Line Following Robot

Group Members:

MUHAMMAD RYAN SABAJA (211244)

MUHAMMAD AHSAN SAJJAD (211256)

HAIDER SAJJAD (211262)

MUHAMMAD MAHMOOD GHAURI (211265)

BE MECHATRONICS (Session 2021-2025)

Project Supervisor

UMER FAROOQ

Lecturer

DEPARTMENT OF MECHATRONICS ENGINEERING

FACULTY OF ENGINEERING

AIR UNIVERSITY, ISLAMABAD

CHAPTER 1 PRELIMINARIES

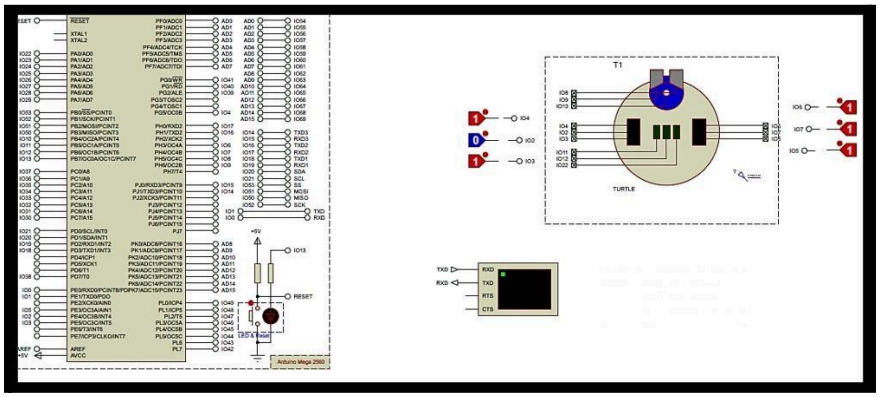
1.1 Proposal

The main objective of this project was to design a line follower according to the instructions provided. Line follower robot basically follows a provided line until that exists or an obstacle is detected in front of the robot . On detection of an obstacle, the robot moves back to the starting point by using the path stored in the SD card. The current and voltage of the battery are displayed on an LCD placed on the robot. The speed of the robot is also displayed on the LCD. Also on detection of an obstacle, the color of the obstacle is displayed on LCD. The values of current, voltage, speed and color of obstacle are wirelessly transmitted to any bluetooth device.

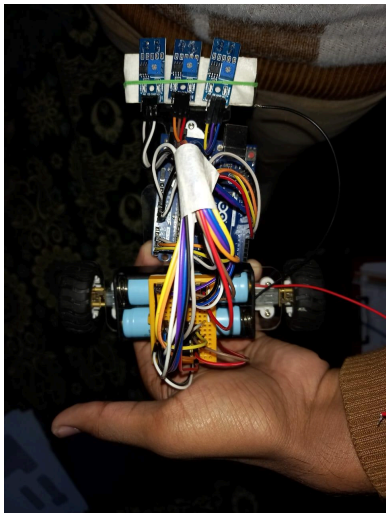
1.2 Initial feasibility

The initial phase was divided into two parts:

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



Secondly , To design a basic line follower using 3 IR sensors, 2 motors, and L293D motor driver. This stimulation was aimed to understand the core working of the line follower.



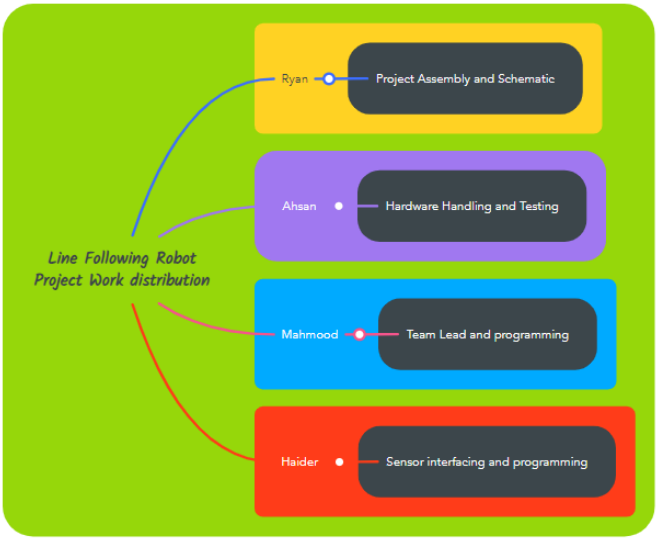
Specifications of deliverables

1. You must design a custom ATMEL controller-based board by yourself for this project.
2. Simulation should be done on Proteus and PCB should be designed in Proteus software.
3. 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).
4. Micro SD card module must be used to store the information for logging.
5. IR, Ultrasonic, Color Sensors can be used to detect red colored Obstacle.
6. On Embedded side we must use I/O, ADC, Timers, Interrupts, and PWM.
7. Robot should send data wirelessly to a remote computer using a wireless module.
8. A report describing your effort Block diagram with pins /Algorithms/State machine, Schematics, component list, Bill of Material, Limitation, future works, references should be submitted.

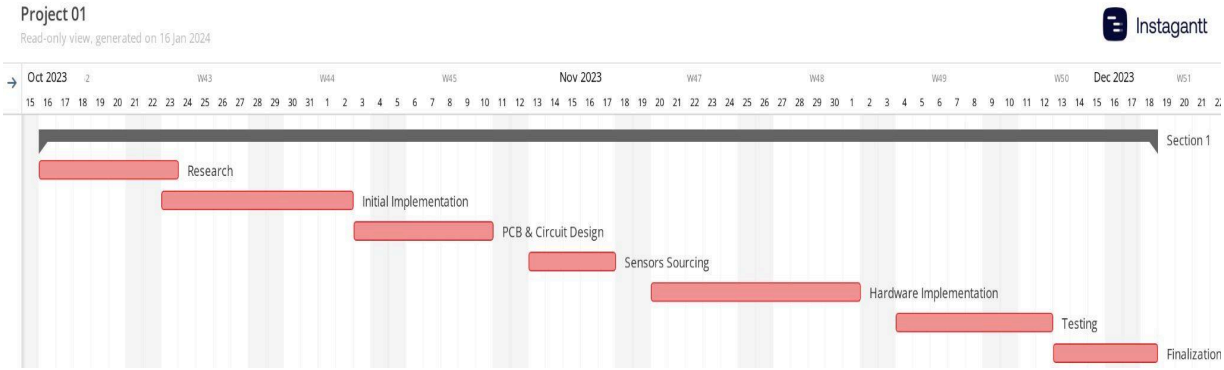
1.3 Team Roles & Details

Team Member Name	Role
MUHAMMAD MAHMOOD GHAURI	Team Lead and programming
MUHAMMAD AHSAN SAJJAD	Hardware Handling and Testing
MUHAMMAD RYAN SABAJA	Project Assembly and Schematic
HAIDER SAJJAD	Sensor interfacing and programming.

1.4 Work Breakdown structure



1.5 Gantt Chart



1.6 Estimated Budget

Initially the estimated cost of this project was Rs.10000.

CHAPTER 2 PROJECT CONCEPTION

2.1 Introduction

Line follower robot basically follows a provided line using 3 IR sensors until that exists or an obstacle is detected in front of the robot . On detection of an obstacle using an Ultrasonic sensor, the robot moves back to the starting point by using the path stored in the SD card. The current and voltage of the battery are displayed on an LCD using an I2C protocol based display module and placed on the robot. The speed of the robot measured using a tachometer is also displayed on the LCD. Also on detection of an obstacle, the color of the obstacle detected using color sensor is displayed on LCD. The values of current, voltage, speed and color of obstacle are wirelessly transmitted to any bluetooth device using the bluetooth module.

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

- 3 IR sensors are used to detect black lines with great precision thus the angles on path such as a 90 turn and 45 turn are also covered.
- The project is also capable of avoiding any obstacles as ultrasonic sensor is used.
- IR sensor is used to implement tachometer determine the speed of motors
- We have used two motors for two wheels mounted
- We have used an SD card to store the path that robot follows .
- Finally, the LCD panel helps to display the input Voltage and current, speed and color of the obstacle.
- Also input voltage and current, speed of motors and color of the obstacle is wirelessly transmitted to any bluetooth device using bluetooth module.

Operation:

The basic operations of line follower are as follows:

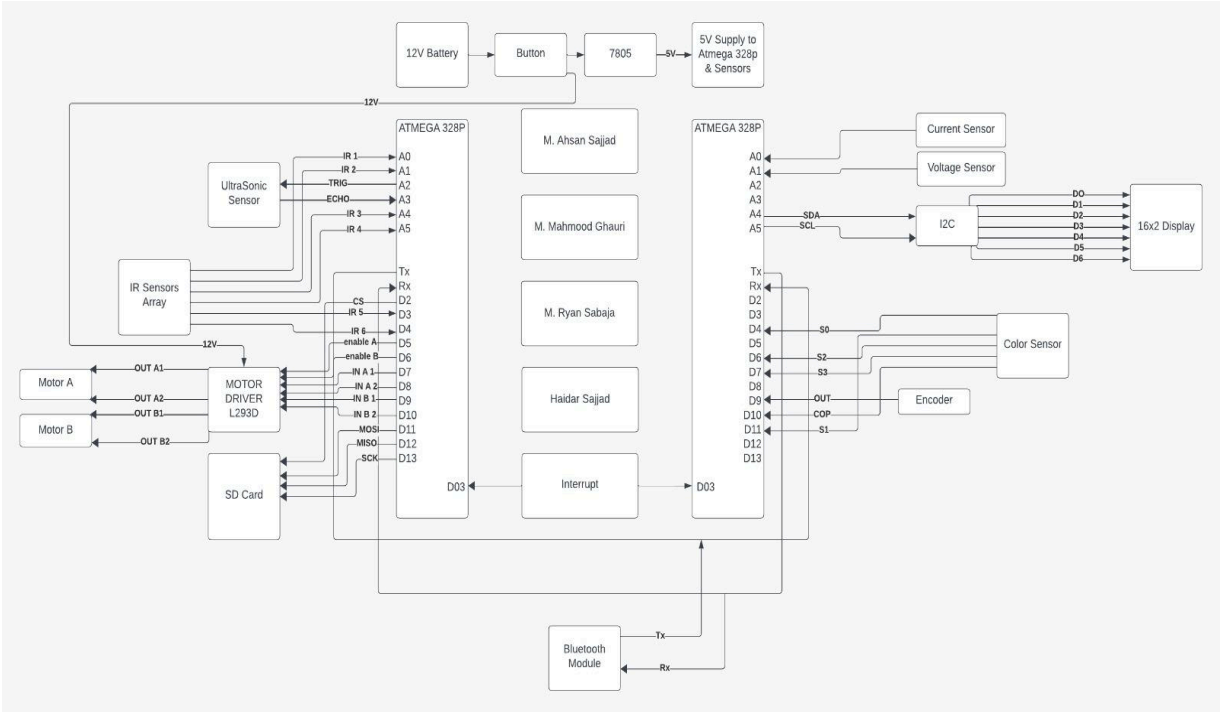
- It follows a black line.
- On detection of obstacle, robot should stop.
- Color of Obstacle should be detected.
- It should stores it path in SD card and on detection of obstacle, by using path stored in SD card robot should move back to its starting position.

2.3 Project Development Process

- After the detailed literature survey through the books, websites and documents provided, the idea of the project is well defined.
- The logic is derived for the intelligence of the robot. It is programmed and burns it to the Atmega 328p by using the software Arduino IDE.
- 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 finishing the programming, electrical and electronics part, the stable, reliable and flexible mechanical design and fabrication is completed.
- Finally, the system is tested and an encountered error is omitted.

2.4 Basic block diagrams of whole system and subcomponents

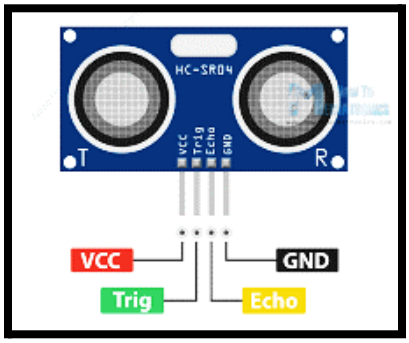
Below is the basic block diagram that shows main components of the circuit design and to give a basic understanding of how the project is planned and designed.



2.5 Complete Speciation of Individual Component

Ultrasonic sensor:

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



Pin Configuration:

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

Atmega 328p:

The ATmega328P is a popular 8-bit microcontroller from the Atmel AVR family. It is widely used in various embedded systems and projects due to its versatility and ease of use. Here's a basic introduction to the ATmega328P:

Memory:

- Flash Memory: 32 KB for storing the program (your code).
- SRAM (Static RAM): 2 KB for storing data during runtime.
- EEPROM (Electrically Erasable Programmable Read-Only Memory): 1 KB for non-volatile data storage.

Clock Speed:

- The ATmega328P typically operates at a clock speed of 16 MHz. However, it can run at lower speeds by configuring the internal clock divider.

Peripherals:

- Timers and Counters: The ATmega328P has several timers/counters that can be used for various purposes, such as generating PWM signals or measuring time intervals.
- Serial Communication: It supports UART (Universal Asynchronous Receiver-Transmitter) for serial communication.
- GPIO (General-Purpose Input/Output): It has multiple digital and analog pins that can be configured as inputs or outputs.

Analog-to-Digital Converter (ADC):

- The ATmega328P has a built-in ADC that allows you to convert analog signals to digital values.

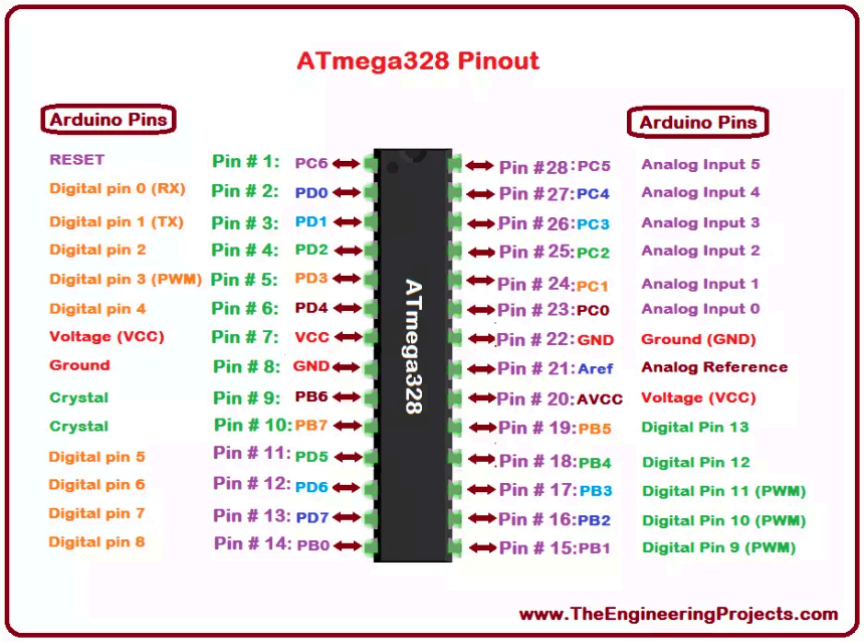
Programming:

- The microcontroller is programmed using a programming language like C or Assembly.
- It is often programmed using the Arduino IDE, which simplifies the development process, making it accessible to a wider audience.

Arduino Compatibility:

- The ATmega328P is the heart of many Arduino boards, such as the Arduino Uno. The Arduino platform provides a user-friendly environment for programming and interfacing with the microcontroller.

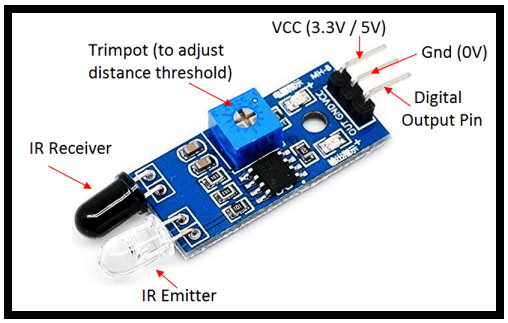
Pin Configuration:



IR sensor:

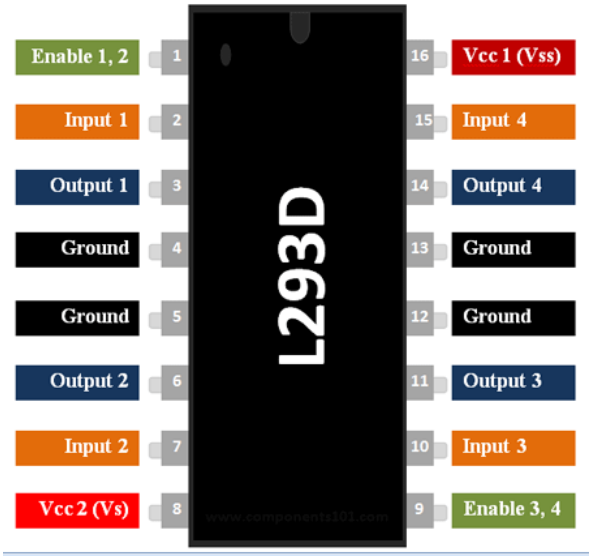
Typically, in an IR sensor module, an IR transmitter or IR Led sends an infrared signal in certain frequency compatible with an IR receiver. IR sensors are typically used for obstacle detection or for distance measurement.

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.



L293D Motor Driver

The L293D is a popular motor driver IC designed for controlling DC motors or stepper motors in various electronic applications. It is specifically designed to drive inductive loads such as motors, making it suitable for robotics, mechatronics, and other projects. The IC provides bidirectional control for two motors, allowing them to be driven forward or backward with speed control. It incorporates built-in diodes to handle back electromotive force (EMF) generated by the motors, ensuring protection against potential damage. Additionally, the L293D offers separate enable and input control for each motor channel, allowing flexibility in motor control configurations.



N20 Gear Motor:

Specifications

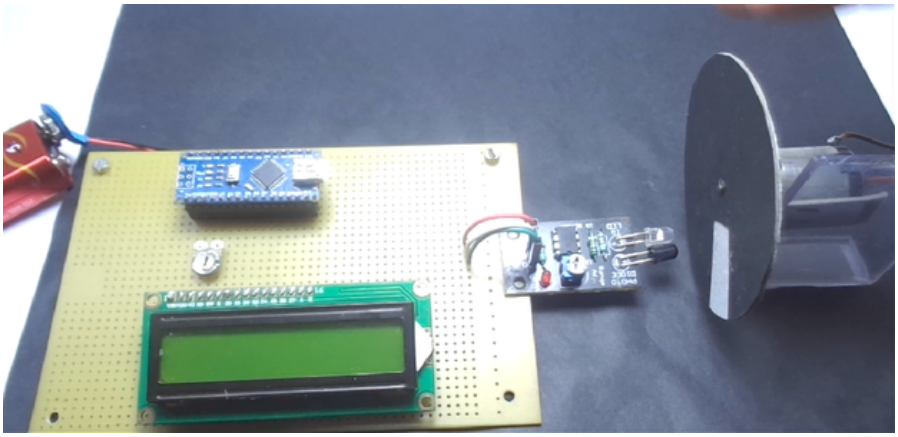
- size: 24 * 12 * 10mm
- Motor shaft diameter: 3mm D type shaft
- Motor shaft length: 9mm
- Length of wire: 8 ~ -14 cm with wire
- Voltage range: 3V-6V suitable 5V
- Motor parameters: When the power is applied 3V, the current is 0.04A, about 250 rpm
- When power is 3V, the current is 0.04A, about 250 rpm
- When power is 6V, the current is 0.04A, about 320 rpm

Features:

- Rated Voltage: 6V,
- Speed: 300RPM ±5%,
- Shaft Diameter: 3mm, D Shaft,
- Shaft Length: 9mm

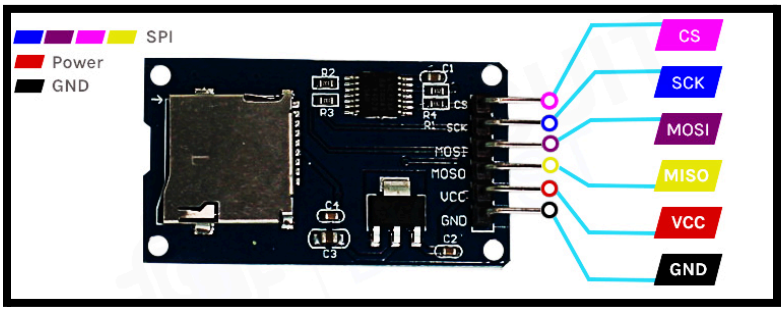
Encoder

N20 gear motors don't have any compatible rotating disc that can be used as reference like the disc used for tt motors. To measure the RPM of the motor we used a technique called Tachometer.



SD Card Module

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



16 x 2 LCD Display with I2C module

16X4 CHARACTER LCD 1604 GREEN LCD DISPLAY is a dot-matrix liquid crystal display module specially used for displaying letters, numbers, symbols, etc. Divided into 4-bit and 8-bit data transmission methods. 1604 Green Character LCD provides rich command settings: clear

display; cursor return to origin; display on/o; cursor on/o; display character ashes; cursor shift; display shift, etc. It can be used in any embedded systems, industrial device, security ,medical and hand-held equipment.

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

LCD Pin Configuration:

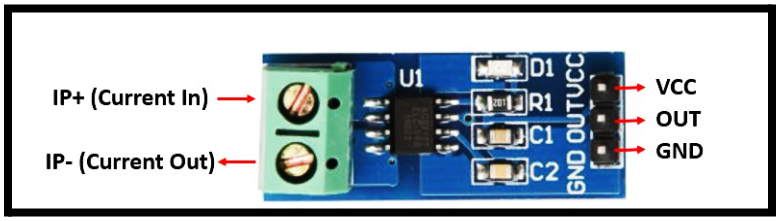
Pin No.	**Symbol**	**Description**
1	V_{SS}	Ground
2	V_{DD}	Power supply for logic
3	V_O	Contrast Adjustment
4	RS	Data/ Instruction select signal
5	R/W	Read/Write select signal
6	E	Enable signal
7~14	DB0~DB7	Data bus line
15	A	Power supply for B/L +
16	K	Power supply for B/L -

I2C ModulePin Configuration:

Pin Name	Pin Type	Pin Description
GND	Power	Ground
VCC	Power	Voltage Input
SDA	I2C Data	Serial Data
SCL	I2C Clock	Serial Clock
A0	Jumper	I2C Address Selection 1
A1	Jumper	I2C Address Selection 2
A2	Jumper	I2C Address Selection 3
Backlight	Jumper	Control Backlight of panel

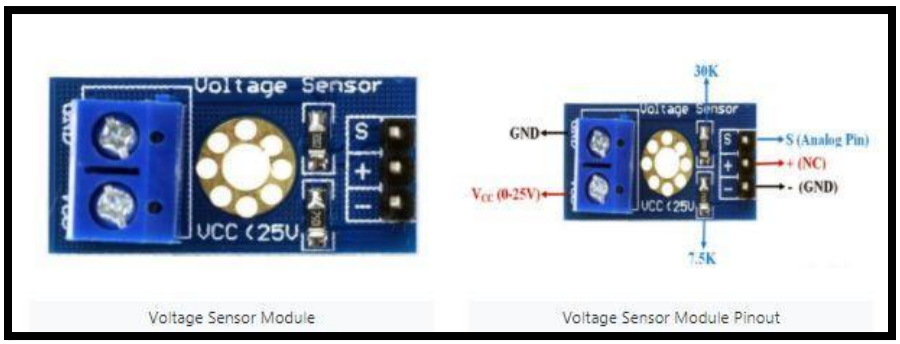
ACS712 Current Sensor

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



Voltage Sensor

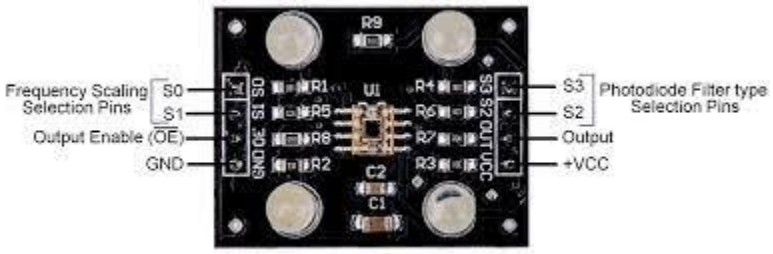
Voltage Sensor is a precise low-cost sensor for measuring voltage. It is based on the principle of resistive voltage divider design. It can make the red terminal connector input voltage 5 times smaller.



Voltage Sensor Module | Voltage Sensor Module Pinout

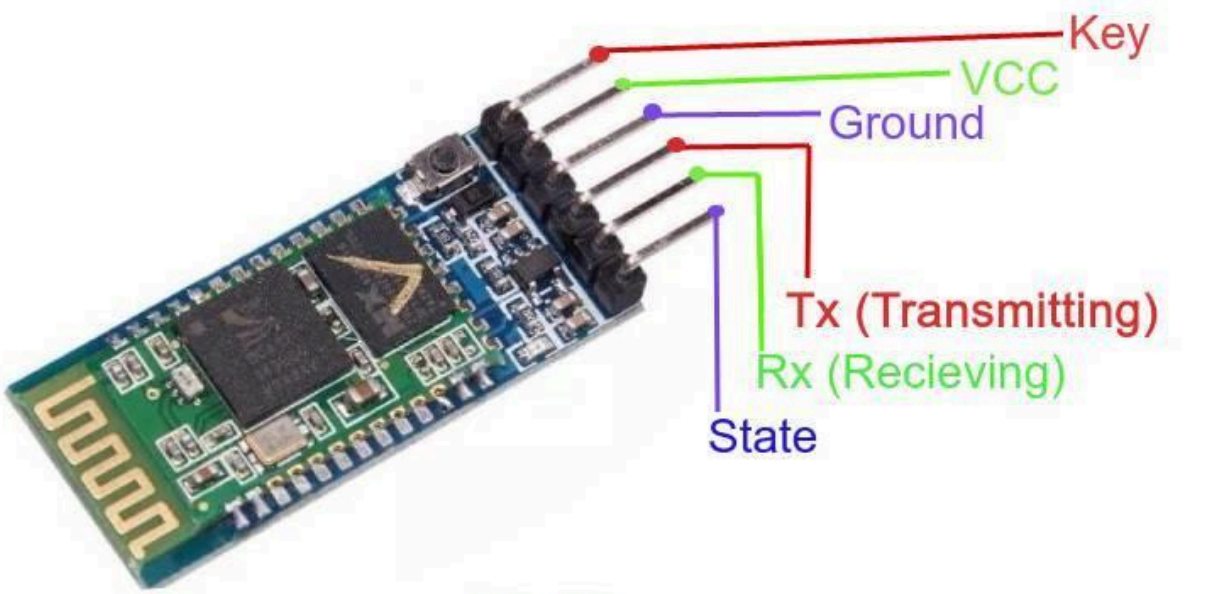
Color Sensor

The TCS3200 is a versatile color sensor module capable of detecting and quantifying colors in a variety of applications. It consists of an 8x8 array of photodiodes, which can be used to sense the intensity of red, green, and blue light, providing a basis for color identification. The sensor includes a white LED that can be used to illuminate the target surface uniformly, ensuring accurate color readings. The TCS3200 offers a programmable color filter array, enabling customization for specific color detection needs. It communicates with a microcontroller or other digital devices through a straightforward interface, providing color information in the form of digital signals. With its simplicity and reliability, the TCS3200 is widely utilized in projects such as color recognition systems, sorting applications, and various electronic devices where precise color sensing is essential.



Bluetooth Module

The HC-05 is a popular Bluetooth module widely used for wireless communication in electronic projects. Operating on Bluetooth 2.0 and EDR (Enhanced Data Rate) standards, the HC-05 module facilitates reliable and low-power communication between devices. It supports multiple communication modes, including Master and Slave modes, allowing it to establish connections with other Bluetooth devices seamlessly. The module features a built-in Bluetooth stack, which simplifies the integration process, and it can be configured easily using AT commands. The HC-05 offers a UART (Universal Asynchronous Receiver-Transmitter) interface, making it compatible with a variety of microcontrollers and embedded systems. With a moderate range, typically around 10 meters, and its ease of use, the HC-05 is commonly employed in applications such as wireless serial communication, Bluetooth-enabled projects, and the creation of interactive electronics with mobile devices.



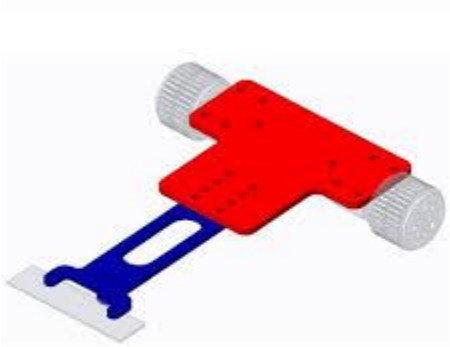
HC-05 (Bluetooth Module)

CHAPTER 3 MECHANICAL DESIGN

Mechanical design plays a very important role 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 and long lasting life of the robot. So, designing the robot in a most efficient way so that external condition or internal condition on that material used and that design are minimum.

3.1. Platform Design

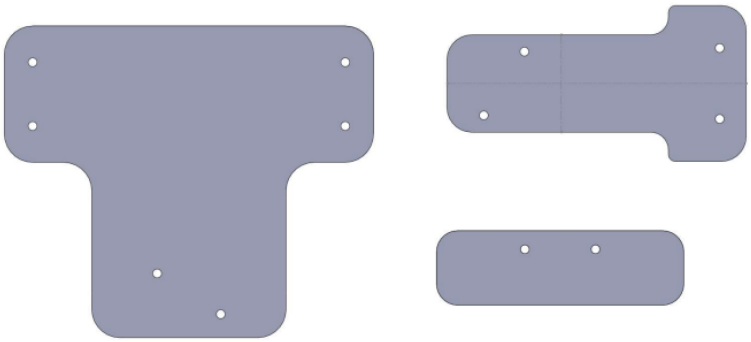
Our robot chassis as our platform is custom made design and we get idea from this chassis:

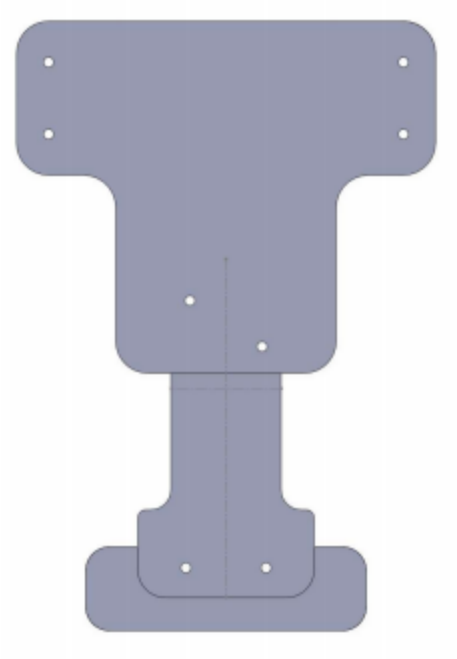


3.2 Material Selection and choices

Initially we were making a platform with an acrylic sheet but in order to reduce the price we used a PCB circuit board as platform or chassis.

3.3 CAD design





4.1 Component Selection

Component	Quantity
PCB Board double sided	1
Battery	1
Atmega328p	2
Crystal	2
Button	1
N20 Gear Motor	2
Tires	2
L293D	1
IR Sensors Array	1
Ultrasonic Sensor	1
SD Card Module	1
16x2 LCD	1
I2C Module	1

Voltage Sensor	1
Current Sensor	1
Encoder	1
Color Sensor	1
Bluetooth Module	1

4.2 Sensors along with specification and features from datasheet

Atmega328p

The ATmega328P is a popular microcontroller chip manufactured by Atmel, which is now a part of Microchip Technology. This microcontroller is widely used in various embedded systems and DIY electronics projects due to its versatility, features, and ease of use. Some features from data sheet are as follows:

· 32KB of programmable FLASH

· 1KB of EEPROM

· 2KB SRAM

· 10,000 Write and Erase Cycles for Flash and 100,000 for EEPROM

· Data retention for 20 years at 85°C and 100 years at 25°C

· Optional bootloader with lock bits

· In System Programming (ISP) by via bootloader

· True Read-While-Write operation

· Programming lock available for software security

· 23 programmable I/O lines

· 28 pin PDIP package

· 0-4 MHz at 1.8-5.5V

· 0-10 MHz at 2.7-5.5V

· 0-20 MHz at 4.5-5.5V

· Active Mode: 0.3 mA

· Power-down Mode: 0.1 μA

· Power-save Mode: 0.8 μA (Including 32 kHz RTC)

N20 Gear Motor

The N20 Gear Motor is a small and versatile DC (direct current) motor with an integrated gearbox, commonly used in various applications that require controlled and precise mechanical movement. Some features from data sheet are as follows:

· size: 24*12*10mm

· Motor shaft diameter: 3mm D type shaft

· Motor shaft length: 9mm

· Voltage range: 3V-6V suitable 5V

· Motor parameters: When the power is applied 3V, the current is 0.04A, about 250 rpm

· When power is 3V, the current is 0.04A, about 250 rpm

· When power is 6V, the current is 0.04A, about 320 rpm

· Rated Voltage: 6V,

· Speed: 300RPM ±5%,

· Shaft Diameter: 3mm, D Shaft,

· Product Dimensions: 1.97 x 1.97 x 1.18 inches

L293D

The L293D is a popular integrated circuit (IC) that serves as a motor driver or motor controller. Designed for driving small DC motors, stepper motors, and other inductive loads, the L293D is widely used in robotics, electronics projects, and various applications that involve motor control. Some features from data sheet are as follows:

· Wide Supply-Voltage Range: 4.5 V to 36 V

· Separate Input-Logic Supply

· Internal ESD Protection

· Thermal Shutdown

· High-Noise-Immunity Inputs

· Functionally Similar to SGS L293 and
· SGS L293D
· SGS L293D
· Peak Output Current 2 A Per Channel (1.2 A for L293D)
· Output Clamp Diodes for Inductive
· Transient Suppression (L293D)

IR Sensors Array

An Infrared (IR) Sensor Array is a collection of multiple infrared sensors organized in a systematic arrangement, often in the form of a matrix or array. Each sensor within the array is designed to detect infrared radiation, and the collective data from these sensors can be used to gather information about the surrounding environment. Some features from data sheet are as follows:

· Length x Width: 96.5mm x 20.3mm

· Distance Range: 1cm ~ 1.5cm

· Operating Voltage: 5V

· Output: Digital signal

· Sensor: TCRT5000L x 5

Ultrasonic Sensor

An ultrasonic sensor is a device that utilizes ultrasonic waves for distance measurement, object detection, and navigation in a variety of applications. It operates on the principle of emitting ultrasonic pulses and measuring the time it takes for the waves to reflect off an object and return to the sensor. Some features from data sheet are as follows:

· Operating voltage: +5V
· Theoretical Measuring Distance: 2cm to 450cm
· Practical Measuring Distance: 2cm to 80cm
· Accuracy: 3mm
· Measuring angle covered: <15°
· Operating Current: <15mA
· Operating Frequency: 40Hz

16x2 LCD

A 16x2 LCD is a type of alphanumeric display commonly used in electronic projects and devices to provide a visual interface. The term "16x2" refers to the display's size, indicating that it can show 16 characters in each of the two rows. Some features from data sheet are as follows:

- 16 Characters x 2 Lines
- Blue Backlight
- 5x7 Dot Matrix Character + Cursor
- HD44780 Equivalent LCD Controller/driver Built-
- 4-bit or 8-bit MPU Interface
- Standard Type
- Works with almost any Microcontroller

I2C Module

The I2C (Inter-Integrated Circuit) module is a communication protocol and hardware interface that facilitates communication between multiple integrated circuits, microcontrollers, and other peripherals using a two-wire serial communication bus. The I2C module simplifies data transfer and control in various electronic systems. Some features from data sheet are as follows:

- Operating Voltage: 5V
- Backlight and Contrast is adjusted by potentiometer
- Serial I2C control of LCD display using PCF8574
- Come with 2 IIC interface, which can be connected by Dupont Line or IIC dedicated cable
- Compatible for 16×2 LCD
- This is another great IIC/I2C/TWI/SPI Serial Interface
- With this I2C interface module, you will be able to realize data display via only 2 wires.

Voltage Sensor

A voltage sensor is an electronic device designed to measure the electrical potential difference (voltage) between two points in an electrical circuit. It plays a crucial role in various applications where monitoring and control of voltage levels are essential. Some features from data sheet are as follows:

· INPUT
  1.GND
  2.0-25V

· OUTPUT
  1.Sense
  2.N/C
  3.GND

Current Sensor

A current sensor is an electronic device designed to measure the electric current flowing through a conductor in an electrical circuit. It plays a crucial role in various applications where monitoring and control of current levels are essential. Some features from data sheet are as follows:

· Measures both AC and DC current

· Available as 5A, 20A and 30A module

· Provides isolation from the load

Colour Sensor

A color sensor is an electronic device designed to detect and identify colors in the visible spectrum. It operates by measuring the intensity of light at different wavelengths or by analyzing the color components of light. Color sensors find applications in various fields, including industrial automation, robotics, quality control, and electronic devices. Some features from data sheet are as follows:

· S0: Output frequency scaling selection input (along with S1)

· S1: Output frequency scaling selection input (along with S0)

· OE: Output Enable – if held low, the output of the module is turned on

· GND: Ground – connect to 0V

· S3: Photodiode type (along with S2)

· S2: Photodiode type (along with S3

· OUT: Output – A square wave appears here showing the intensity of the detected colour

· VCC: Power – connect to 2.7V-5.5V

Bluetooth Module

A Bluetooth module is a compact electronic device that integrates Bluetooth wireless technology into various electronic systems, enabling wireless communication between devices. These modules facilitate short-range data transmission and are widely used in applications such as wireless audio streaming, data transfer between devices, and IoT (Internet of Things) connectivity. Some features from data sheet are as follows:

· Typical -80dBm sensitivity

· Up to +4dBm RF transmit power

· Low Power 1.8V Operation ,1.8 to 3.6V I/O

· PIO control

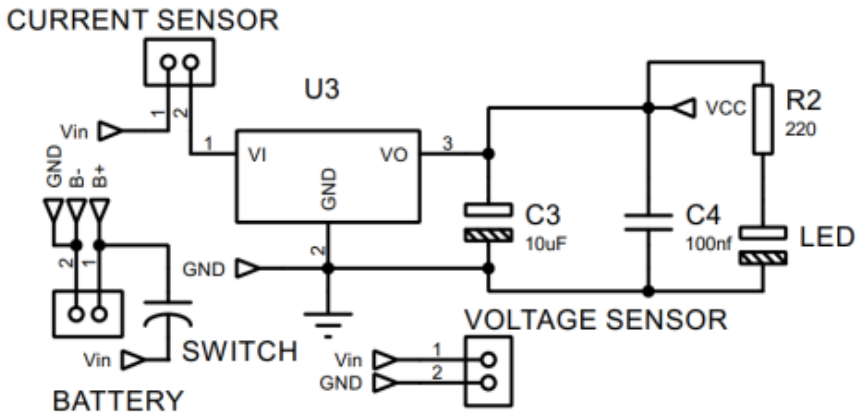
· UART interface with programmable baud rate

· With integrated antenna

4.3 Power requirements and power supply design

The power requirements for this system are specified as 5 volts (5V) and 12 volts (12V). In electronics, voltage is a crucial parameter that signifies the electric potential difference. The 5V requirement is common for low-power components such as microcontrollers and sensors, typically associated with digital circuits. On the other hand, the 12V requirement is intended for higher-power devices like motors and certain electronic components that demand a greater supply of electrical energy for efficient operation.

Schematic Design of power supply



4.4 Motor Selection Current/Voltage/Speed/Torque

N20 gear motors are a popular choice for various applications due to their compact size, high torque output, and versatility. Their small form factor makes them suitable for space-constrained environments, while their ability to deliver significant torque makes them valuable in applications requiring rotational force. The gear reduction mechanism in N20 motors enhances precision and accuracy, making them suitable for tasks where controlled motion is crucial. Additionally, these motors are cost-effective, energy-efficient, and exhibit low noise and vibration. With a wide range of applications, including robotics, automation, and electronics, N20 gear motors offer a reliable and practical solution for projects with diverse requirements.

· Maximum current 0.70 amperes

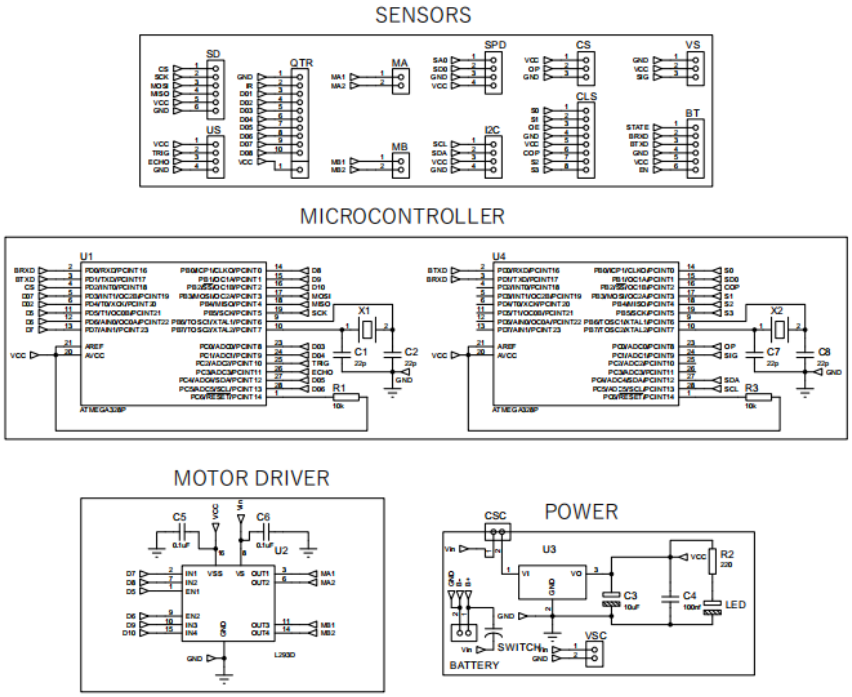
· Maximum voltage 12 volts

· Maximum speed 320 rotations per minute

· Maximum torque 27 kg.mm

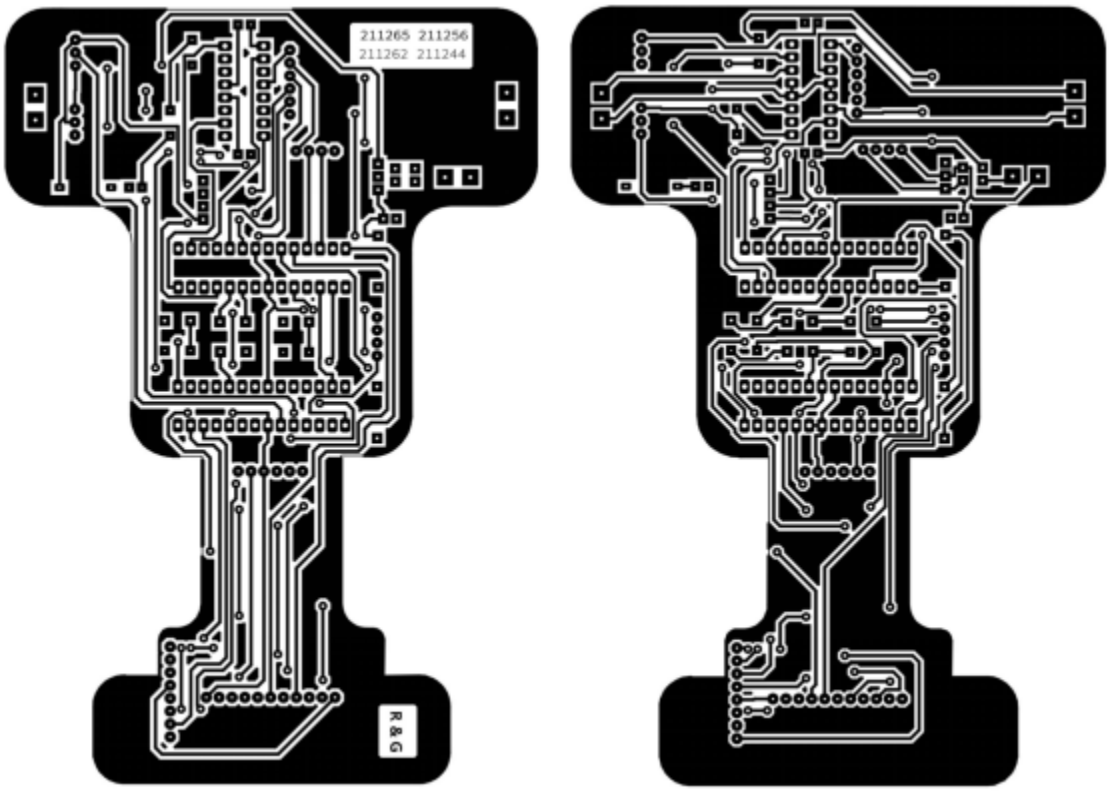
4.5 Deliverable of complete electronics design with A4 size Schematic and PCB with discussion

Circuit schematic



PCB design

PCB is designed in Proteus 8 professional software according to the given settings given below:



o Trace style T25

o Via style V60

o Boundary T25

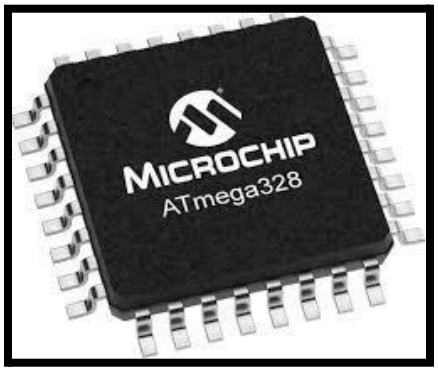
o Clearance 30th

o Pad hole octagonal

CHAPTER 5 SOFTWARE/FIRMWARE DESIGN

5.1 Controller Selections with features

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



Parametrics:

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

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

Advantages:

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

Disadvantages:

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

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

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

5.2 Software Design details & user Requirements

Components

Components that we used it the software design

- Ultrasonic sensor
- SD card module
- N20 gear motor
- IR sensors array
- L293D motor driver
- Tachometer
- 16x2 LCD display
- Current sensor.
- Voltage sensor
- Color sensor
- Bluetooth module
- I2C Display module

Inputs

- Left IR Sensor (LS)
- Right IR Sensor (RS)
- Centre IR Sensor (CS)
- Ultrasonic Sensor
- Voltage sensor
- Current sensor
- Encoder (Tachometer)
- Color sensor

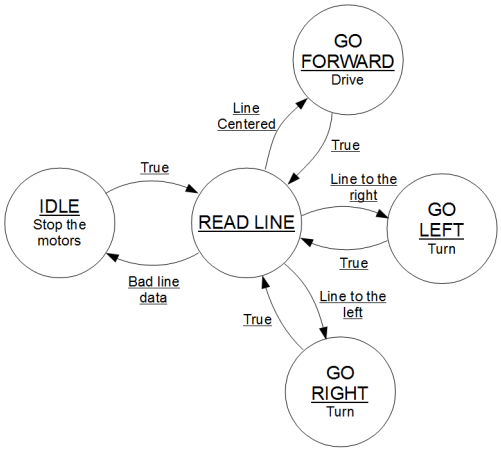
Outputs

- Left Motor
- Right Motor
- SD card
- 16 X 2 LCD display
- I2C display module
- bluetooth module
- L293D motor driver

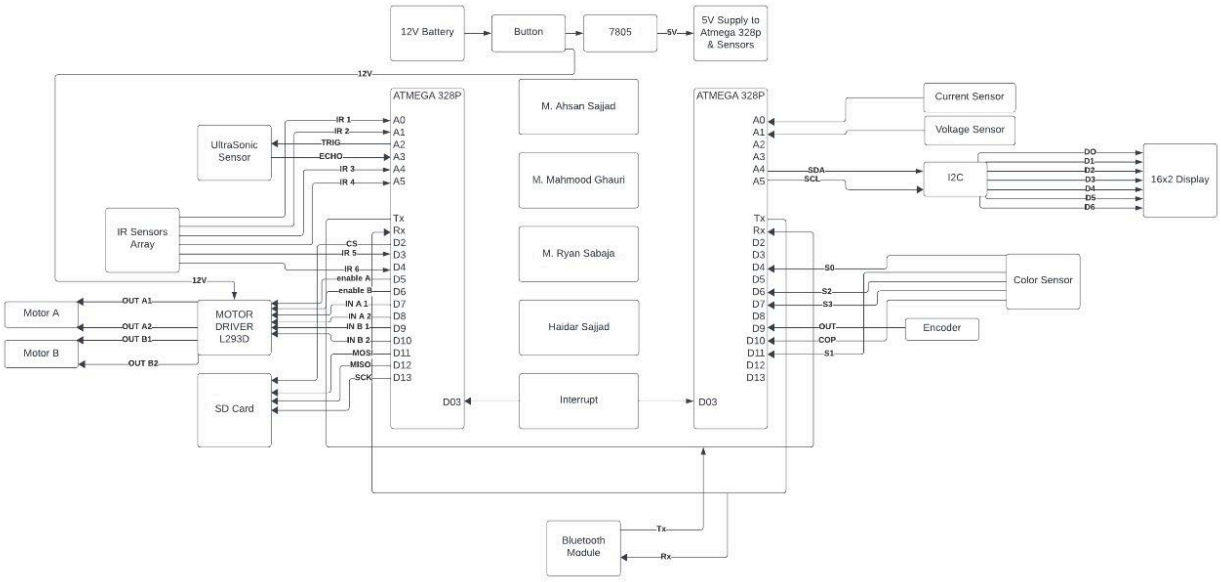
Communication

- bluetooth module
- SD card

5.3 State Machine & System flow diagram



5.4 Detailed block 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**

6.1 Integrations and testing all hardware and software component separately

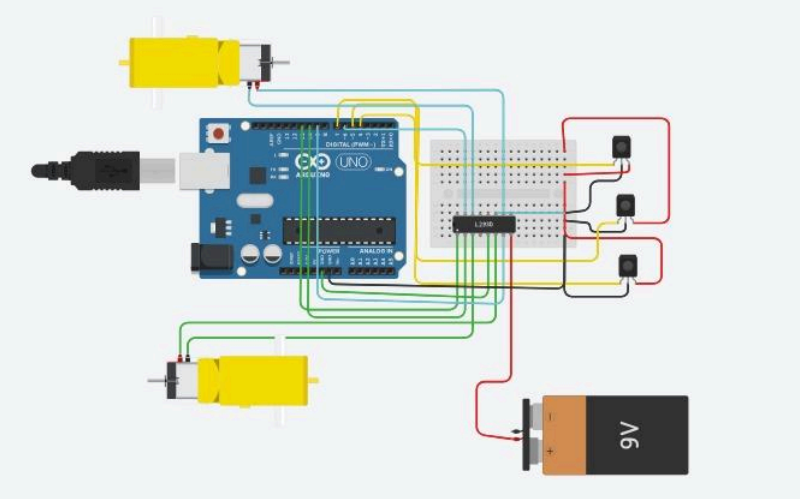
3 IR Sensors & Motor Driver:

Code

```
//Line Tracking IO define
#define LT_R !digitalRead(A0)
#define LT_M !digitalRead(A1)
#define LT_L !digitalRead(A4)

#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 10

#define carSpeed 150

void forward(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("go forward!");
}

void left(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("go left!");
```

```
}

void right(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("go right!");
}

void stop(){
   digitalWrite(IN1, LOW);
   digitalWrite(IN2, LOW);
   digitalWrite(IN3, LOW);
   digitalWrite(IN4, LOW);
   Serial.println("Stop!");
}

void setup(){
  Serial.begin(9600);
  pinMode(LT_R,INPUT);
  pinMode(LT_M,INPUT);
  pinMode(LT_L,INPUT);
}

void loop() {
  if(LT_M){
    forward();
  }
  else if(LT_R) {
    right();
    while(LT_R);
  }
  else if(LT_L) {
    left();
    while(LT_L);
  }
```

```
  else{
    stop();
  }
}
```

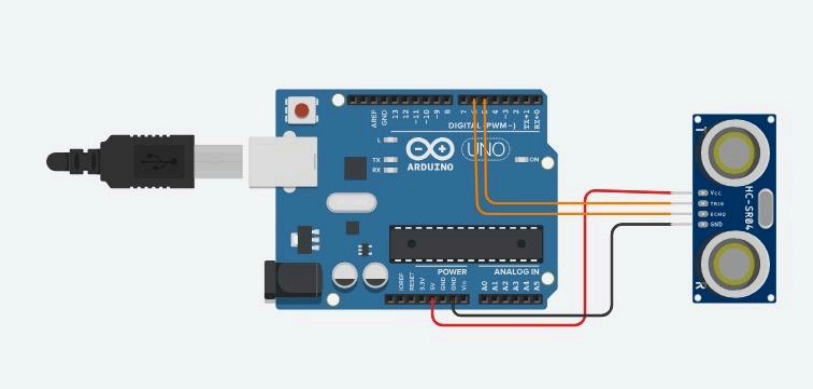


Ultrasonic Sensor:

Code

```
#define trigPin A2
#define echoPin A3

float duration, distance;

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {

  // Write a pulse to the HC-SR04 Trigger Pin

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
```

```
  digitalWrite(trigPin, LOW);

  // Measure the response from the HC-SR04 Echo Pin

  duration = pulseIn(echoPin, HIGH);

  // Determine distance from duration
  // Use 343 metres per second as speed of sound

  distance = (duration / 2) * 0.0343;

  // Send results to Serial Monitor

  Serial.print("Distance = ");
  if (distance >= 400 || distance <= 2) {
     Serial.println("Out of range");
  }
  else {
    Serial.print(distance);
    Serial.println(" cm");
    delay(500);
  }
  delay(500);
}
```

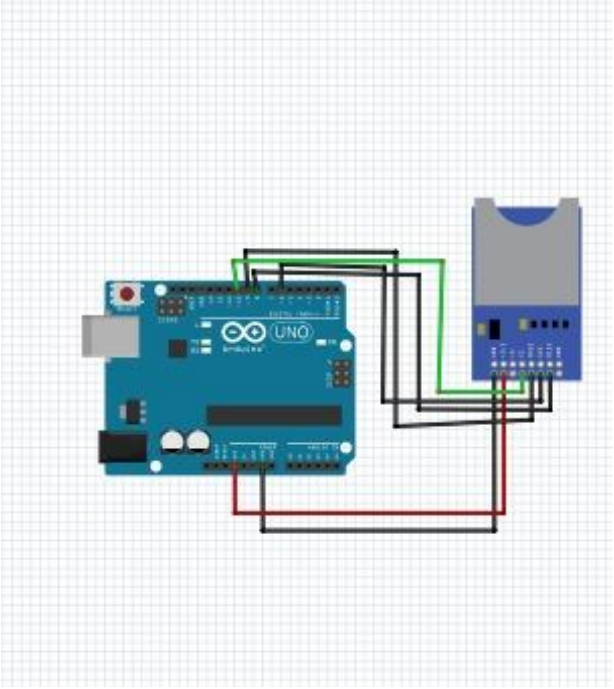


SD Card Module:

Code

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

File myFile;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port
only
  }


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

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done.");

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

  // if the file opened okay, write to it:
  if (myFile) {
    Serial.print("Writing to test.txt...");
    myFile.println("testing 1, 2, 3.");
    // close the file:
    myFile.close();
    Serial.println("done.");
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
```

```
  }

  // re-open the file for reading:
  myFile = SD.open("test.txt");
  if (myFile) {
    Serial.println("test.txt:");

    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
}

void loop() {
  // nothing happens after setup
}
```



Voltage Sensor:

Code

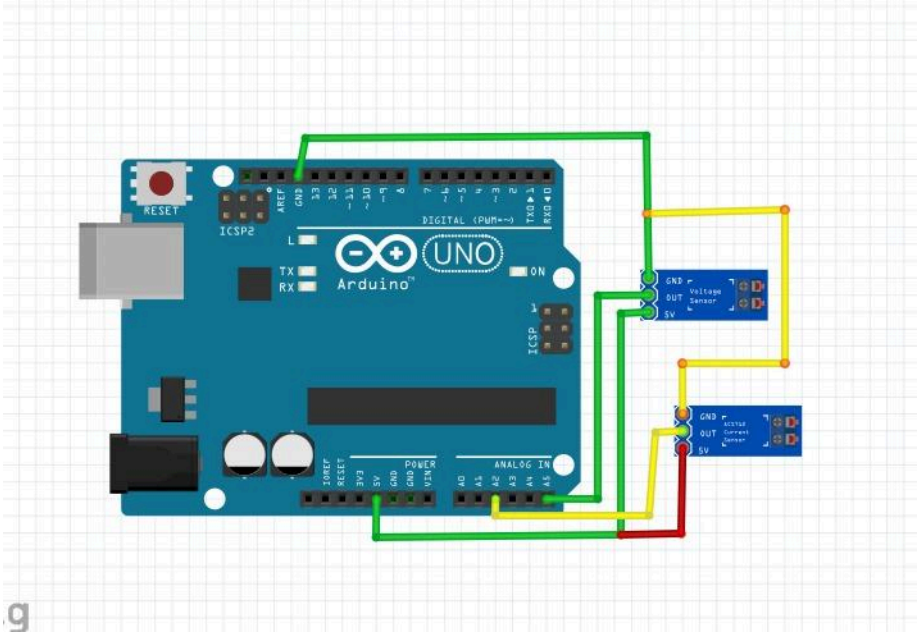
```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage
(0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

Current Sensor:

Code

```
// Variables for Measured Voltage and Calculated Current
double Vout = 0;
double Current = 0;

// Constants for Scale Factor
// Use one that matches your version of ACS712

//const double scale_factor = 0.185; // 5A
const double scale_factor = 0.1; // 20A
//const double scale_factor = 0.066; // 30A

// Constants for A/D converter resolution
// Arduino has 10-bit ADC, so 1024 possible values
// Reference voltage is 5V if not using AREF external reference
// Zero point is half of Reference Voltage

const double vRef = 5.00;
```

```
const double resConvert = 1024;
double resADC = vRef/resConvert;
double zeroPoint = vRef/2;


void setup(){
  Serial.begin(9600);
}

void loop(){

  // Vout is read 1000 Times for precision
  for(int i = 0; i < 1000; i++) {
    Vout = (Vout + (resADC * analogRead(A1)));
    delay(1);
  }

  // Get Vout in mv
  Vout = Vout /1000;

  // Convert Vout into Current using Scale Factor
  Current = (Vout - zeroPoint)/ scale_factor;

  // Print Vout and Current to two Current = ");

  Serial.print("Vout = ");
  Serial.print(Vout,2);
  Serial.print(" Volts");
  Serial.print("\t Current = ");
  Serial.print(Current,2);
  Serial.println(" Amps");

  delay(1000);

}
```



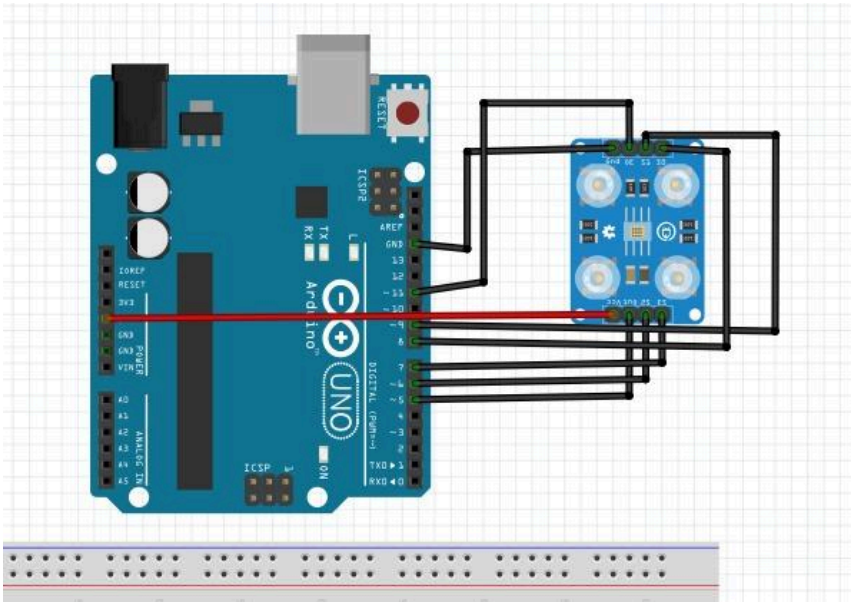
Color Sensor & Bluetooth Module:

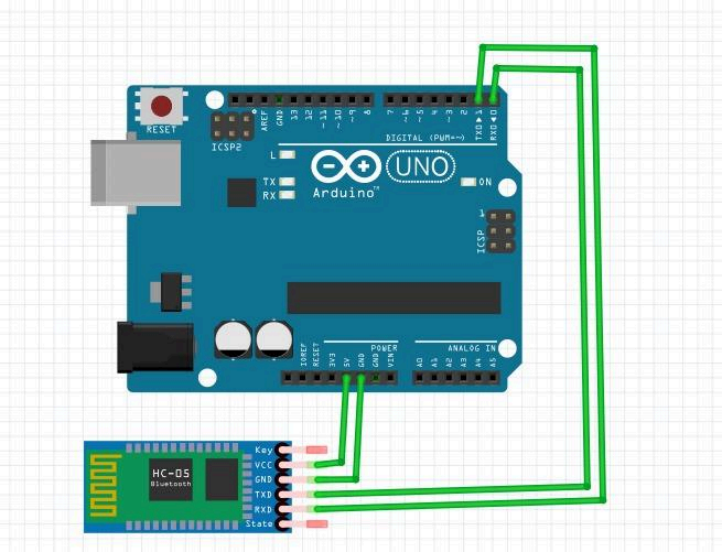
Code

```
// TCS230 or TCS3200 pins wiring to Arduino
#define S0 4
#define S1 5
#define S2 6
#define S3 7
#define sensorOut 8

// Stores frequency read by the photodiodes
int red = 0;
int green = 0;
int blue = 0;

void setup() {
  // Setting the outputs
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);

  // Setting the sensorOut as an input
  pinMode(sensorOut, INPUT);
```

```
  // Setting frequency scaling to 20%
  digitalWrite(S0,HIGH);
  digitalWrite(S1,HIGH);

  Serial.begin(9600);
}

void loop(){
  color();
  if(red<blue && red<green && red<25){
    Serial.print("Color detected : RED\n\tFrequency: ");
    Serial.println(red);
    delay(100);
  }
  else if(blue < red && blue < green && blue<25) {
    Serial.print("Color detected : BLUE\n\tFrequency: ");
    Serial.println(blue);
    delay(100);
  }
  else if (green < red && green < blue && green<25) {
    Serial.print("Color detected : GREEN\n\tFrequency: ");
    Serial.println(green);
    delay(100);
  }
  else {
    Serial.print("NO COLOR DETECTED\n");
    Serial.println(red);
    Serial.println(blue);
    Serial.println(green);
    delay(100);
  }

}

void color() {
  // Setting RED (R) filtered photodiodes to be read
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
```

```
  // Reading the output frequency
  red = pulseIn(sensorOut, LOW);
  delay(100);

  // Setting GREEN (G) filtered photodiodes to be read
  digitalWrite(S2,HIGH);
  digitalWrite(S3,HIGH);
  // Reading the output frequency
  green = pulseIn(sensorOut, LOW);
  delay(100);

  // Setting BLUE (B) filtered photodiodes to be read
  digitalWrite(S2,LOW);
  digitalWrite(S3,HIGH);
  // Reading the output frequency
  blue = pulseIn(sensorOut, LOW);
  delay(100);
}
```



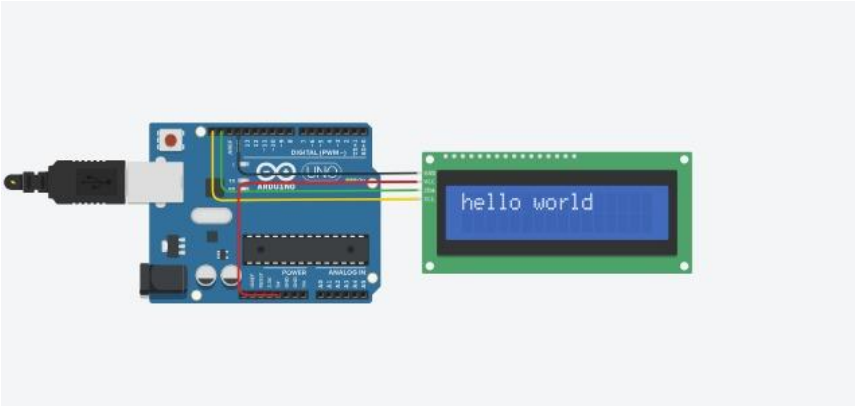


I2C Display Module & 16x2 LCD Display:

Code

```
/*I2C_scanner
  This sketch tests standard 7-bit addresses.
  Devices with higher bit address might not be seen properly.*/

#include <Wire.h>

void setup() {
  Wire.begin();

  Serial.begin(9600);
  while (!Serial);
  Serial.println("\nI2C Scanner");
}

void loop() {
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for (address = 1; address < 127; address++ ) {
```

```
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0) {
      Serial.print("I2C device found at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.print(address, HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error == 4) {
      Serial.print("Unknown error at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.println(address, HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);
}
```



Encoder (Tachometer):

Code

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(12,11,6,5,4,3);
float value=0;
float rev=0;
int rpm;
int oldtime=0;
int time;

void isr() //interrupt service routine
{
  rev++;
}

void setup()
{
  lcd.begin(16,2);                //initialize LCD
  attachInterrupt(0,isr,RISING);  //attaching the interrupt
}

void loop()
{
  delay(1000);
  detachInterrupt(0);           //detaches the interrupt
  time=millis()-oldtime;        //finds the time
  rpm=(rev/time)*60000;         //calculates rpm
  oldtime=millis();             //saves the current time
  rev=0;
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("__TACHOMETER__");
  lcd.setCursor(0,1);
  lcd.print(     rpm);
  lcd.print(" RPM");
  lcd.print("   ");
  attachInterrupt(0,isr,RISING);
}
```

Compiled Arduino code:

Driver Controller (Atmega 328p)

```
#include <SPI.h>
#include <SD.h>
File myFile;

#define trigPin A2
#define echoPin A3

//Line Tracking IO define
#define LT_R !digitalRead(A0)
#define LT_M !digitalRead(A1)
#define LT_L !digitalRead(A4)

#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 10

#define carSpeed_R 80
#define carSpeed_L 80

float duration, distance;

void forward(){
  analogWrite(ENA, carSpeed_R);
  analogWrite(ENB, carSpeed_L);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("go forward!");
}

void left(){
  analogWrite(ENA, carSpeed_R);
```

```
  analogWrite(ENB, carSpeed_L);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("go left!");
}

void right(){
  analogWrite(ENA, carSpeed_R);
  analogWrite(ENB, carSpeed_L);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("go right!");
}

void stop(){
   analogWrite(ENA, carSpeed_R);
   analogWrite(ENB, carSpeed_L);
   digitalWrite(IN1, LOW);
   digitalWrite(IN2, LOW);
   digitalWrite(IN3, LOW);
   digitalWrite(IN4, LOW);
   Serial.println("Stop!");
}

void get_distance(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) * 0.0343;
}

void isr() //interrupt service routine
```

```
{
  stop();
  delay(3000);
}

void setup(){
  Serial.begin(9600);
  pinMode(LT_R,INPUT);
  pinMode(LT_M,INPUT);
  pinMode(LT_L,INPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  while (!Serial) {;}
  Serial.print("Initializing SD card...");
  if (!SD.begin(2)) {
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done.");

  attachInterrupt(1,isr,RISING);  //attaching the interrupt
}

void loop() {
  get_distance();
  if(distance<=6){
    stop();

    myFile = SD.open("test.txt", FILE_WRITE);
    if (myFile) {
      Serial.print("Writing to test.txt...");
      myFile.println("S");
      Serial.println("done.");
    }
    else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
    }
```

```
}
else if(LT_M){
  forward();

  myFile = SD.open("test.txt", FILE_WRITE);
  if (myFile) {
    Serial.print("Writing to test.txt...");
    myFile.println("F");
    Serial.println("done.");
  }
  else {
  // if the file didn't open, print an error:
  Serial.println("error opening test.txt");
  }

}
else if(LT_R) {
  right();

  myFile = SD.open("test.txt", FILE_WRITE);
  if (myFile) {
    Serial.print("Writing to test.txt...");
    myFile.println("R");
    Serial.println("done.");
  }
  else {
  // if the file didn't open, print an error:
  Serial.println("error opening test.txt");
  }

  while(LT_R);
}
else if(LT_L) {
  left();

  myFile = SD.open("test.txt", FILE_WRITE);
  if (myFile) {
    Serial.print("Writing to test.txt...");
```

```
      myFile.println("L");
      Serial.println("done.");
    }
    else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
    }

    while(LT_L);
  }
  else{
    stop();

    myFile = SD.open("test.txt", FILE_WRITE);
    if (myFile) {
      Serial.print("Writing to test.txt...");
      myFile.println("S");
      Serial.println("done.");
    }
    else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
    }

  }
  myFile.close();
}
```

Sensors reading Controller (Atmega 328p)

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 4);

#define VOLTAGE_IN_PIN A1       //Voltage Sensor
#define CURRENT_IN_PIN A0       //Current Sensor

// TCS230 or TCS3200 pins wiring to Arduino
#define S0 8
#define S1 11
#define S2 12
```

```
#define S3 13
#define sensorOut 10

#define irSensorPin 3

// Stores frequency read by the photodiodes
int red = 0;
int green = 0;
int blue = 0;

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

//For Current Sensor
double Vout = 0;                // Variables for Measured Voltage and
Calculated Current
double Current = 0;
const double scale_factor = 0.1;// 20A scaling factor
const double vRef = 5.00;       // Constants for A/D converter
resolution
const double resConvert = 1024; // Arduino has 10-bit ADC, so 1024
possible values
double resADC = vRef/resConvert;// Reference voltage is 5V if not using
AREF external reference
double zeroPoint = vRef/2;      // Zero point is half of Reference
Voltage

//For Speed meaasurement in m/s
const int whiteTapeThreshold = 800;             // Adjust this threshold
based on your sensor readings
int lastSensorState = HIGH;                     // Initialize the last
state of the sensor
unsigned long lastTime = 0;                     // Time when the last
```

```
state change occurred
volatile float speed = 0;                // speed calculated
const float distancePerRevolution = 135.0;  // Distance covered in one
revolution (in millimeters)

#define LED 4
void isr() //interrupt service routine
{
  digitalWrite(LED,HIGH);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("INTERRUPT");
  delay(3000);
  digitalWrite(LED,LOW);
  lcd.clear();
}

void setup(){
  // Setting the outputs
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);

  // Setting the sensorOut as an input
  pinMode(sensorOut, INPUT);

  pinMode(irSensorPin, INPUT);    // for speed measurement

  pinMode(LED, OUTPUT);
  attachInterrupt(1,isr,RISING);  //attaching the interrupt

  // Setting frequency scaling to 20%
  digitalWrite(S0,HIGH);
  digitalWrite(S1,HIGH);

  // Setup Serial Monitor
  Serial.begin(9600);
  lcd.init();
```

```
  lcd.backlight();
  Serial.println("DC Voltage & Current Test & Colour Detection");
}

void loop(){
  adc_value = analogRead(VOLTAGE_IN_PIN);                                  //
Read the Analog Voltage Input
  adc_voltage  = (adc_value * ref_voltage) / 1024.0;                       //
Determine voltage at ADC input
  in_voltage = adc_voltage / (R2/(R1+R2));                                 //
Calculate voltage at divider input

  for(int i = 0; i < 1000; i++) {                                          //
Vout is read 1000 Times for precision
    Vout = (Vout + (resADC * analogRead(CURRENT_IN_PIN)));
    delay(1);
  }
  Vout = Vout /1000;                                                       //
Get Vout in mv
  Current = (Vout - zeroPoint)/ scale_factor;                              //
Convert Vout into Current using Scale Factor

  color();
  speed_cal();
  lcd.clear();

  if(red<blue && red<green && red<25){
    Serial.print("Input Voltage = ");                                      //
Print results of Voltage Sensor in Serial Monitor
    Serial.println(in_voltage, 2);
    Serial.print("\t Current = ");                                         //
Print results of Current Sensor in Serial Monitor
    Serial.print(Current,2);
    Serial.println(" Amps");
    Serial.print("\t Speed = ");                                          //
Print speed of robot in Serial Monitor
    Serial.print(speed,2);
    Serial.println(" m/s");
    Serial.print("Color detected : RED\n\tFrequency: ");
```

```
    Serial.println(red);

    lcd.setCursor(0, 0);
    lcd.print("V:");
    lcd.print(in_voltage);
    lcd.print("v ");
    lcd.print("C:");
    lcd.print(Current);
    lcd.print("A");
    lcd.setCursor(0, 1);
    lcd.print(speed);
    lcd.print("m/s ");
    lcd.print(" RED");
    delay(100);
  }
  else if(blue < red && blue < green && blue<25) {
    Serial.print("Input Voltage = ");                                    //
Print results of Voltage Sensor in Serial Monitor
    Serial.println(in_voltage, 2);
    Serial.print("\t Current = ");                                       //
Print results of Current Sensor in Serial Monitor
    Serial.print(Current,2);
    Serial.println(" Amps");
    Serial.print("\t Speed = ");                                        //
Print speed of robot in Serial Monitor
    Serial.print(speed,2);
    Serial.println(" m/s");
    Serial.print("Color detected : RED\n\tFrequency: ");
    Serial.println(red);
    Serial.print("Color detected : BLUE\n\tFrequency: ");
    Serial.println(blue);

    lcd.setCursor(0, 0);
    lcd.print("V:");
    lcd.print(in_voltage);
    lcd.print("v ");
    lcd.print("C:");
    lcd.print(Current);
    lcd.print("A");
```

```
    lcd.setCursor(0, 1);
    lcd.print(speed);
    lcd.print("m/s ");
    lcd.print(" BLUE");
    delay(100);
  }
  else if (green < red && green < blue && green<25) {
    Serial.print("Input Voltage = ");                                    //
Print results of Voltage Sensor in Serial Monitor
    Serial.println(in_voltage, 2);
    Serial.print("\t Current = ");                                       //
Print results of Current Sensor in Serial Monitor
    Serial.print(Current,2);
    Serial.println(" Amps");
    Serial.print("\t Speed = ");                                        //
Print speed of robot in Serial Monitor
    Serial.print(speed,2);
    Serial.println(" m/s");
    Serial.print("Color detected : RED\n\tFrequency: ");
    Serial.println(red);
    Serial.print("Color detected : GREEN\n\tFrequency: ");
    Serial.println(green);

    lcd.setCursor(0, 0);
    lcd.print("V:");
    lcd.print(in_voltage);
    lcd.print("v ");
    lcd.print("C:");
    lcd.print(Current);
    lcd.print("A");
    lcd.setCursor(0, 1);
    lcd.print(speed);
    lcd.print("m/s ");
    lcd.print(" GREEN");
    delay(100);
  }
  else {
    lcd.setCursor(0, 0);
    lcd.print("V:");
```

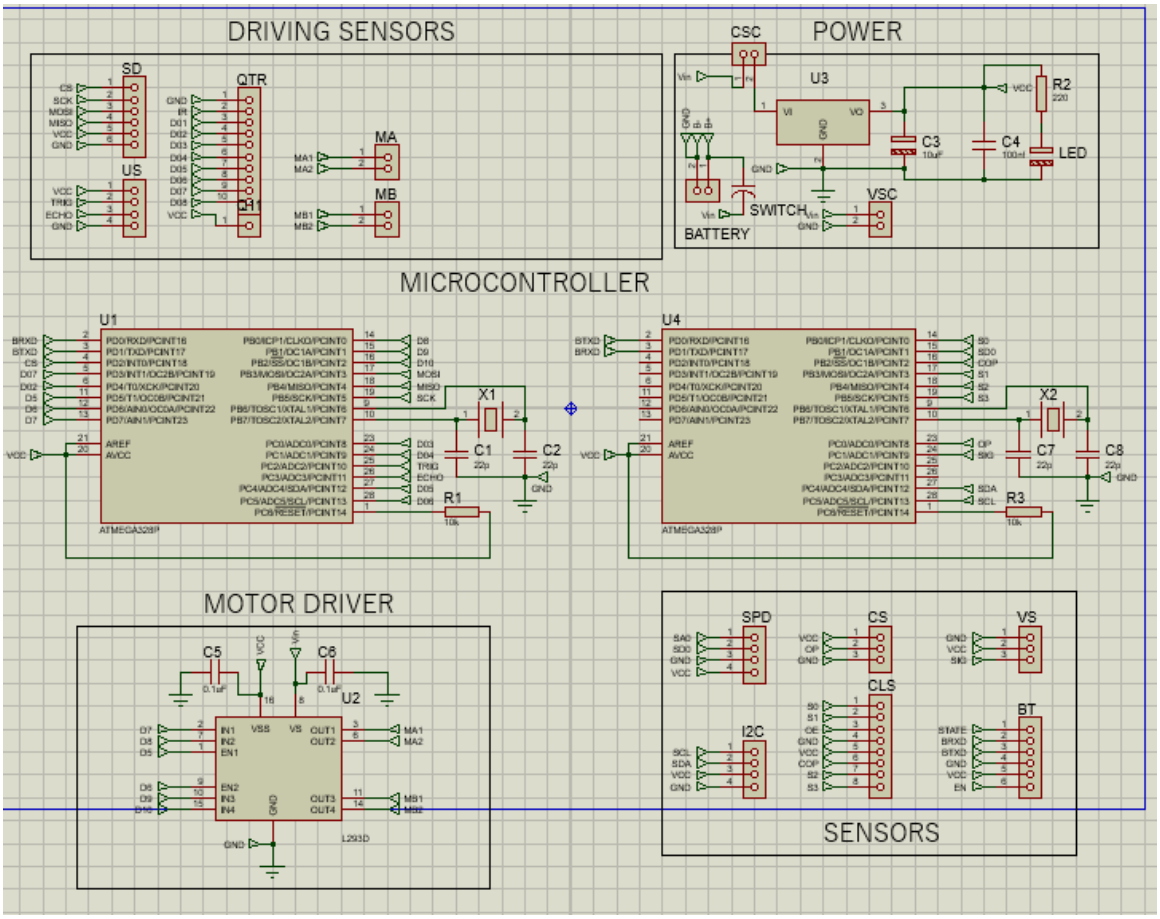
```
    lcd.print(in_voltage);
    lcd.print("v ");
    lcd.print("C:");
    lcd.print(Current);
    lcd.print("A");
    lcd.setCursor(0, 1);
    lcd.print(speed);
    lcd.print("m/s ");
    lcd.print(" NO");

    Serial.print("Input Voltage = ");                                    //
Print results of Voltage Sensor in Serial Monitor
    Serial.println(in_voltage, 2);
    Serial.print("\t Current = ");                                       //
Print results of Current Sensor in Serial Monitor
    Serial.print(Current,2);
    Serial.println(" Amps");
    Serial.print("\t Speed = ");                                        //
Print speed of robot in Serial Monitor
    Serial.print(speed,2);
    Serial.println(" m/s");
    Serial.print("Color detected : RED\n\tFrequency: ");
    Serial.println(red);
    Serial.print("NO COLOR DETECTED\n");
    Serial.println(red);
    Serial.println(blue);
    Serial.println(green);
    delay(10);
  }

  delay(100);
}

void color() {
  // Setting RED (R) filtered photodiodes to be read
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
  // Reading the output frequency
  red = pulseIn(sensorOut, LOW);
```

```
  delay(100);

  // Setting GREEN (G) filtered photodiodes to be read
  digitalWrite(S2,HIGH);
  digitalWrite(S3,HIGH);
  // Reading the output frequency
  green = pulseIn(sensorOut, LOW);
  delay(100);

  // Setting BLUE (B) filtered photodiodes to be read
  digitalWrite(S2,LOW);
  digitalWrite(S3,HIGH);
  // Reading the output frequency
  blue = pulseIn(sensorOut, LOW);
  delay(100);
}

void speed_cal(){
  int sensorState = digitalRead(irSensorPin);
  // Check for a falling edge (white tape detected)
  if (sensorState == LOW && lastSensorState == HIGH) {
    // Calculate the time since the last state change
    unsigned long currentTime = millis();
    unsigned long deltaTime = currentTime - lastTime;

    // Calculate Speed based on time taken per every revolution
    speed = distancePerRevolution/deltaTime;

    // Update the last time
    lastTime = currentTime;
  }
}
```

6.2 Simulation Schematic

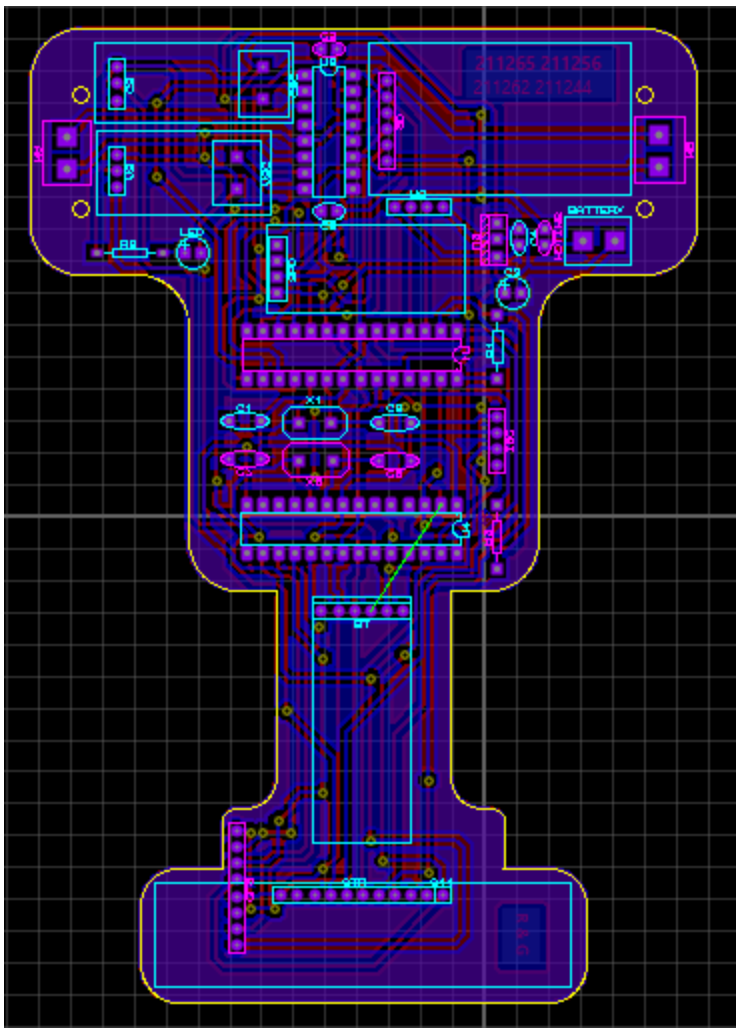
We did our simulations in Proteus ISIS software. For our task many packages of microcontroller and sensors were required so we firstly added the libraries and then we integrated them together

in our simulation according to the pinouts and block diagram we decided. Here is our diagram of our simulation in Proteus ISIS.

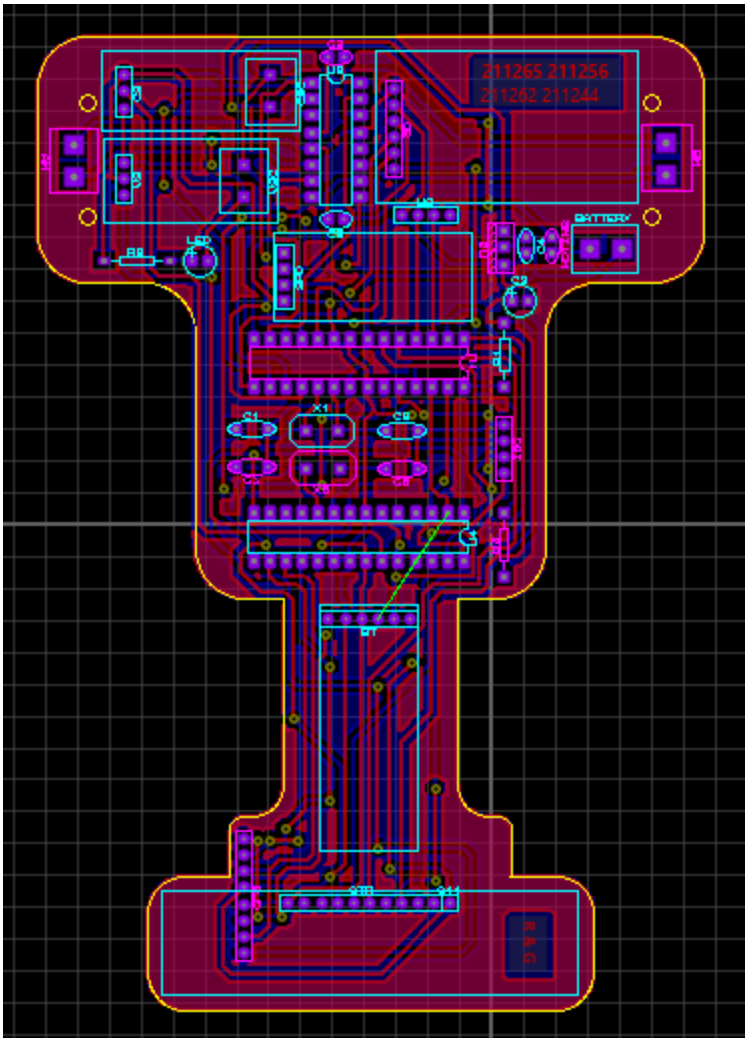


6.3 Simulation PCB

Bottom Copper



Top Copper



CHAPTER 7 SYSTEM TEST PHASE

7.1 Final testing

First stage

We first checked our PCB board that we fabricated. We fabricated PCB boards 3 times because the first time the placements of sensors were not matching in the upper layer and bottom layer. In the second attempt most of the paths braked. In the final attempt we successfully fabricated a double sided PCB board.

Second stage

In Second stage we checked our modules separately. Almost all of them were working properly but the motor driver we purchased first wasn't working properly so we had to replace it. then we integrated all the components

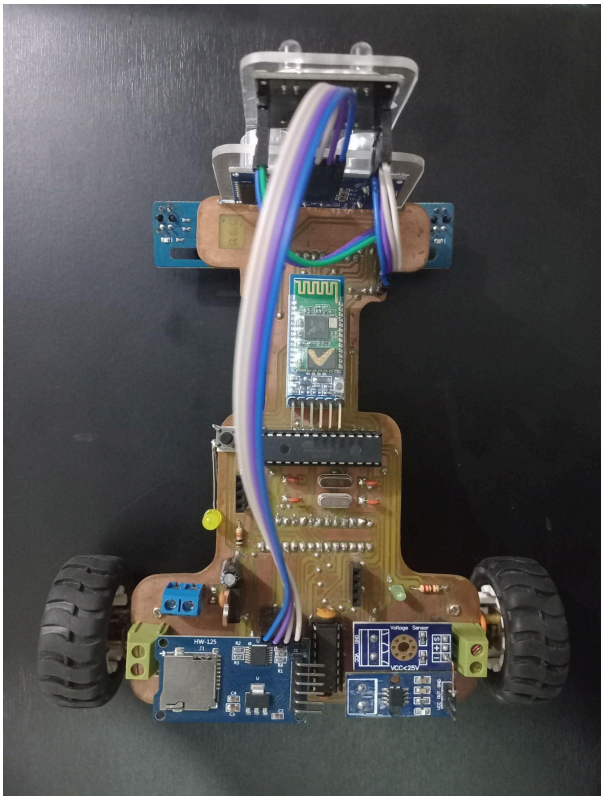
Third stage

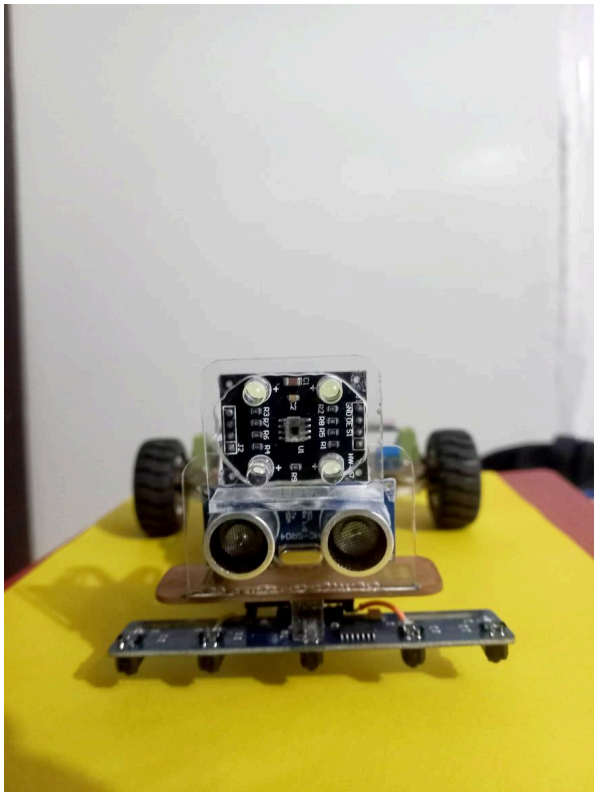
In the Third stage we integrated our modules together and some calibration of IR sensors & color sensor was needed to be done.

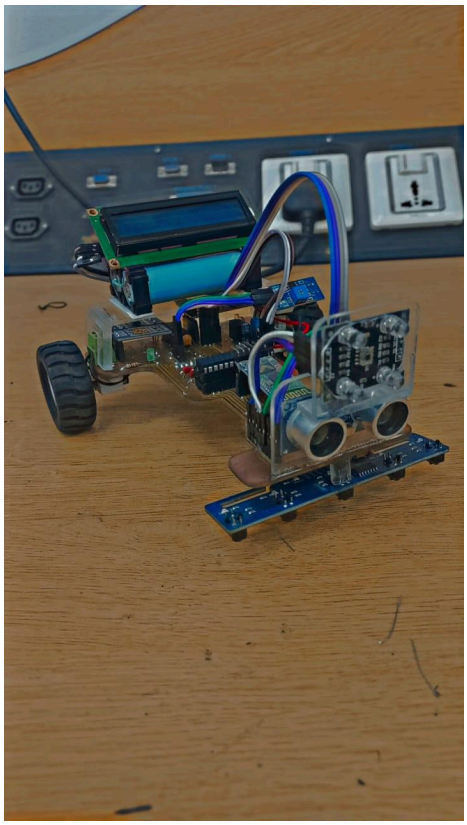
7.2 Things that are questionable and get burned again and again

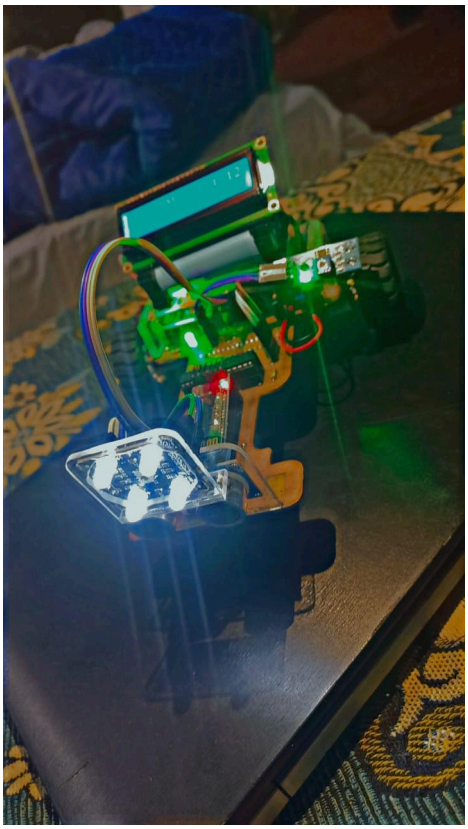
If the somehow direct 12V is supplied to Microcontrollers then they will be burned. Also if input voltage is greater than 14 volt then the motor driver will be burned. The accuracy of the color sensor and control of motor speed using the motor driver is questionable. The height of ir sensors plays an important role in its accuracy and ultrasonic sensor HC-SR04.

7.3 Project actual Pictures









CHAPTER 8 PROJECT MANAGEMENT

Team lead

As a group leader I worked with all my team members and helped them out in their work assigned. I managed to complete my portion of the project which was programming. I helped ahsan in hardware like cutting and drilling of PCB. I helped Haider in interfacing sensors and helped Ryan in routing of circuit in proteus.

8.1 Comment individually about success /failure of your project

As we had completed 90% of the project, so in my opinion we are 90% successful. We failed to move the robot back to starting position on detection of an obstacle using a path stored in the sd card. We had figured out how to read a file from the start but failed to read a file from the end.

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

Component	Price
PCB Board double sided	4500
Battery	700
Atmega 328p	2700
Crystal	80
Button	20
Jumper Wires	150
N20 Gear Motor	750
Tires	340
L293D	300
IR Sensors Array	500
Ultrasonic Sensor	200
SD Card Module	150
16x2 LCD	420
I2C Module	220

Voltage Sensor	100
Current Sensor	320
Encoder	120
Color Sensor	1200
Bluetooth Module	750
Total	**Rs. 13520**

8.3 Risk management that you learned

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

Several risks were encountered during this project, including PCB etching, which if not done correctly required us to purchase a new one from a store, which was inconvenient. Another is the drilling and soldering of PCB holes. Because our PCB was double-layered, PCB alignment was a major challenge for us. In Order to avoid issues when using acids, we took care of all the precautionary measures. Because many components were not available in proteus, we created them ourselves.

CHAPTER 8 FEEDBACK FOR PROJECT AND COURSE

I really like doing projects because projects tell you about real world problems that you will face. In theoretical problems we are solving problems in an ideal world. There is a huge difference between a real world problem and an ideal world problem.

I like that the course is designed more on a practical problem rather than theoretical based. But I disagree with the method of examination because handwritten code cant be verified so exam of this should be taken on a computer.

CHAPTER 9 DETAILED YOUTUBE VIDEO EXPLAINING OUR WORK OF EACH MEMBER

GITHUB:

github link	https://github.com/MahmoodGhauri/LFR-OA-

YOUTUBE:

Part:01 youtube link	https://www.youtube.com/watch?v=lqV07G4gr8Q
Part:02 youtube link	