 321-5355 859

Date: 21/3/24

To

| No. | PARTICULARS | Qty | Unit Price | AMOUNT |
|---|---|---|---|---|
| 1 | Mega 2560 | 1 | | 3500/- |
| 2 | cable | 1 | | 1000/- |
| 3 | | | | |
| 4 | Jumper set | 2 | | 360/- |
| 5 | Tyre | 2 | | 368/- |
| 6 | | | | |
| 7 | caster | 1 | | 250/- |
| 8 | Screw | | | 40/- |
| 9 | Chassis | 1 | | 350/- |
| 10 | L298 | 1 | | 380/- |
| 11 | Array | 1 | | 550/- |
| 12 | | | | |
| 13 | LiPO 35 | 1 | | 5000/- |
| 14 | Charger | 1 | | 1400/- |
| 15 | XT-60 | 1 | | 150/- |
| | | | GRAND TOTAL | 12440/- |

Signature

خریداہوامال واپس یاتبدیل نہ ہوگا۔

Address: Shop No. 7, Ground Floor, Raza Plaza, DAV College Road, Rawalpindi - Pakistan

Web: www.electronicsworldonline.com facebook/electronicsworld.com.pk Email: electronicsworld.pk@gmail.com


**Total Bill = 19595 Rs**

### 7.3 Give a word count how many words each member write in final report in their allocated colour as assigned:

| Name | Roll number | Words count |
|---|---|---|
| Anab Shazil | 221682 | **4000** |
| M. Abdullah Khan | 221748 | **935** |

| | | |
|---|---|---|
| **Anab Shazil** | **221682** | Yellow |
| **Muhammad Abdullah Khan** | **221748** | Red |

### 7.4 Failure and its solution:

1. One of the failures was PCB etching, the copper traces were missing and we had to solder the missing phases.

2. Another issue was Libraries in Arduino IDE. The libraries for ESP32 and OLED even after downloading said that missing libraries.

## CHAPTER 8

The course has three projects of absolute 10 marks which is not worth it as it is much more time consuming and effort consuming. Otherwise the course is great on theoretical perspective and we expect it to run smoothly.