LINE FOLLOWING ROBOT WITH MICROCONTROLLER

Group Members:

Faizan Ahmad Siddiqui	**(212003)**
Sania Ali	**(211194)**
Arooj Aftab	**(211209)**

BE MECHATRONICS (Session 2021-2025)

Project Supervisor

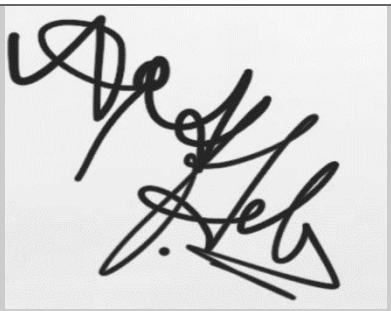
Sir Umer Farooq

Lecturer

DEPARTMENT OF MECHATRONICS ENGINEERING

FACULTY OF ENGINEERING

AIR UNIVERSITY, ISLAMABAD

Team Member Name	Role	Word count	Signature
Faizan Ahmad Siddiqui	**Team Leader**	7134	
Sania Ali	**Technical**	6032	
Arooj Aftab	**Project Manager**	10159	

Abstract

This project focuses on creating a Line Follower Robot using Arduino technology. The robot uses infrared (IR) sensors to detect and follow a line at different predefined distances. With an integrated input and a robust chassis that includes an Arduino board, a driver and a DC motor, the robot can continuously read sensor data and update the motor. The Arduino based control algorithm creates a feedback loop to enable changes to various graphics and environment. The project provides a source code that allows effective customization and support the search for additional feature such as speed control. Some of its applications includes industrial automation, warehousing, transportation and even in education as learning platforms. Overall, Line Follower Robot provide an easy to access learning platform for people interested in robotics and automation.

Table of Contents

Chapter 1-Prelimiaries

1.1 Proposal

Our purpose is to make the line follower robot which can follow the black line over the white surface on any given path in the minimum possible time. It displays the voltage, current, speed obstacle on LCD and save the data in SD card by SD card module. We have design the Embedded System for line follower robot that can follow the black line on the white surface at given path. It stops where black line will finish. For this we have used IR sensors which detect the black line H bridge which controls the working of the wheels, the Voltage sensor, current sensor, ultrasonic sensor and speed sensor show the data on LCD screen while SD card module store it in SD card.

1.2 Concept

The concept of line follower robot revolves around the integration of sensors, control systems and actuators to provide autonomous navigation along the predetermined path. The basic idea is to make the robot that can follow different lines on the surface, usually black line on the white surface. Infrared sensors are often used to detect the changes in interference, allowing the robot to adjust its movement depending on the desired direction while in motion. The control system usually implemented on the microcontroller platform such as Arduino, interprets sensor data and commands actuators (usually DC motors) to adjust the robot's orientation accordingly. The concept is versatile, providing opportunities for additional collaboration such as adaption protection and exploration of different controls to improve the robot's navigation performance.

1.3 Initial Feasibility

The initial feasibility for a line follower robot provides a good assessment of the viability and effectiveness of the project. Key consideration include the availability of essential components such as Infrared sensors, motors and microcontrollers. At first our project seems difficult to implement, especially on AVR programmer as we have used 3 IR sensors, ultrasonic sensor, voltage sensor, current sensors and storing all the information in SD card and showing all of them on LCD. But it became easier when we used Arduino IDE instead of AVR programmer.

1.4 Comparison Table of Products

IR Sensors

Product A	Product B	Selected
Units of IR sensors	Array of IR sensors	
Cheap	Costly	
Easily available	Not available everywhere	
An individual sensor can be replaced	As they are tightly attached by PCB designing so can't be replaced by new sensor	Product A
Can be spaced according to coding/desire	**Pre-defined equally spaced**	

Table 1.1 Comparison of IR sensors

Microcontroller

Product A	Product B	Selected
ATMEGA2560	Arduino UNO	
Costly	cheap	Product A
Not easily available	easily available everywhere	
Provide more ports	Provide Less ports	

Table 1.2 Comparison of Microcontrollers

1.5 Technical Standards

The speed of the motor is kept under 200rpm. We have used motor under speed of 200rpm using enable pins of motor driver. These values may decrease if it give too high speed.

1.6 Team Roles and Details

Team Members	Roles
Faizan Ahmed Siddiqui	Software Design
Sania Ali	Hardware Design
Arooj Aftab	Report writing

Table 1.3 Team Roles

1.7 Work Breakdown Structure

Phase	Work Distribution
Phase 1	Selection and finalization of components
Phase 2	Buying Components
Phase 3	Schematic design Proteus
Phase 4	PCB design on Proteus
Phase 5	Implementation on breadboard
Phase 6	Implementation on PCB
Phase 7	Final Testing
Phase 8	Completion of project
Phase 9	Report making

Table 1.4 Work Breakdown

1.8 Gantt Chart

CONTENTS	16th NOV	17th NOV	18th NOV	20th NOV	22nd NOV	25th NOV	30th NOV	8th DEC	14th DEC	15th DEC	16th DEC
Finalizing Components											
Buying Components											
Schematic on Proteus											
Breadboard Implementation											
PCB design on Proteus											
PCB Implementation											

Final Testing											
Project Completion											
Report Writing											

Table 1.5 Gantt Chart

1.9 Expected Budget

Components	Price- (Rs)	Quantity
ATMEGA2560	3500-4500	1
Wires (Male to female, male to male, female to female)	600-750	-
Sticker Sheet	40-50	5
Ultrasonic Sensor	300-400	1
glue	90-130	1
Voltage Sensor	300-450	1
Current Sensor	300-400	1
Double sided tape	150-200	1
Ferric chloride	200-300	200g
IR sensors	200-400	3
L298N motor driver	300-450	1
Robotic chassis (2 tires, 2 motors, bolts, nuts, and base)	900-1200	1
PCB	400-650	1
Bread board	200-250	1
18650 battery cells	200-400	1(Pair)
LCD Display	800-1000	1
Power bank module	400-550	1
Total	11000-13000	

Table 1.6 Estimated Budget

Chapter 2-Project Conception

2.1 Introduction

The term **'Robotics'** describes the science of robotics and automation. A robot is a programmable machine that can perform tasks same as human beings. Each robot has different degree of freedom. There are many different types of robots as there have different tasks for them to perform. Robots can perform some better task than human e.g. they work in an unpredictable environment to spot hazards like gas leaks. Also they deliver online orders, room services and even food packets during emergencies.

A **line follower robot** is a mobile machine that detects and follows line drawn on the ground. Usually the path is predefines and can be seen as a black line with different color white or not visible as a magnetic field. Of course, such a robot needs to know the lines coming from the infrared (IR) sensors attached to the bottom of the robot. The data is then transferred to the processor via a special switching bus. Therefore the processor will determine the appropriate command and send it to the driver, so the line follower robot will follow the path. They can be mostly used in manufacturing transportation, space exploration, mass production of consumer and commercial products.

2.2 Literature Review

Research papers

> Line Following is one of the most important aspects of robotics. A Line Following Robot is an autonomous robot which is able to follow either a black line that is drawn on the surface consisting of a contrasting color. It is designed to move automatically and follow the line. The robot uses arrays of optical sensors to identify the line, thus assisting the robot to stay on the track. The array of four sensor makes its movement precise and flexible. [1].
> It is usually developed for reducing risk factor for human work and increase comfort of any worker. High performance, high accuracy, lower labor cost and the ability to work in hazardous places have put robotics in an advantageous position over many other such technologies. In this paper a line tracer or follower has been presented which will trace a black line on a white surface or vice-versa. [2]

References

[1]https://www.researchgate.net/publication/327814718_PROJECT_REPORT_LINE_FOLLOWING_ROBOT

[2]https://www.ripublication.com/ijaerspl2019/ijaerv14n13spl_21.pdf

2.3 Product Specifications

We have used different components for the optimization of our project i.e. for **Line follower robot.** Their specifications are given below.

Arduino MEGA2560

The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. The Arduino MEGA 2560 is designed for projects that require more I/O lines, more sketch memory and more RAM. Its specifications are given below.

1. 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.
2. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.
3. It has operating voltage: 5V, DC current per I/O pin: 40mA, DC current per I/O pin: 40mA, flash Memory: 256 KB, 8KB used by bootloader, SRAM: 8 KB, EEPROM: 4KB.

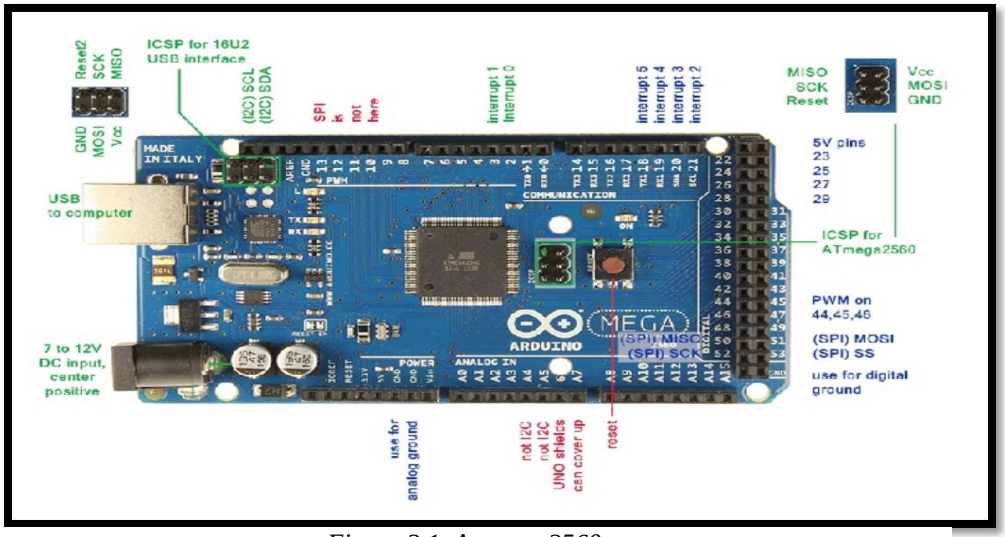


Figure 2.1 At mega 2560

Pin Configuration

Pin Number	Pin Name	Mapped Pin Name
1	PG5 (OC0B)	Digital pin 4 (PWM)
2	PE0 (RXD0/PCINT8)	Digital pin 0 (RX0)
3	PE1 (TXD0)	Digital pin 1 (TX0)
4	PE2 (XCK0/AIN0)	
5	PE3 (OC3A/AIN1)	Digital pin 5 (PWM)
6	PE4 (OC3B/INT4)	Digital pin 2 (PWM)
7	PE5 (OC3C/INT5)	Digital pin 3 (PWM)
8	PE6 (T3/INT6)	
9	PE7 (CLKO/ICP3/INT7)	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin 17 (RX2)
13	PH1 (TXD2)	Digital pin 16 (TX2)
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin 6 (PWM)
16	PH4 (OC4B)	Digital pin 7 (PWM)
17	PH5 (OC4C)	Digital pin 8 (PWM)
18	PH6 (OC2B)	Digital pin 9 (PWM)
19	PB0 (SS/PCINT0)	Digital pin 53 (SS)
20	PB1 (SCK/PCINT1)	Digital pin 52 (SCK)
21	PB2 (MOSI/PCINT2)	Digital pin 51 (MOSI)
22	PB3 (MISO/PCINT3)	Digital pin 50 (MISO)
23	PB4 (OC2A/PCINT4)	Digital pin 10 (PWM)
24	PB5 (OC1A/PCINT5)	Digital pin 11 (PWM)
25	PB6 (OC1B/PCINT6)	Digital pin 12 (PWM)

26	PB7 (OC0A/OC1C/PCINT7)	Digital pin 13 (PWM)
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	
30	RESET	RESET
31	VCC	VCC
32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin 49
36	PL1 (ICP5)	Digital pin 48
37	PL2 (T5)	Digital pin 47
38	PL3 (OC5A)	Digital pin 46 (PWM)
39	PL4 (OC5B)	Digital pin 45 (PWM)
40	PL5 (OC5C)	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42
43	PD0 (SCL/INT0)	Digital pin 21 (SCL)
44	PD1 (SDA/INT1)	Digital pin 20 (SDA)
45	PD2 (RXDI/INT2)	Digital pin 19 (RX1)
46	PD3 (TXD1/INT3)	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38
51	PG0 (WR)	Digital pin 41
52	PG1 (RD)	Digital pin 40

53	PC0 (A8)	Digital pin 37
54	PC1 (A9)	Digital pin 36
55	PC2 (A10)	Digital pin 35
56	PC3 (A11)	Digital pin 34
57	PC4 (A12)	Digital pin 33
58	PC5 (A13)	Digital pin 32
59	PC6 (A14)	Digital pin 31
60	PC7 (A15)	Digital pin 30
61	VCC	VCC
62	GND	GND
63	PJ0 (RXD3/PCINT9)	Digital pin 15 (RX3)
64	PJ1 (TXD3/PCINT10)	Digital pin 14 (TX3)
65	PJ2 (XCK3/PCINT11)	
66	PJ3 (PCINT12)	
67	PJ4 (PCINT13)	
68	PJ5 (PCINT14)	
69	PJ6 (PCINT 15)	
70	PG2 (ALE)	Digital pin 39
71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27
74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22
79	PJ7	

80	VCC	VCC
81	GND	GND
82	PK7 (ADC15/PCINT23)	Analog pin 15
83	PK6 (ADC14/PCINT22)	Analog pin 14
84	PK5 (ADC13/PCINT21)	Analog pin 13
85	PK4 (ADC12/PCINT20)	Analog pin 12
86	PK3 (ADC11/PCINT19)	Analog pin 11
87	PK2 (ADC10/PCINT18)	Analog pin 10
88	PK1 (ADC9/PCINT17)	Analog pin 9
89	PK0 (ADC8/PCINT16)	Analog pin 8
90	PF7 (ADC7)	Analog pin 7
91	PF6 (ADC6)	Analog pin 6
92	PF5 (ADC5/TMS)	Analog pin 5
93	PF4 (ADC4/TMK)	Analog pin 4
94	PF3 (ADC3)	Analog pin 3
95	PF2 (ADC2)	Analog pin 2
96	PF1 (ADC1)	Analog pin 1
97	PF0 (ADC0)	Analog pin 0
98	AREF	Analog Reference
99	GND	GND
100	AVCC	VCC

Table 2.1 AT MEGA 2560 Pins

LCD Display

The term LCD stands for liquid crystal display. It is used to display letters, numbers, symbols etc. on screen. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. We

have used (X,Y) 16 x 2 LCD Display that displays 2 lines with 16 characters in each line. Here X represents column and Y represents rows.

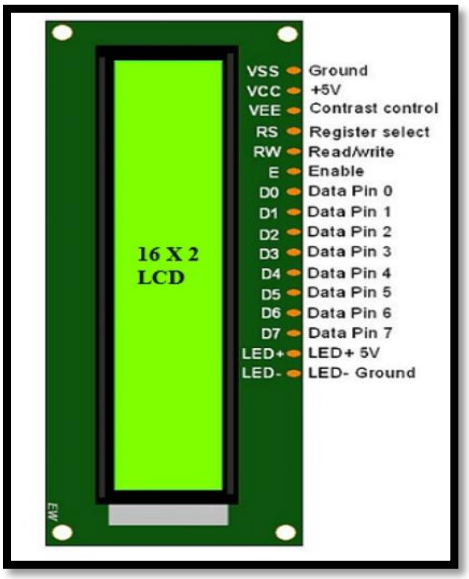


Figure 2.2 16 X 2 LCD Pin diagram

Pin Configuration

Pin Number	Function	Name
1	This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.	(Ground/Source Pin)
2	This is the voltage supply pin of the display, used to connect the supply pin of the power source.	(VCC/Source Pin)
3	This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.	(V0/VEE/Control Pin)
4	This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).	(Register Select/Control Pin)

5	This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).	(Read/Write/Control Pin)
6	This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.	(Enable/Control Pin)
7	These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.	(Data Pins)
8		
9		
10		
11		
12		
13		
14		
15	This pin is connected to +5V	(+ve pin of the LED)
16	This pin is connected to GND.	(-ve pin of the LED)

Table 2.2 LCD Pins

Robotic Chassis

The robotic chassis kit contains acrylic base, gear motors, compatible wheels, free moveable wheels etc.

Package Contains

1. 1 x Acrylic base

2. 2 x Motors

3. 1 x Nylon free moveable wheel

4. 1 x Battery box (2 x AA batteries)

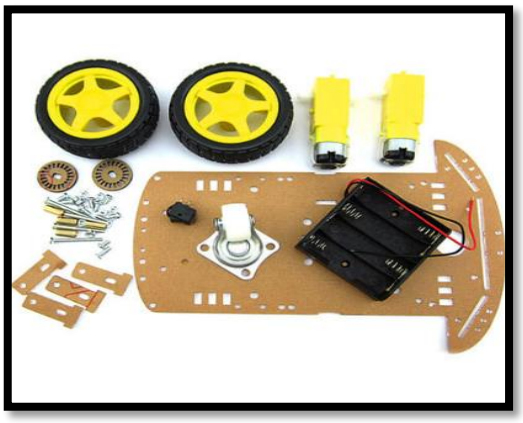


Figure 2.3 Robotic Chassis

DC Gear Motor

A DC motor or direct current motor is an electrical machine that transforms electrical energy into mechanical energy by creating a magnetic field that is powered by direct current. When a DC motor is powered, a magnetic field is created in its stator. The field attracts and repels magnets on the rotor; this causes the rotor to rotate. To keep the rotor continually rotating, the commutator that is attached to brushes connected to the power source supply current to the motors wire windings DC motors have high sustained torque. This makes them ideal for robotics applications that require a constant force.

Specifications

Voltage: 3-6VDC

Current: 80-150mA

Weight: 50grams

Gearbox/Motor Dimensions: 20mm x 22mm x 65mm

Load Speed: 3V-95 rev/min, 5V-160 rev/min, 6V-175 rev/min

No Load Speed: 3V-125 rev/min, 5V-200 rev/min, 6V-230 rev/min

Output Torque: 3V-0.8kg.cm, 5V-1.0kg.cm, 6V-1.1kg.cm

Figure 2.4 DC Gear Motor

Specifications of Wheels

- Width: 25mm
- Diameter: 65mm

Figure 2.5 Wheels

SD Card Module:

SD Card Module is a breakout board used for SD card processes such as reading and writing with a microcontroller. The board is compatible with microcontroller systems like Arduino. A standard SD card can be directly inserted into the board, but to use microSD cards, you need to use an adapter. **SD Card Module** can be used in any project that requires data reading and writing. The board has built in 3.3V voltage regulator.

Specifications

Operating Voltage: 4.5V - 5.5V DC

Current Requirement: 0.2-200 mA

3.3 V on-board Voltage Regulator

Supports FAT file system

Supports micro SD up to 2GB

Supports Micro SDHC up to 32GB

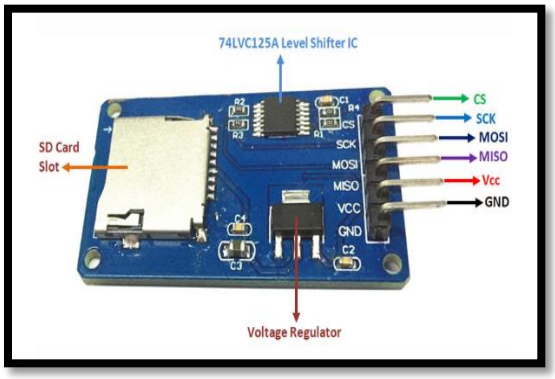


Figure 2.6 SD Card Module

Pin Configuration

Pin Type	Pin Description
GND	Ground
VCC	Voltage Input
MISO	Master In Slave Out(SPI)
MOSI	Master Out Slave In(SPI)
SCK	Serial Clock(SPI)
CS	Chip Select(SPI)

Table 2.3 SD Card Module Pins

Motor Driver

We have used L298N motor driver in our robot. L298N is a high voltage, high current dual full-bridge motor driver IC. It accepts standard TTL logic levels (Control Logic) and controls inductive loads such as relays, solenoids, DC and Stepper motors. This is a 15 pin IC. According to the L298 datasheet, its operating voltage is +5 to +46V, and the maximum current allowed to draw through each output 3A. This IC has two enable inputs, these are provided to enable or disable the device independently of the input signals.

The module has an on-board 78M05 5V Voltage regulator. This Voltage regulator will be performed only when the **5V Enable jumper** is placed. When the power supply is less than or equal to 12V, then the internal circuitry will be powered by the voltage regulator, and the 5V pin can be used as an output pin to power the microcontroller or other circuitry (sensor).
The jumper should not be placed when the power supply is greater than 12V and separate 5V should be given through 5V terminal to power the internal circuitry.

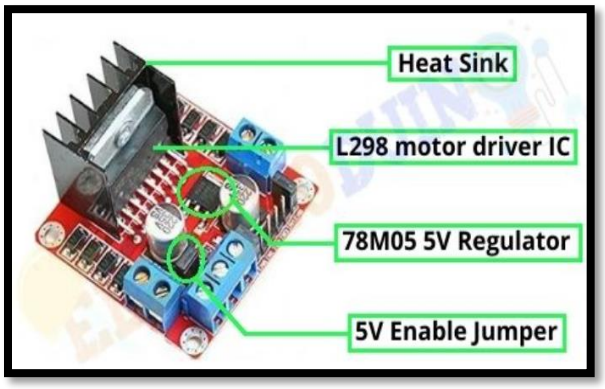


Figure 2.7 L298N Motor Driver

Pin Configuration

Pin No.	Pin Name	Description
Power Supply Pins		
1	VCC	VCC pin is used to supply power to the motor. Its input voltage is between 5 to 35V.
2	GND	GND is a ground pin. It needs to be connected to the power supply ground (negative).
3	+5V	+5V pin supplies power for the switching logic

		circuitry inside the L298N IC. If the 5V-EN jumper is in place, this pin acts as output and can be used to power up a microcontroller or other circuitry (sensor). If the 5V-EN jumper is removed, you need to connect it to the 5V power supply of the microcontroller.
Control Pins		
1	IN1	These pins are input pins of **Motor A**. These are used to control the rotating direction of Motor A. When one of them is HIGH and the other is LOW, Motor A will start rotating in a particular direction. If both the inputs are either HIGH or LOW the Motor A will stop.
2	IN2	
3	IN3	These pins are input pins of **Motor B**. These are used to control the rotating direction of Motor A. When one of them is HIGH and the other is LOW, Motor A will start rotating in a particular direction. If both the inputs are either HIGH or LOW the Motor A will stop.
4	IN4	
Speed Control Pins		
1	ENA	ENA pin is used to control the speed of **Motor A**. If a jumper is present on this pin, so the pin connected to +5 V and the motor will be enabled, then the Motor A rotates maximum speed. If we remove the jumper, we need to connect this pin to a PWM input of the microcontroller. In that way, we can control the speed of Motor A. If we connect this pin to Ground the Motor A will be disabled.
2	ENB	ENB pin is used to control the speed of **Motor B**. If a jumper is present on this pin, so the pin connected to +5 V and the motor will be enabled, then the Motor B rotates maximum speed.
Output Pins		
1	OUT1 & OUT2	This terminal block will provide the output for Motor A.
2	OUT3 & OUT4	This terminal block will provide the output for Motor B.

Table 2.4 Motor Driver Pins

Controlling two DC motors

L298N motor driver is Controlling 2 DC Motor with +5V Arduino onboard Power Supply. Following shows the circuit connection use on-board +5V power supply from Arduino board. The module has two direction control pins. The IN1 and IN2 pins control the spinning direction of motor A; While IN3 and IN4 control the spinning direction of motor B. The spinning direction of the motor can be controlled by applying logic HIGH (5V) or logic LOW (Ground) to these inputs.

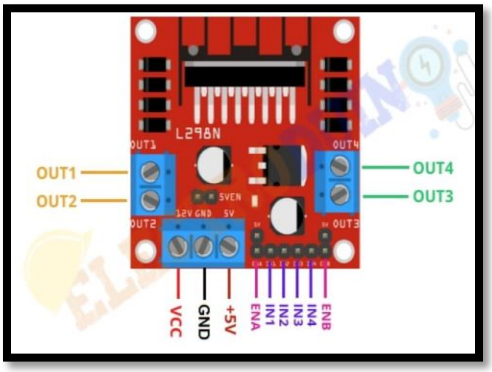


Figure 2.8 Pin Diagram of L298N Motor Driver

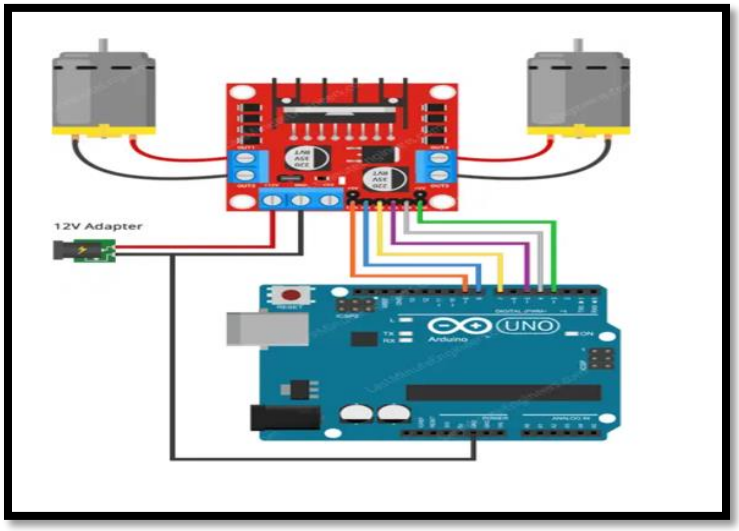


Figure 2.9 L298N with Arduino

Features & Specifications

- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N
- Motor Supply Voltage (Maximum): 46V

- Motor Supply Current (Maximum): 2A
- Logic Voltage: 5V
- Driver Voltage: 5-35V
- Driver Current:2A
- Logical Current:0-36mA
- Maximum Power (W): 25W
- Current Sense for each motor
- Heatsink for better performance
- Power-On LED indicator

Rechargeable Cells

We have used 18650 rechargeable battery. 18650 battery is a lithium-ion battery. The name derives from the battery's specific measurements: 18mm x 65mm. For scale, that's larger than an AA battery. The 18650 battery has a voltage of 3.6v and has between 2600mAh and 3500mAh (mili-amp-hours). These batteries are used in flashlights, laptops, electronics and even some electric cars because of their reliability, long run-times, and ability to be recharged hundreds of times over. 18650 batteries are what would be considered a "high drain battery." This means that the battery is designed to generate high output voltage and current to meet the power demands of the portable device in which it is being used. Hence why these powerful little batteries are utilized in more complex, power-hungry electronics that require a constant, high level of power for operation. It also has a high depth of discharge, meaning that the battery can be drained all the way down to 0% and still have the capacity to fully recharge the battery. However, this is not recommended practice, as overtime this will cause long term damage to the battery and affect its overall performance.

Figure 2.10 18650 Lithium Rechargeable Cells

Ultrasonic Sensor

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

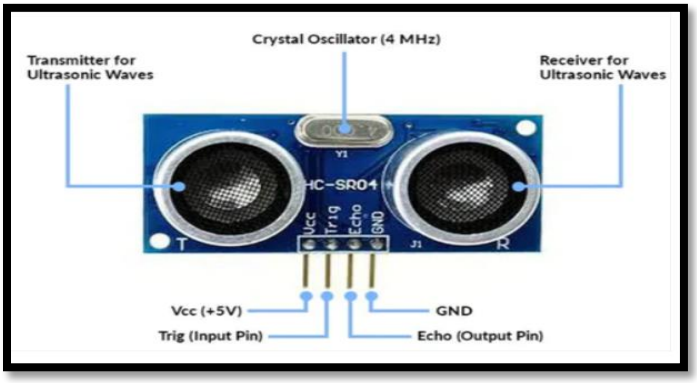


Figure 2.12 Ultrasonic Sensor

Working

An ultrasonic sensor is an electronic device that measures the distance to an object by emitting ultrasonic waves and converting the reflected sound into electrical signals. Ultrasound travels faster than audible sound (that is, sound that humans can hear). An ultrasonic sensor consists of

two main components: a transmitter (which uses a piezoelectric crystal to emit sound) and a receiver.

While some sensors use separate sound emitters and receivers, it is also feasible to merge both functions into a single device by using an ultrasonic element to switch between sending and receiving signals in a continuous cycle. The transmitter of the module transmits an ultrasonic sound. This sound will be reflected if an object is present in front of the ultrasonic sensor. The reflected sound is received by the receiver present in the same module. An ultrasonic signal is propagated by a wave at an angle of 30°. The above- depicted Figure illustrates how the ultrasonic signal propagates from the transmitter. Measuring angles should be at least 15° for maximum accuracy. In this case, external objects that fall under this measurement angle interfere with determining the distance to the desired object. The distance is determined by measuring the travel time of ultrasonic sound and its speed.

Distance = Time x Speed of sound / 2

Features

- Supply voltage +5 V;
- Consumption in silent mode 2 mA;
- Consumption at work of 15 mA;
- Measurement range - 2 to 400 cm;
- Effective measuring angle 15°;
- The dimensions are 45×20×15 mm.

Pin Configuration

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

3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	Ground	This pin is connected to the Ground of the system.

Table 2.5 Ultrasonic Sensors Pins

IR Sensors

An infrared (IR) sensor is an electronic device that measures and detects infrared radiation in its surrounding environment. Infrared radiation was accidentally discovered by an astronomer named William Herchel in 1800. While measuring the temperature of each color of light (separated by a prism), he noticed that the temperature just beyond the red light was highest. IR is invisible to the human eye, as its wavelength is longer than that of visible light (though it is still on the same electromagnetic spectrum). Anything that emits heat (everything that has a temperature above around five degrees Kelvin) gives off infrared radiation.

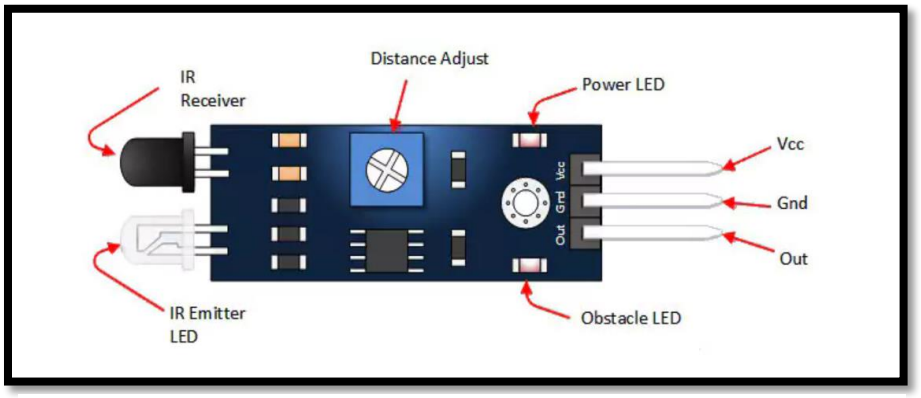


Figure 2.13 IR Sensor

Pin Configuration

Pin Name	Description
VCC	Power Supply Input
GND	Power Supply Ground

OUT	Active High Output

Table 2.6 IR Sensors Pins

Specifications

- Board size: 3.2 x 1.4cm
- Working voltage: 3.3 to 5V DC
- Operating Voltage: 3.3V: ~23mA, to 5V: ~43mA
- Detection range: 2cm to 30cm (Adjustable using potentiometer)
- Active output level: The output is 0 (Low) when an obstacle is detected

Voltage Sensor

A voltage sensor is a device that measures voltage. Voltage sensors can measure the voltage in various ways, from measuring high voltages to detecting low current levels. These devices are essential for many applications, including industrial controls and power systems.

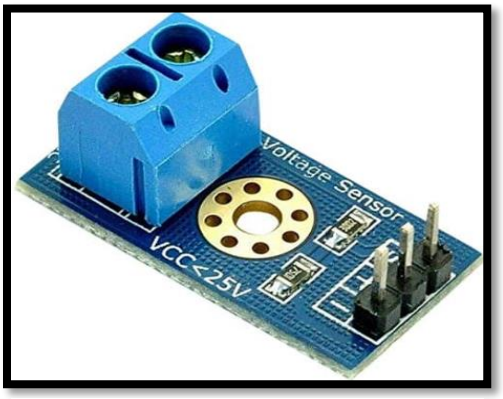


Figure 2.14 Voltage sensor

Pin Configuration

Pin Name	Description
VCC	Positive terminal of the External voltage source (0-25V)
GND	Negative terminal of the External voltage source

S	Analog pin connected to Analog pin of Arduino
+	Not Connected
-	Ground Pin connected to GND of Arduino

Table 2.7 Voltage Sensor Pins

Specifications

- Input Voltage: 0 to 25V
- Voltage Detection Range: 0.02445 to 25
- Analog Voltage Resolution: 0.00489V
- Needs no external components
- Easy to use with Microcontrollers
- Small, cheap and easily available
- Dimensions: 4 × 3 × 2 cm

Current Sensor

A device that is used to detect current to assessable output voltage is known as a current sensor. This output voltage is simply proportional to the current flow throughout the measured path. After that, this output voltage signal is used to display the current measured within an ammeter, for controlling purposes or simply stored for more analysis within a data acquisition system.

There are different types of sensors available in the market and each sensor is used for a particular range of current & ecological conditions. As compared to all these sensors, a current sensor is most frequently used by considering it as a current-to-voltage converter by placing a resistor into the flow of the current path, so the current is changed into voltage in a linear mode. The technology utilized by this sensor is significant because different kinds of sensors can contain different characteristics for various applications.

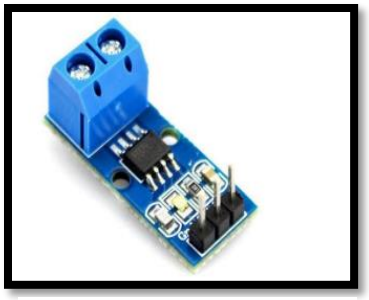


Figure 2.15 Current Sensor

Pin Number	Pin Name	Description
1	Vcc	Input voltage is +5V for typical applications
2	Output	Outputs Analog voltage proportional to current
3	Ground	Connected to ground of circuit
T1	Wire In	The wire through current has to be measured is connected here
T2	Wire Out	

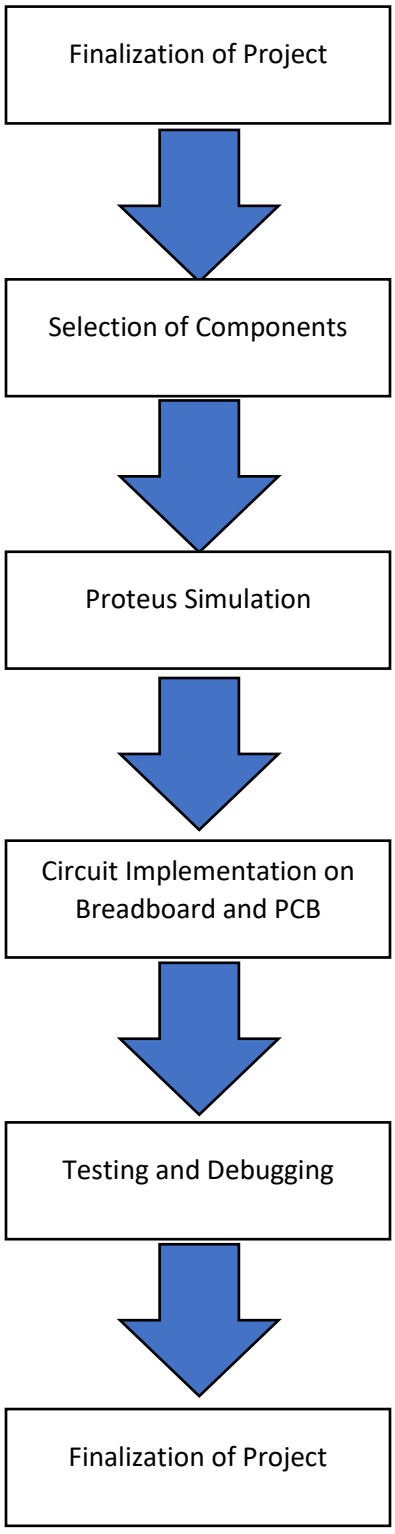
Figure 2.8 Current Sensor Pins

Specifications

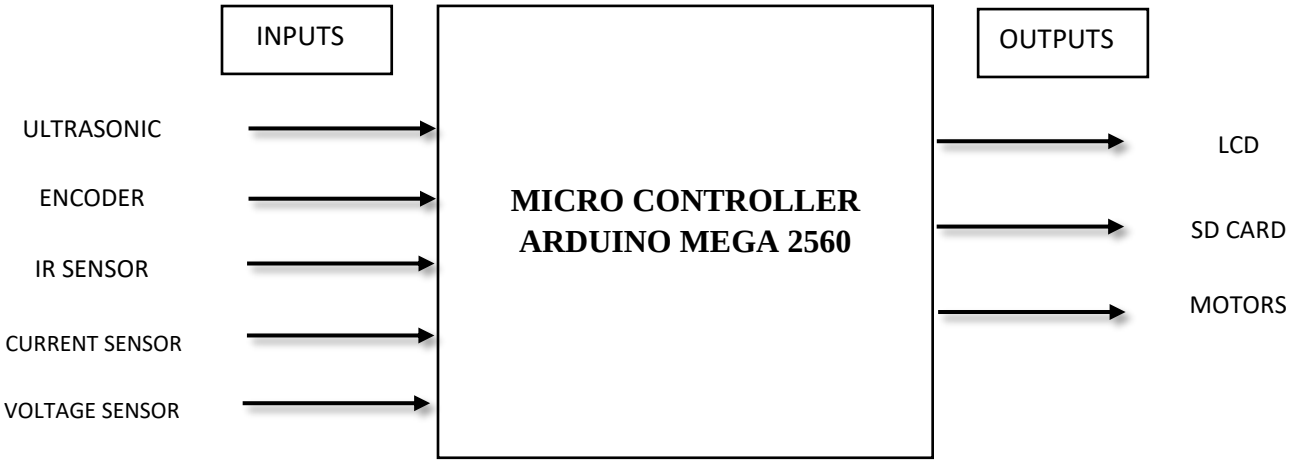
- The measuring range is the highest flow of current that a current sensor can measure up to 120A.
- The required voltage to activate the device is +5V.
- The range of frequency of this sensor can operate is 20Hz – 20kHz
- The response time of this sensor is the time taken between the input excitation application & the appearance of the equivalent o/p signal. The response time of this sensor is < 20 ns.
- The accuracy of the current sensor is above 90%.

2.4 Project Development Process

The project development for our project i.e. the line follower robot is that first we did the detail literature review in which we have selected the components by studying from their datasheets. Secondly, we have selected the software i.e. Proteus for the designing phase of our project. Then we implemented our project first on breadboard and then on PCB. The flow chart for project development process is given below.


Finalization of Project
Selection of Components
Proteus Simulation
Circuit Implementation on Breadboard and PCB
Testing and Debugging
Finalization of Project

2.5 Basic Block Diagram Of Whole System And Subsystem



2.6 Detailed Implementation Block Diagram:

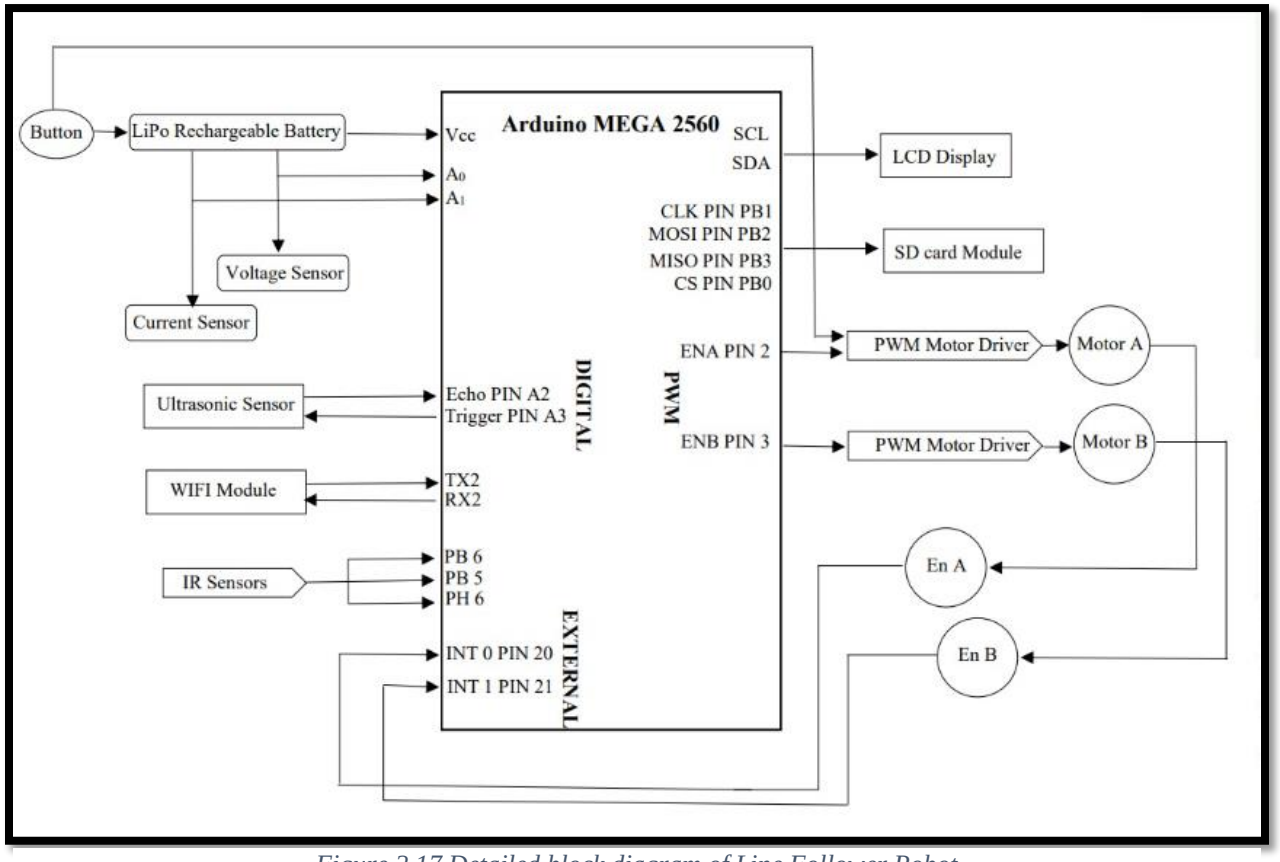


Figure 2.17 Detailed block diagram of Line Follower Robot

Chapter 3-Project Design

3.1 System Consideration for the Design

Since our project is based on the line follower robot so the sensors used in our project are based on logical framework for guiding the robot along the desired path. So we used truth table to describe all possible situations. The results derived from the truth table were subsequently simplified and utilized in programming the Microcontroller.

Now the design may contain 2 Arduino UNO or 1 ATMEGA2560. Choosing anyone can affect the design in a lot of ways in PCB specifically. Other sensor and modules like motor driver, voltage, current, speed, ultrasonic sensor or the voltage booster will change the design accordingly. Specifications of motor were considered so to check whether the motor can bear the load of our project and it is working requirements like Voltage and current needed.

2.1 Criteria for Component Selection

Criteria of component selection deals with why we have selected 5 IR sensors, 2 dc motors, power bank, Voltage booster, Current, voltage, speed, Ultrasonic sensors, and an SD card module.

Microcontroller

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

Sometimes referred to as an embedded controller or microcontroller unit (MCU), microcontrollers are found in vehicles, robots, office machines, medical devices, mobile radio transceivers, vending machines and home appliances, among other devices. They are essentially simple miniature personal computers (PCs) designed to control small features of a larger component, without a Complex front-end operating system (OS).

Working of Microcontroller

A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor. The temporary information that the microcontroller receives is stored in its data memory, where the processor accesses it and uses instructions stored in its program memory to decipher and apply the incoming data. It then uses its I/O peripherals to communicate and enact the appropriate action.

Arduino Mega 2560

The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. The Arduino MEGA 2560 is designed for projects that require more I/O lines, more sketch memory and more RAM. Its specifications are given below.

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 an AC-to-DC adapter or battery to get started.It has operating voltage: 5V, DC current per I/O pin: 40mA, DC current per I/O pin: 40mA, flash Memory: 256 KB, 8KB used by bootloader, SRAM: 8 KB, EEPROM: 4KB.

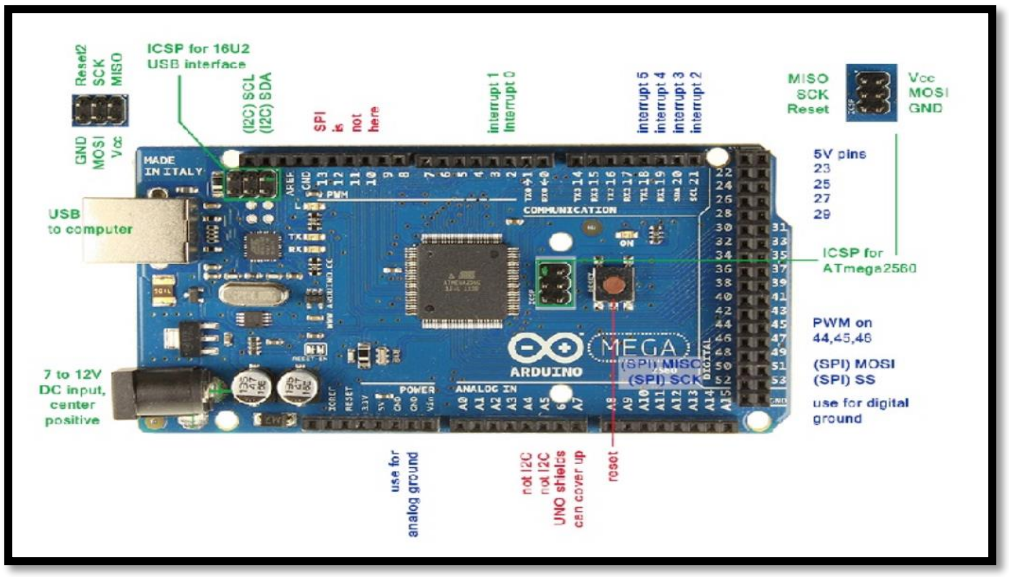


Figure 3.1 At Mega 2560

IR Sensors

The best sensor selection for our project i.e. for the line follower robot is IR sensors for line detection. In our case the sensor will detect the black line as path and white area as obstacle. In line follower robot, we can use atleast 2 IR sensors but as we have advised to use atleast 3 or more IR sensors in our project so that it will increase the complexity of our project. Also it will improve the accuracy of our robot. So we have used 3 IR sensors.

Working Principle

An IR sensor consists of two parts, the emitter circuit, and the receiver circuit. The emitter is an IR LED and the detector is an IR photodiode. The IR photodiode is sensitive to the IR light emitted by an IR LED. The photodiode's resistance and output voltage change in proportion to the IR light received. This is the underlying working principle of the IR sensor.

The type of incidence can be direct incidence or indirect incidence. In direct incidence, the IR LED is placed in front of a photodiode with no obstacle.

In indirect incidence, both the diodes are placed side by side with an opaque object in front of the sensor. The light from the IR LED hits the opaque surface and reflects back to the photodiode.

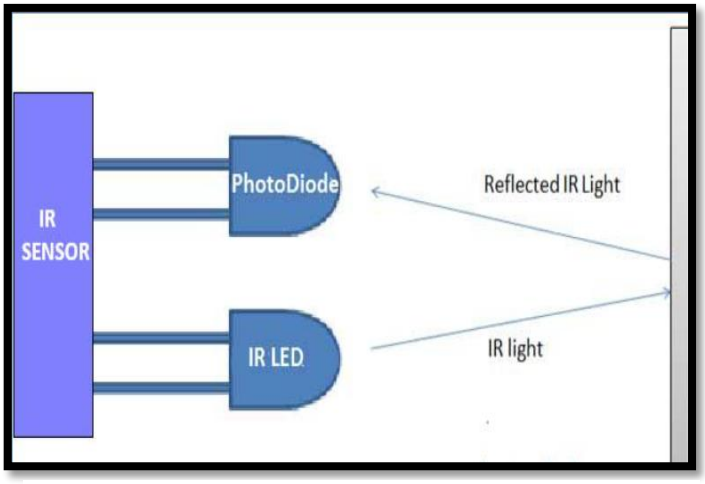


Figure 3.2 Working of IR sensor

Ultrasonic Sensor:

We have selected an ultrasonic sensor in our project i.e. line follower robot to detect any obstacle. The robot will stop movement when it detects any obstacle in front of it. Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target). This sensor in well-known sensor used to measure the distance and sensing the objects. It supplies the voltage of +5V.

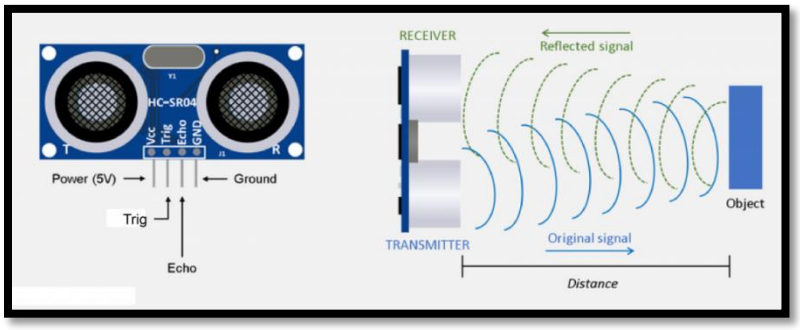


Figure 3.3 Working of Ultrasonic Sensor

Voltage Sensor:

We have selected the voltage sensor for our robot to check the voltage of the circuit at any point. Voltage sensors can determine the AC voltage or DC voltage level. The input of this sensor is the

voltage, whereas the output is the switches, analog voltage signal, a current signal, or an audible signal etc. This type of sensor is often used with microcontrollers since they can easily measure changes in electromagnetic fields around them with the help of built-in analog-to-digital converters (ADCs).

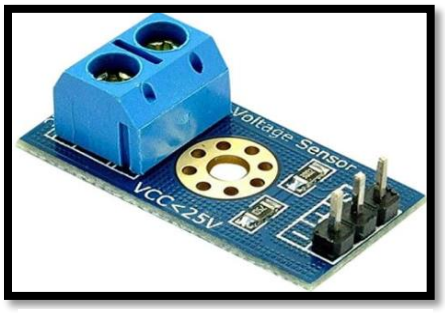


Figure 3.4 Voltage Sensor

Current sensor

We have selected the current sensor for our robot to if proper amount of current is provided to the circuit for robot working or not. It measures both AC and DC current. Its provide insolation from the load. The required voltage to activate the device is +5V. Easy to integrate with microcontroller, since it outputs analog voltage.

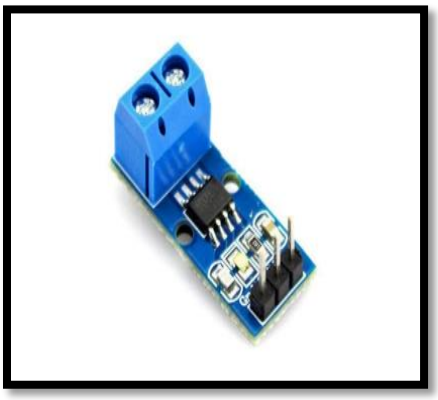


Figure 3.5 Current

LCD Display

We have used LCD (Liquid Crystal Display) in our robot. This LCD will show the current and voltage output on the screen. LCD is used to display letters, numbers, symbols etc. on screen. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are

mainly preferred for multi-segment light-emitting diodes and seven segments. We have used (X,Y) 16 x 2 LCD Display that displays 2 lines with 16 characters in each line. Here X represents column and Y represents rows.

Figure 3.6 LCD Display

Motor Driver

We have used L298N motor driver module in our robot. L298N is a high voltage, high current dual full-bridge motor driver IC. It accepts standard TTL logic levels (Control Logic) and controls inductive loads such as relays, solenoids, DC and Stepper motors. This is a 15 pin IC. According to the L298 datasheet, its operating voltage is +5 to +46V, and the maximum current allowed to draw through each output 3A. This IC has two enable inputs, these are provided to enable or disable the device independently of the input signals.

The module has an on-board 78M05 5V Voltage regulator. This Voltage regulator will be performed only when the **5V Enable jumper** is placed. When the power supply is less than or equal to 12V, then the internal circuitry will be powered by the voltage regulator, and the 5V pin can be used as an output pin to power the microcontroller or other circuitry (sensor).
The jumper should not be placed when the power supply is greater than 12V and separate 5V should be given through 5V terminal to power the internal circuitry.

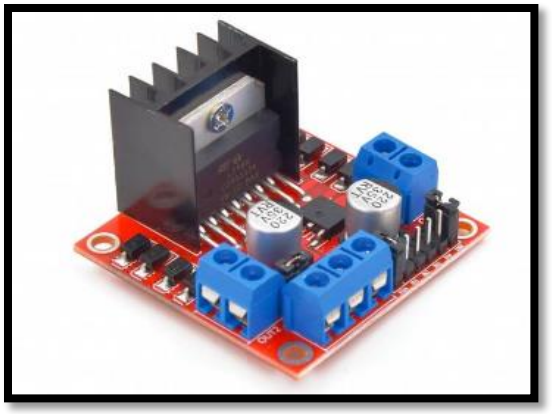


Figure 3.7 L298N Motor Driver

DC MOTORS:

We have used two DC motors in our robot. Dc motors are most easy to control as compared to AC, and they can do the required work. One dc motor requires only 2 signals for its operation. If we want to change its direction just reverse the polarity of the power supply across it.

Mathematical interpretation:

As rotational power is constant for DC motors for constant input electrical power. Thus, torque is inversely proportional speed.

Why 2 Motors

We are using 2 motors in our robot so that our robot can move easily in any direction. This steering mechanism of the robot is called differential drive.

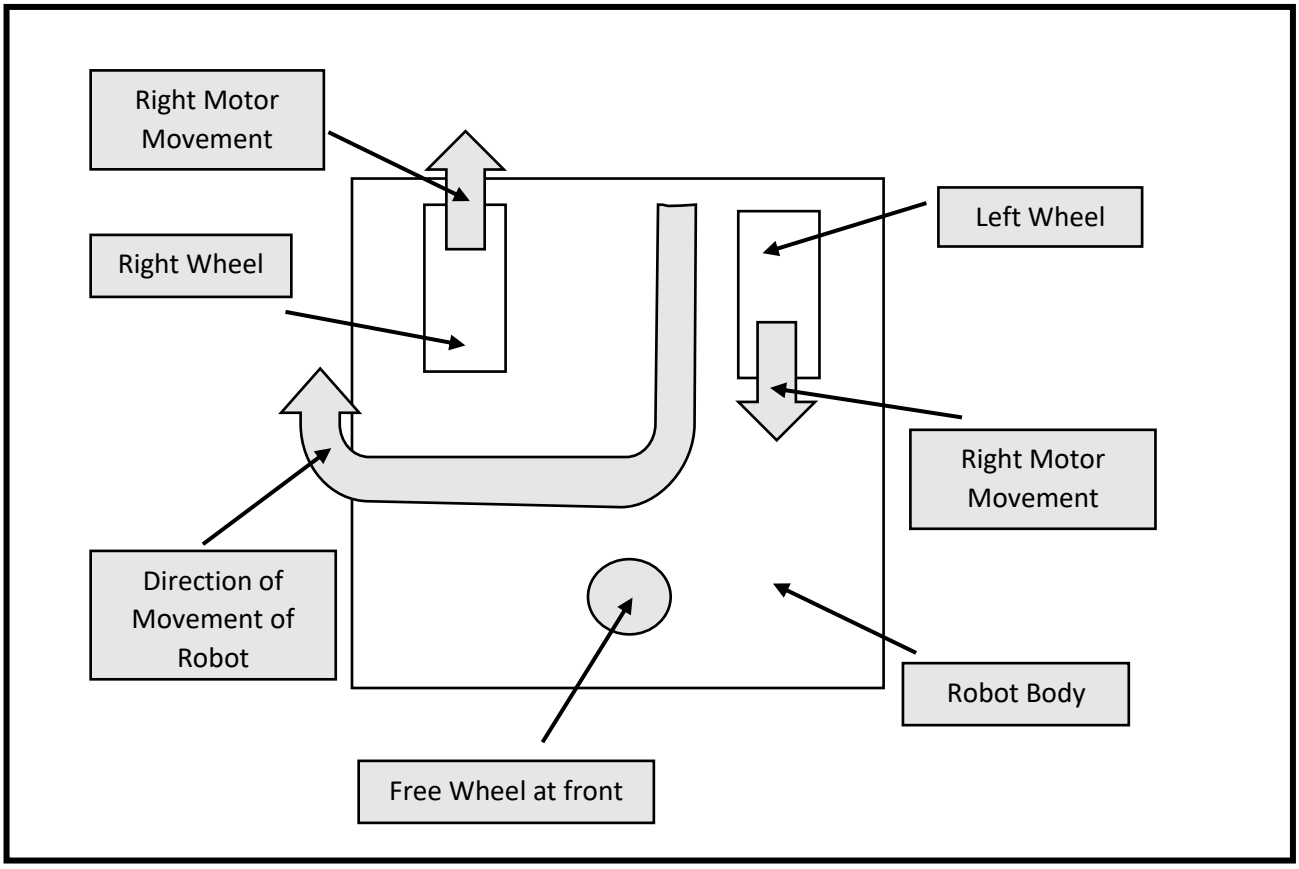


Figure 3.7 Description of Parts of Motors

Left Motor	Right Motor	Robot Movement
Straight	Straight	Straight
Stop	Straight	Left
Reverse	Straight	Left
Straight	Stop	Right
Straight	Reverse	Right
Reverse	Reverse	Reverse

Table 3.1 Combinations of Motor Movements

Why above Components

The selection of above components are strategic for a line follower robot. The combination of powerful microcontroller and all the sensors to detect black line and prevent obstacle, motor control and power monitoring makes the robot efficient, accurate and flexible based on different activities.

The combination of these components ensure that the line follower robot in not only reliable but environment friendly, economical and cost effective.

Chapter 4- Mechanical Design

The mechanical design for the line follower robot is an important factor that directly affects its performance and flexibility. The chassis, wheels and overall structural layout must be carefully considered to ensure stability and accuracy along the way. The chassis must be robust enough, be suitable for necessary electronic components and be easy to maintain. The type of wheel chosen affects the robot's ability to navigate turns and curves affecting its grip on the surface. Additionally, the placement of sensors (e.g. IR sensors) plays an important role in the robot's line ability. So, designing the robot in a most sufficient way so that external condition or internal condition that material used and that design are minimum.

4.1 Mechanism selection

The mechanism used for our robot is the base (shown in figure 4.1) which is the main body, and the PCB is mounted to the base with the connection of some height. The lower part of base is connected by a rectangular mount of 2 motors. Each at the back (left and right). These motors are connected to the wheel for the movement of robot (front, left or right). The front side of the robot is directly connected to the freely moveable wheel. There are circuits on the PCB consisting of microcontroller, voltage sensor, current sensor, SD Card module. Bolts and nuts are used in many places to ensure sealing. 3 IR sensors are used in front of the base according to the truth table condition.

Figure 4.1 Base of Robot

4.2 Platform Design

There are many designs which can be used for the line follower robot. It depends upon that what and how components are using in the line follower robot. Some components are easily available in the market at very low cost and some others we design by ourselves. The designs are of many shapes either rectangular, triangular or square. It's all depends upon our feasibility. This is because our design wouldn't affect the project as our project contains simple configuration above the base. We are using Power bank, Motor driver and PCB that includes all the components. The motors and tires are below so they got their own space for the working of robot.

4.3 Material Selection and choices

Material selection is very important thing to do while doing an engineering project, especially when the project is of high level are there are lot of loads over it. Commonly the material selected for different projects are steel, brass, aluminum, rubber, plastic etc. Different materials have different properties according to which they are selected.

For our project, we have used the base of acrylic. This is because it is very light weight, durable and relatively easy to work with it and is suitable for our project as well.

The bults and nuts are of steel so that they can hold things together tightly for times. The connectors are of plastic (Glass). These materials are chosen for our project because it is easily available in the market at very low cost and feasible for our project.

4.4 Actuation with speciation and datasheet

In the line follower robot actuation refers responsibility of initiating and controlling movement which is usually accomplished by the motor that drives the wheel. The choice of motor and

specifications are important for a robot to work. For instance, consider the widely used DC motor which features high torque and precise control. A suitable motor driver such as L298N can help control the motor. Motors specifications such as voltage, current and speed are important. Additionally, the use of encoders can increase the accuracy of wheel movement and provide feedback to the robotic control system. Referring to the datasheet of motor and driver instructions, such as operating voltage, current rating and motor speed specifications, to demonstrate the integration of complete line follower robot design.

Chapter 5- Electronic Design and Sensor Selections

5.1 Component Selection

The components we have used in our project i.e. Line Follower Robot are as follows:

- Arduino At Mega 2560
- L298N Motor Driver
- IR Sensor
- Ultrasonic Sensor
- Voltage Sensor
- Current Sensor
- LCD Display
- SD Card Module
- ESP8266 Wifi Module

5.2 Sensors along with specification and features from datasheet

L298N Motor Driver

We can control the speed of the DC motor by simply controlling the input voltage to the motor and the most common method of doing that is by using PWM signal.

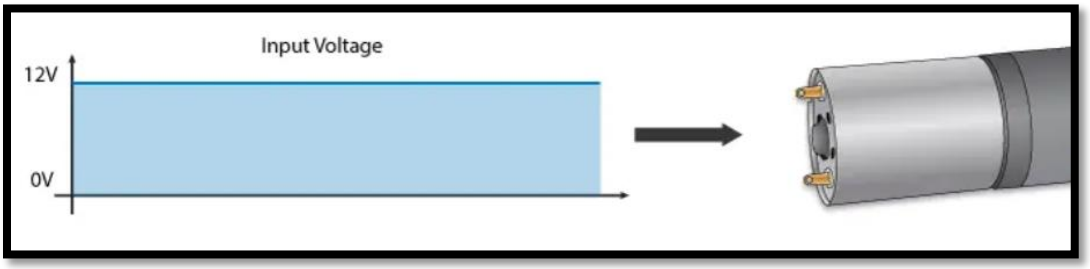


Figure 5.1 Input Voltage for Motor Driver

PWM DC Motor Control

PWM, or pulse width modulation is a technique which allows us to adjust the average value of the voltage that's going to the electronic device by turning on and off the power at a fast rate. The average voltage depends on the duty cycle, or the amount of time the signal is ON versus the amount of time the signal is OFF in a single period of time.

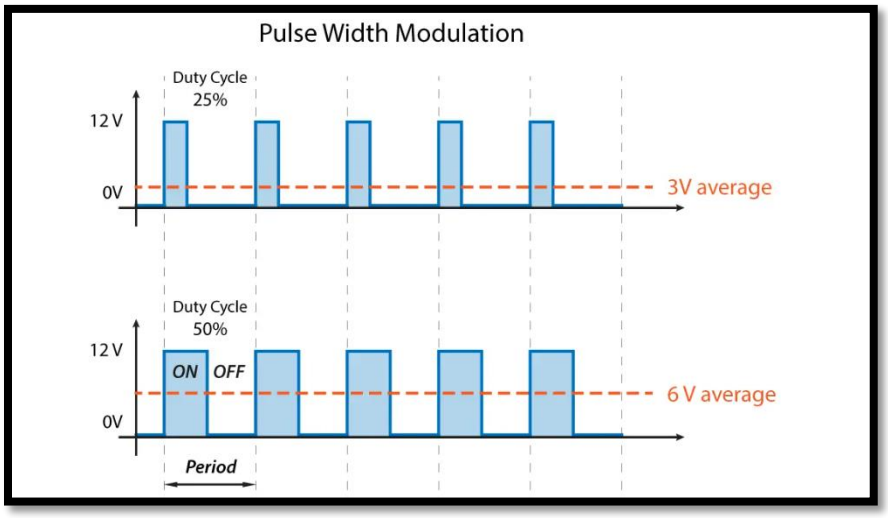


Figure 5.2 PWM DC Motor Control

H-Bridge DC Motor Control

On the other hand, for controlling the rotation direction, we just need to inverse the direction of the current flow through the motor, and the most common method of doing that is by using an H-Bridge. An H-Bridge circuit contains four switching elements, transistors or MOSFETs, with the motor at the center forming an H-like configuration. By activating two particular switches at the same time we can change the direction of the current flow, thus change the rotation direction of the motor.

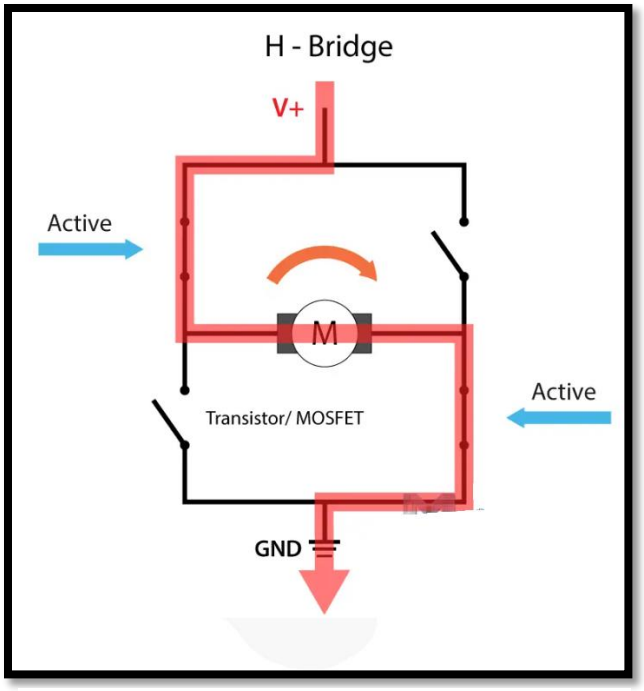


Figure5.3 Circuit Diagram of H-Bridge DC Motor Control

So if we combine these two methods, the PWM and the H-Bridge, we can have a complete control over the DC motor. There are many DC motor drivers that have these features and the L298N is one of them.

L298N Driver

The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.

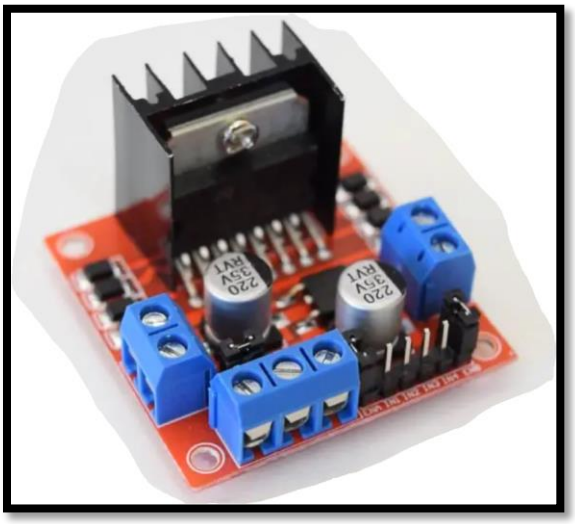


Figure5.4 L298N Motor Driver

Let's take a closer look at the pinout of L298N module and explain how it works. The module has two screw terminal blocks for the motor A and B, and another screw terminal block for the Ground pin, the VCC for motor and a 5V pin which can either be an input or output.

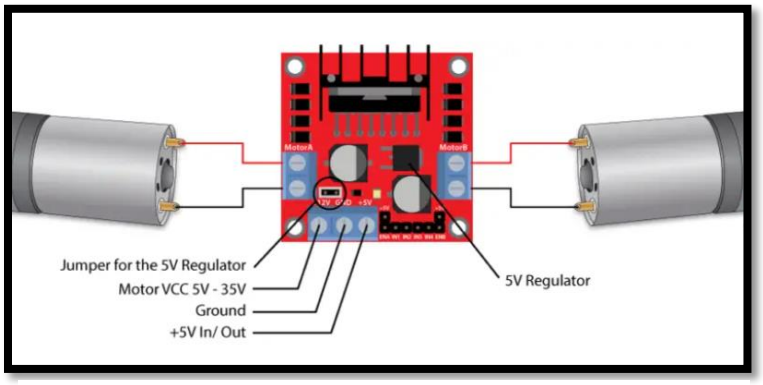


Figure 5.5 L298N Controlling two Motors

This depends on the voltage used at the motors VCC. The module have an onboard 5V regulator which is either enabled or disabled using a jumper. If the motor supply voltage is up to 12V we can enable the 5V regulator and the 5V pin can be used as output, for example for powering our Arduino board. But if the motor voltage is greater than 12V we must disconnect the jumper because those voltages will cause damage to the onboard 5V regulator. In this case the 5V pin will be used as input as we need connect it to a 5V power supply in order the IC to work properly.

We can note here that this IC makes a voltage drop of about 2V. So for example, if we use a 12V power supply, the voltage at motors terminals will be about 10V, which means that we won't be able to get the maximum speed out of our 12V DC motor.

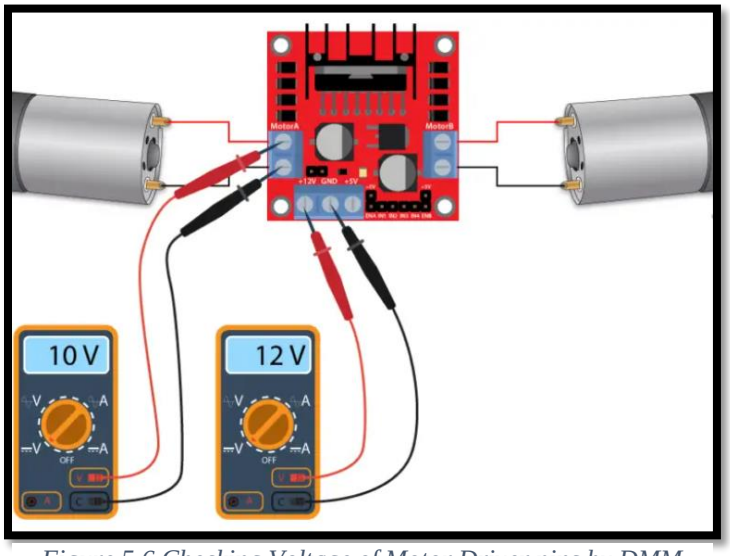


Figure 5.6 Checking Voltage of Motor Driver pins by DMM

HC-SR04 Ultrasonic Sensor Working & Interfacing With Arduino

HC-SR04 Ultrasonic Distance Sensor can report the range of objects up to 13 feet away. This is a good thing to know when we're trying to save our robot from hitting a wall.

They are low power (suitable for battery operated devices), affordable, easy to interface and extremely popular with hobbyists.

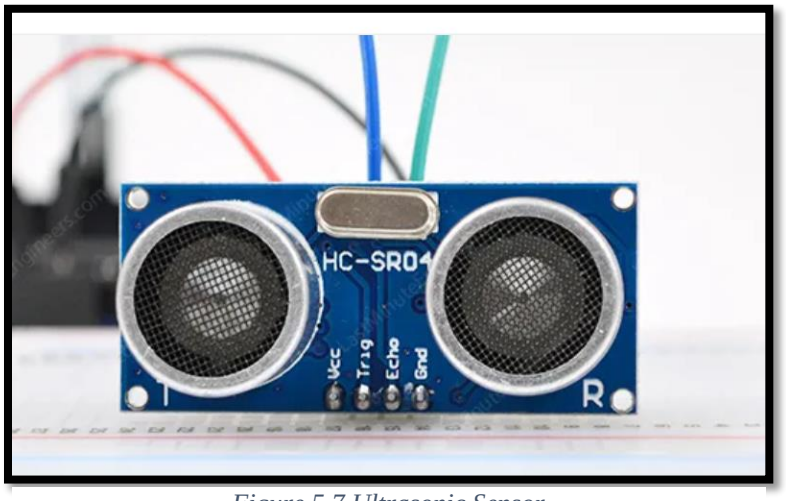


Figure 5.7 Ultrasonic Sensor

What is Ultrasound?

Ultrasound is a high-pitched sound wave whose frequency exceeds the audible range of human hearing.

Humans can hear sound waves that vibrate in the range of about 20 times a second (a deep rumbling noise) to 20,000 times a second (a high-pitched whistle). However, ultrasound has a frequency of more than 20,000 Hz and is therefore inaudible to humans.

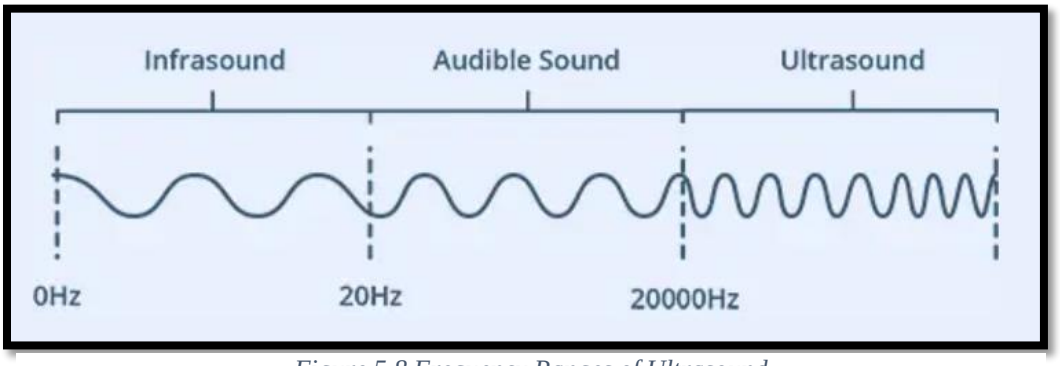


Figure 5.8 Frequency Ranges of Ultrasound

HC-SR04 Hardware Overview

An HC-SR04 ultrasonic distance sensor actually consists of two ultrasonic transducers.

One acts as a transmitter that converts the electrical signal into 40 KHz ultrasonic sound pulses. The other acts as a receiver and listens for the transmitted pulses.

When the receiver receives these pulses, it produces an output pulse whose width is proportional to the distance of the object in front.

This sensor provides excellent non-contact range detection between 2 cm to 400 cm (~13 feet) with an accuracy of 3 mm.

Since it operates on 5 volts, it can be connected directly to an Arduino or any other 5V logic microcontroller.

Technical Specifications

Operating Voltage	DC 5V
Operating Current	15mA
Operating Frequency	40KHz
Max Range	4m
Min Range	2cm
Ranging Accuracy	3mm
Measuring Angle	15 degree
Trigger Input Signal	10µS TTL pulse
Dimension	45 x 20 x 15mm

Figure 5.9 Specs of Ultrasonic Sensor

HC-SR04 Ultrasonic Sensor Pinout

1. **VCC** supplies power to the HC-SR04 ultrasonic sensor. You can connect it to the 5V output from your Arduino.
2. **Trig(Trigger)** pin is used to trigger ultrasonic sound pulses. By setting this pin to HIGH for 10µs, the sensor initiates an ultrasonic burst.
3. **Echo** pin goes high when the ultrasonic burst is transmitted and remains high until the sensor receives an echo, after which it goes low. By measuring the time the Echo pin stays high, the distance can be calculated.
4. **GND** is the ground pin. Connect it to the ground of the Arduino.

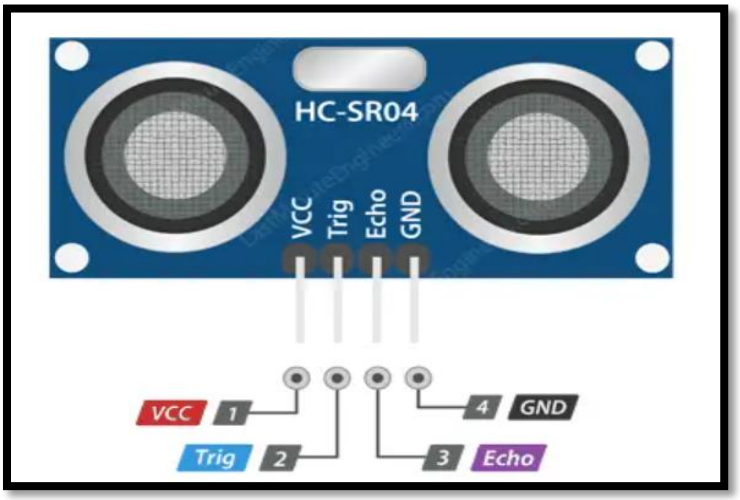


Figure 5.10 Pinout of Ultrasonic Sensor

HC-SR04 Ultrasonic Distance Sensor Working

It all starts when the trigger pin is set HIGH for 10µs. In response, the sensor transmits an ultrasonic burst of eight pulses at 40 kHz. This 8-pulse pattern is specially designed so that the receiver can distinguish the transmitted pulses from ambient ultrasonic noise.

These eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the echo pin goes HIGH to initiate the echo-back signal.

If those pulses are not reflected back, the echo signal times out and goes low after 38ms (38 milliseconds). Thus a pulse of 38ms indicates no obstruction within the range of the sensor.

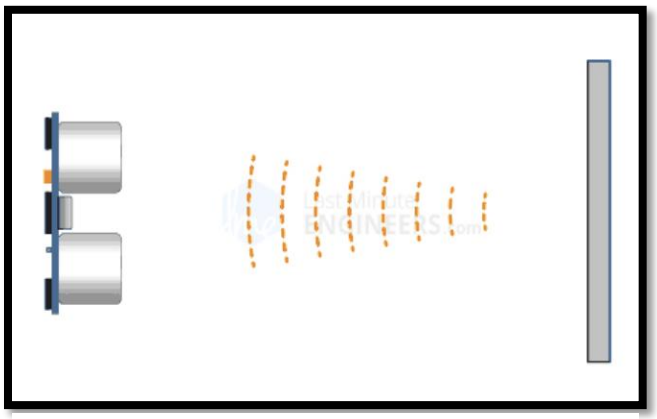


Figure 5.11 Working of Ultrasonic Sensor

If those pulses are reflected back, the echo pin goes low as soon as the signal is received. This generates a pulse on the echo pin whose width varies from 150 µs to 25 ms depending on the time taken to receive the signal.

Figure 5.12 Pulses of Ultrasonic Sensors

Calculating the Distance

The width of the received pulse is used to calculate the distance from the reflected object. This can be worked out using the simple distance-speed-time equation we learned in high school.

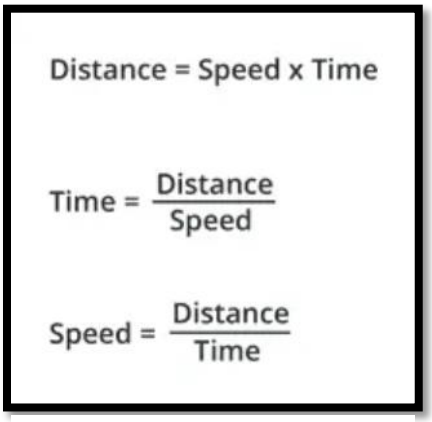


Figure 5.13 Calculation of Distance of Ultrasonic Sensor

Suppose we have an object in front of the sensor at an unknown distance and we receive a pulse of 500µs width on the echo pin. Now let's calculate how far the object is from the sensor. For this we will use the below equation.

Distance = Speed x Time

Here we have the value of time i.e. 500 µs and we know the speed. Of course it's the speed of sound i.e. **340 m/s.** To calculate the distance we need to convert the speed of sound into cm/µs. It is **0.034 cm/µs.** With that information we can now calculate the distance given by:

Distance = 0.034 cm/µs x 500 µs

Note: Remember that the echo pulse indicates the time it takes for the signal to be sent and reflected back. So to get the distance, you have to divide your result by two.

Distance = (0.034 cm/µs x 500 µs) / 2

Distance = 8.5 cm

Now we know that the object is 8.5 cm away from the sensor.

Example:

Wiring an HC-SR04 Sensor to an Arduino UNO

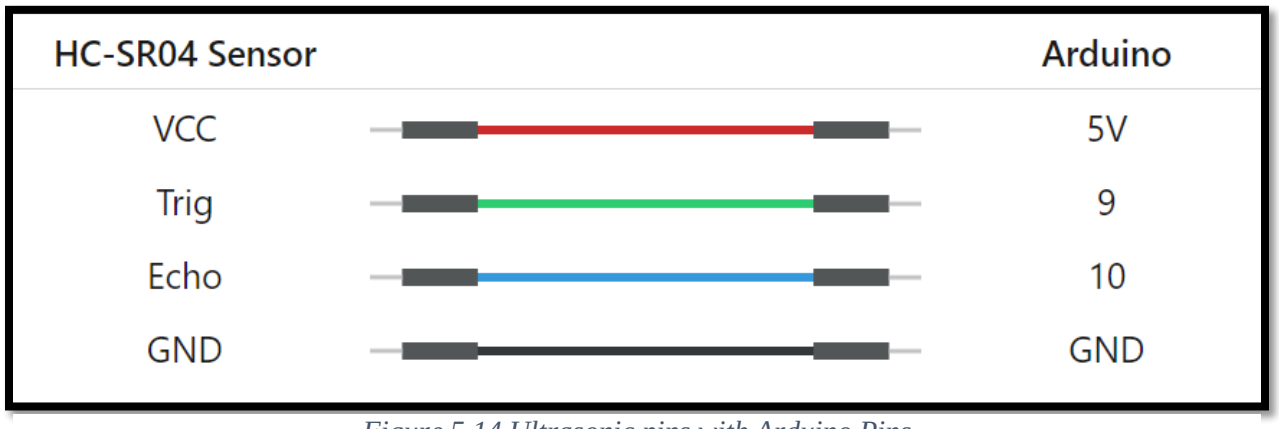


Figure 5.14 Ultrasonic pins with Arduino Pins

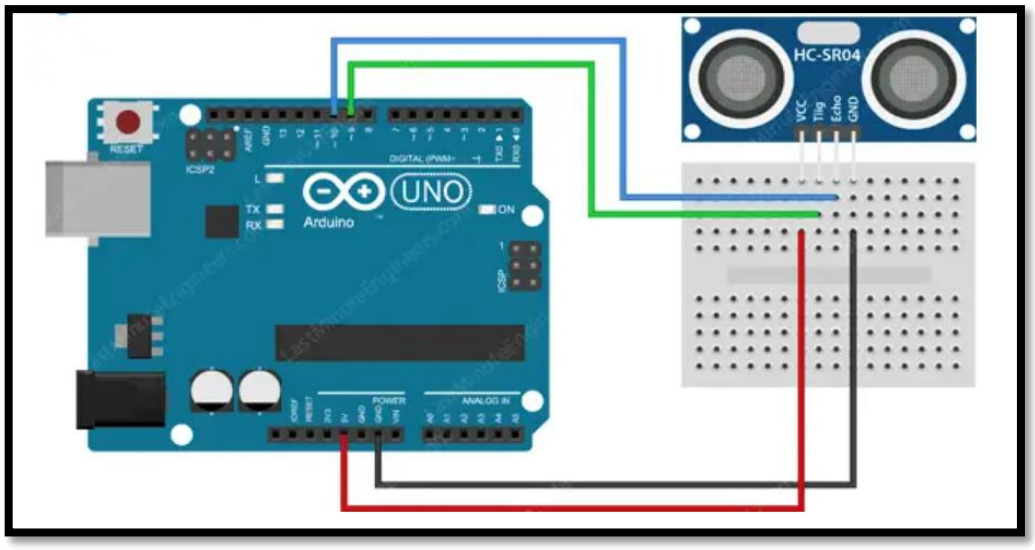


Figure 5.15 Interfacing Arduino with Ultrasonic

Arduino Example Code

Here is a simple sketch that uses the serial monitor to display a distance measured in centimeters. Give this sketch a try before we start a detailed analysis of it.

```
// Include NewPing Library
#include "NewPing.h"

// Hook up HC-SR04 with Trig to Arduino Pin 9, Echo to Arduino pin 10
#define TRIGGER_PIN 9
#define ECHO_PIN 10

// Maximum distance we want to ping for (in centimeters).
#define MAX_DISTANCE 400

// NewPing setup of pins and maximum distance.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

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

```
}
void loop() {
        Serial.print("Distance = ");
        Serial.print(sonar.ping_cm());
        Serial.println(" cm");
        delay(500);
}
```

ESP-01 ESP8266 Wi-Fi module

The ESP-01 ESP8266 is a Wi-Fi module that allows microcontrollers to access a Wi-Fi network. ESP-01s WIFI Module is a self-contained System On a Chip (SoC) that has a 32-bit microcontroller, 1MB of flash memory, and 2 GPIOs. ESP-01 can be programmed using the Arduino IDE or the ESP8266 AT firmware.

The ESP-01 is a Wi-Fi module that is based on the ESP8266 SoC. ESP8266 ESP-01 WIFI Module is a small, low-cost module that can use to add Wi-Fi connectivity to a variety of projects. The ESP-01 has two GPIOs, which can use to control external devices. It also has a built-in ADC, which can use to read analog signals.

The ESP8266 ESP-01 WIFI Module operates at a voltage of 3.3V. The ESP-01 has a built-in voltage regulator that converts the input voltage to 3.3V. The input voltage can be from 2.5V to 3.6V. If you power the ESP-01 with a voltage that is higher than 3.6V, the voltage regulator may be damaged.

It is important to note that the ESP-01's I/O pins are also 3.3V. If you connect the ESP-01's I/O pins to a 5V signal, the ESP-01 may damage. You can use a level shifter to convert the 5V signal to 3.3V before connecting it to the ESP-01's I/O pins.

The ESP8266 WIFI Module ESP-01 works on the principle of the 802.11b/g/n Wi-Fi protocol. The 802.11b/g/n Wi-Fi protocol is a set of standards that define how devices communicate over a wireless network. The ESP8266-01 / ESP-01 uses the 802.11b/g/n Wi-Fi protocol to connect to a Wi-Fi network and transmit data.

The ESP-01 also has a built-in ADC that can use to read analog signals. The ADC can use to read the temperature, humidity, and other analog signals.

Features

The ESP8266 WIFI Module ESP-01 has a 32-bit microcontroller that is responsible for controlling the Wi-Fi module. The microcontroller also has a built-in flash memory that is used to store the code and data for the ESP-01. The ESP8266 WIFI Module ESP-01 has two GPIOs that can be used to control external devices. The GPIOs can be used to control LEDs, motors, and other devices. The ESP-01 has many benefits, including:

- Low cost
- Easy to use
- Wide range of features
- Small size
- Low power consumption
- Open-source

ESP-01 ESP8266 Wi-Fi Module Specifications

The ESP8266 ESP-01 WIFI Module is easy to use. ESP8266 WIFI Module ESP-01 can be programmed using the Arduino IDE or the ESP8266 AT firmware. ESP8266 WIFI Module has a wide range of features, making it a versatile module that can be used for a variety of projects.

The ESP8266 ESP-01 Module is a small module, making it easy to integrate into your projects. The ESP-01 WIFI Module consumes less power than some other Wi-Fi modules, making it a good choice for battery-powered projects. The ESP-01 is an open-source module, which means that the firmware and documentation are freely available. This makes it easy to learn about the ESP-01. Here are some of the specifications of the ESP-01:

- **Microcontroller:** Tensilica Xtensa LX106 32-bit
- **Flash memory:** 1MB
- **Operating voltage:** 3.3V
- **GPIOs:** 2
- **ADC:** 10-bit
- **Wi-Fi:** 802.11b/g/n
- **Dimensions:** 14.3 x 24.8mm
- **Weight:** 1.5g

ESP-01 ESP8266 Wi-Fi Module Applications

The ESP-01 is a small, low-cost module that can use to add Wi-Fi connectivity to a variety of projects. ESP8266 ESP-01 WIFI Module has two GPIOs, which can use to control external devices. ESP8266 ESP-01 WIFI Module also has a built-in ADC, which can use to read analog signals.

The ESP-01 is a versatile and affordable module that can use to create a variety of IoT projects. If we are looking for a way to add Wi-Fi connectivity in the project, the ESP-01 is a great option to consider. Here are some of the projects that we can do with the ESP-01:

- Smart home devices
- Sensor networks
- Web servers
- Remote control devices
- Robotic applications

ESP-01 ESP8266 Wi-Fi Module Pin Description

The ESP-01 also has a built-in LED that connected to the GPIO2 pin. The LED will light up when the ESP-01 is power on. Here is the pin description of the ESP-01:

- **VCC:** This pin provides the main power to the ESP-01. The voltage should be 3.3V.
- **GND:** This pin is the ground pin.
- **TX:** This pin is the transmit pin. It is use to send data from the ESP-01 to another device.
- **RX:** This pin is the receive pin. It is use to receive data from another device to the ESP-01.
- **GPIO0:** This pin can use as a GPIO or as a reset pin. When the GPIO0 pin is pull low, the ESP-01 will reset.
- **GPIO2:** This pin can use as a GPIO.
- **CH_PD:** This pin is use to power on the ESP-01. When the CH_PD pin is pulled high, the ESP-01 will power on.

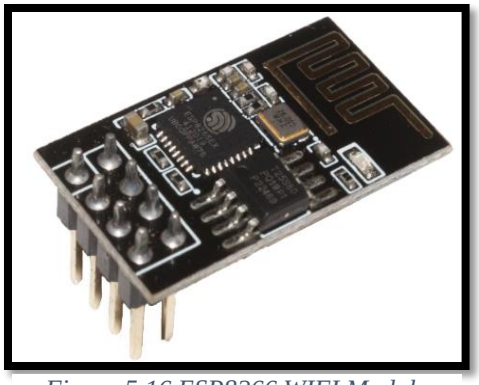


Figure 5.16 ESP8266 WIFI Module

Interfacing Voltage Sensor with Arduino

The Arduino Uno, like many microcontrollers, has a built-in Analog to Digital Converter (ADC) that can convert an analog voltage on a pin to a digital number. However, the maximum analog input pin voltage is limited to 5V.

There's an easier way to measure voltages lower than 25V using this Voltage Sensor. It is a pre-made voltage divider circuit that uses precision resistors to provide accurate readings.

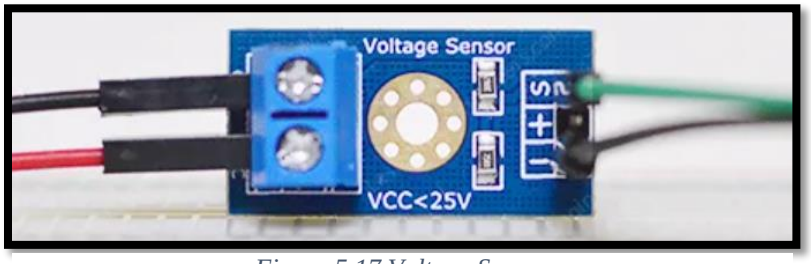


Figure 5.17 Voltage Sensor

Hardware Overview

The Voltage Sensor, in essence, is a simple voltage divider circuit composed of two resistors.

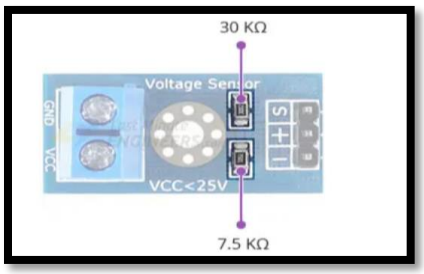


Figure 5.18 Voltage Sensor Containing two resistors

The schematic of the Voltage Sensor is illustrated in the following image.

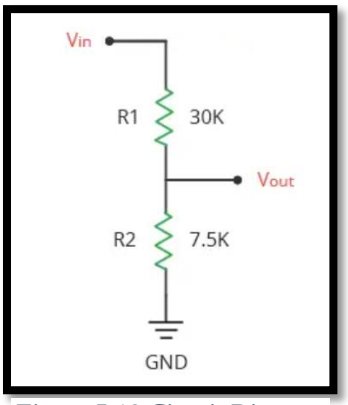


Figure 5.19 Circuit Diagram of Voltage Sensor

There are two resistors in this circuit. The resistor (R1) closest to the input voltage, has a value of 30 KΩ, and the resistor (R2) closest to ground, has a value of 7.5 KΩ. The voltage drop across R2 is our divided voltage. This signal is broken out to a header pin labeled S.

This simple circuit divides the input voltage by a factor of 5. That's why this voltage sensor can help you measure voltages that are less than 25 volts with an Arduino.

Reading the Voltage Sensor

For reading the voltage sensor or any voltage divider, we can use the voltage divider equation. The voltage divider equation assumes the three values of the above circuit, the input voltage (Vin), and both resistor values (R1 and R2). Given those values, we can use this equation to find the output voltage (Vout)

$$V_{out} = V_{in} \frac{R2}{R1 + R2}$$

However, in our case, we will be measuring the output voltage (Vout) from the voltage divider circuit using Arduino's ADC. Therefore, the value we do not know is Vin.

Let's rearrange the above equation to solve for the input voltage (Vin).

$$V_{in} = V_{out} \frac{R1 + R2}{R2}$$

Voltage Sensor Pinout

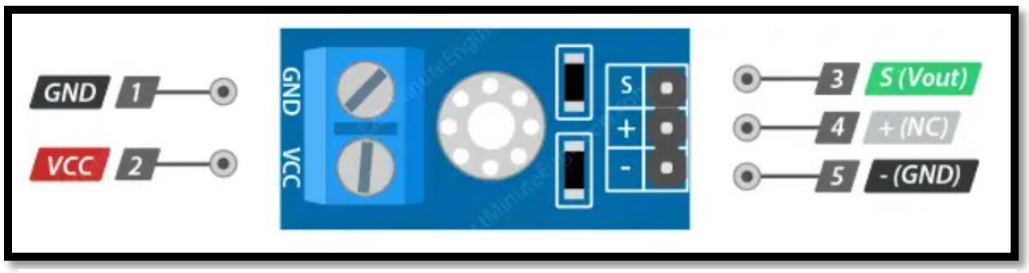


Figure 5.20 Pinout of Voltage Sensor

Input Terminal

- **VCC** is connected to the positive terminal of the voltage source you want to measure. The recommended voltage range for this pin is 0 to 25V.
- **GND** is connected to the negative terminal of the input voltage source.

Output Header

- **S** is the signal output pin of the voltage sensor module. It provides an analog voltage that is proportional to the input voltage level. It's usually connected to one of the analog input pins on the Arduino.
- '+' is not connected to anything.
- '-' is the common ground pin.

Hardware

To connect the voltage sensor to Arduino, connect the source that you want to measure to the input screw terminal. Then, connect the 'S' pin on the voltage sensor to the 'A0' analog pin on the Arduino and the '-' pin to ground.

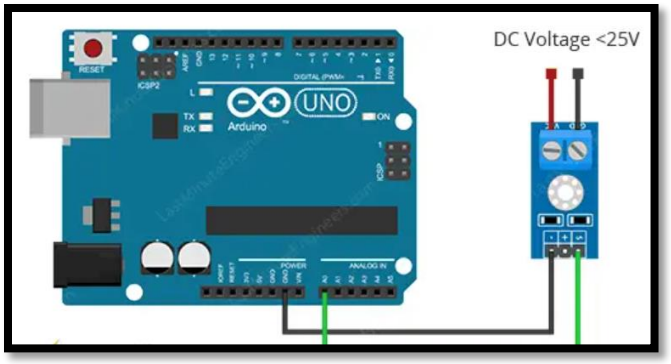


Figure 5.21 Interfacing of Voltage Sensor with Arduino

Arduino Example Code

Here is a simple sketch that reads the analog voltage on the analog pin A0, calculates the input voltage using voltage divider equation and prints the results to the Serial Monitor.

```
// Define analog input
#define ANALOG_IN_PIN A0

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

// Floats for resistor values in divider (in ohms)
float R1 = 30000.0;
float R2 = 7500.0;

// Float for Reference Voltage
float ref_voltage = 5.0;

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

```
void setup(){
  // Setup Serial Monitor
  Serial.begin(9600);   }

void loop(){
  // Read the Analog Input
  adc_value = analogRead(ANALOG_IN_PIN);

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

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

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

  // Short delay
  delay(500);   }
```

Acs712 (Hall Effect) current sensor interfacing with Arduino

Acs712 is used for current sensing in industry, power sector and communication applications. Acs712 can measure both AC current and DC current. AC current measurement and DC current measurement has many applications. Like in **power systems protection**, ac current measurement techniques are used to measure **over load current** for protection of **transformers** and generators. And similarly, we use ac or dc current measurement circuits to design ac power meter or energy meters. It is also used in **three phase induction motor** current measurement for feedback control.

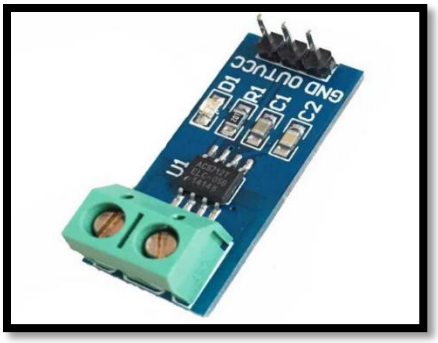


Figure 5.22 Current Sensor

Introduction to Acs712 current sensor

Acs712 is hall effect based current sensor. It can measure both direct current and alternating current. It is a linear type sensor. This is very a famous integrated circuit designed by Allegro. It has features of noise cancellation, very high response time. Output error is about 1.5 percent but it can tackled with some intelligent programming and multiplying measured value with standard error of sensor. If we give dc current to its input, it will give proportional dc voltage at the output of sensor and if we give ac current at the input of acs712, it will give us proportional ac voltage at the output. Proportional term depends on the output sensitivity of the sensor.

Working of acs712 current sensor

This acs712 sensor consists of a linear hall effect circuit along with copper conduction path. Copper conduction path is located around the surface of the die. When ac or dc current passes through a copper conduction path, it produces magnetic field. This electromagnetic field interacts with hall effect sensor. Hall effect circuit converts this electromagnetic field into proportional voltage either ac or dc depending on input current type. This output voltage is measured with the help of Arduino or any microcontroller. After measuring this voltage, we convert it back into current using sensitivity equations.

Pin diagram of acs712 hall effect current sensor

Pin out of acs712 current sensor is given below. Pin number 1, 2 and 3, 4 are used for current sampling. In other words. These pins are connected in series with the load of which current you want to measure.

Pin number 5 is ground connection of 5 volt power supply and pin number 6 is used to connect filter capacitor. One terminal of filter capacitor should be connected with pin number 6 and other terminal should be connected with ground. Similarly pin number 8 vcc is a power supply pin and we should connect dc 5 volt with it. Pin number 7 is the output pin of acs712 current sensor. From output pin, we will measure voltage with the help of Arduino. ***Make sure to not connect your load in parallel with IP+ and IP+ it will damage your device.***

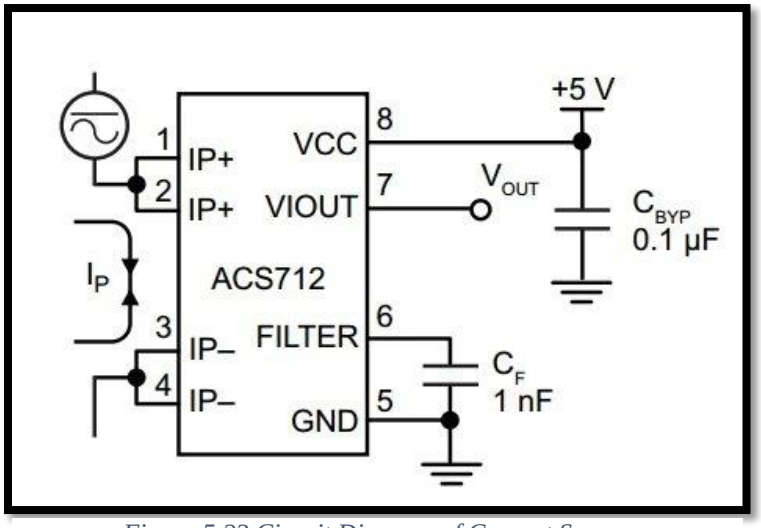


Figure 5.23 Circuit Diagram of Current Sensor

Number	Name	Description
1 and 2	IP+	Terminals for current being sampled; fused internally
3 and 4	IP–	Terminals for current being sampled; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

Table 5.1 Pinout of Current Sensor

How to measure current from output voltage of acs712 sensor?

To calculate current from output voltage of acs712 current sensor, we should make calculations according to following points:

- When there is no current flowing through the sensor, output voltage will be Vcc / 2. Where Vcc is power supply voltage given to acs712 to current sensor.
- if the Vcc = 5 volt, then the output voltage of current sensor will be equal to 2.5 when there is no current passing through a sensor.
- 2.5 volt is the offset voltage or base voltage of the sensor which should be subtracted from the measured voltage.
- The output voltage decreases when current starts passing through the sensor.
- So we can calculate dc current by using following commands:

```
Adcvalue = analogRead(A0);

Voltage = (adcvalue / 1024.0) * 5000;

current = ((Voltage – voltage_offset) / mVperAmp);
```

We can measure current by using above three lines of Arduino code. In the first line, we are using Arduino built in library analogRead function to measure output voltage of hall effect current sensor. The measured digital value is stored in variable 'Adcvalue'. In second line, we are converting digital value of voltage back into analog voltage in mili ampere by multiplying it with resolution factor and divided by 1000 to convert it into mili ampere voltage. In third line, measured voltage is subtracted from offset voltage and divided by sensitivity factor to get current from measured voltage.

Acs712 hall effect sensor interfacing with Arduino

Figure below shows the connection diagram of interfacing acs712 current sensor with Arduino. This sensor is also available in the form of module below:

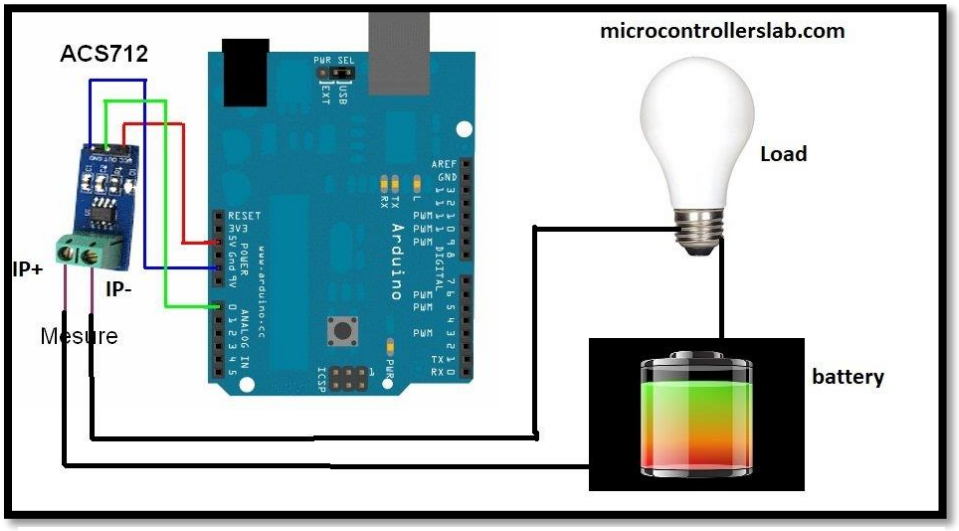


Figure 5.24 Interfacing Arduino with Current Sensor

Vcc pin of current sensor module is connected with 5V pin of Arduino and Ground pin is connected with Ground pin of Arduino and output pin of current sensor module is connected with Analog channel 0 of Arduino which built in **analog to digital converter** of Arduino. Load is connected in series with IP+ and IP- pin and dc battery.

Dc current measurement using acs712 current sensor and Arduino UNO example

Schematic and Proteus simulation for dc current measurement is given below:

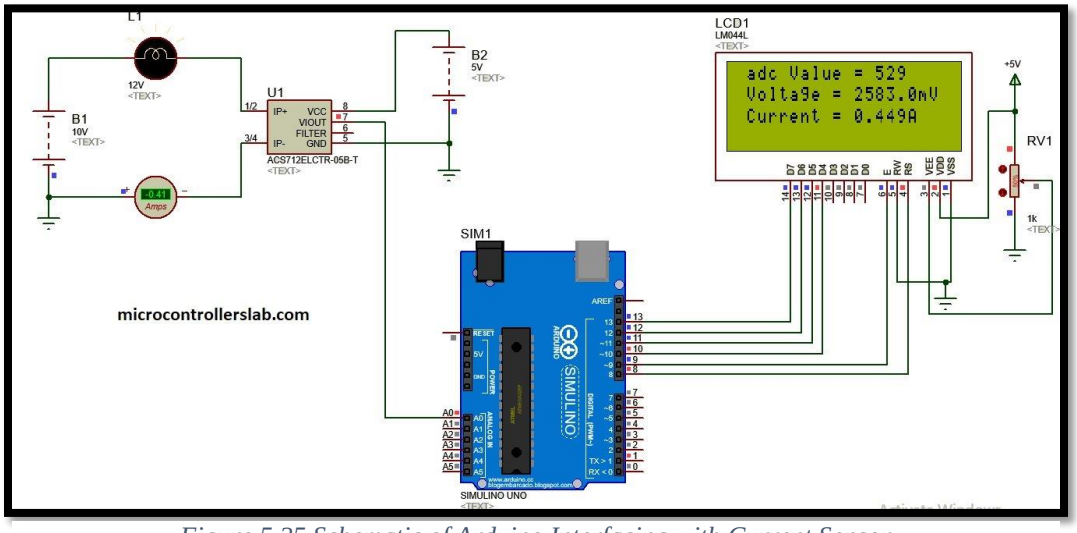


Figure 5.25 Schematic of Arduino Interfacing with Current Sensor

As shown on LCD first line is showing measured ADC value and second line is showing voltage and third line is showing measured which is exactly the same current we measured with virtual

ampere meter in Proteus. Code for dc current measurement using acs712 hall effect sensor is given below.

Code for dc current measurement using acs712 current sensor

```
// include the library code:
#include <LiquidCrystal.h> //library for LCD
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(8, 9, 10, 11, 12, 13);
//Measuring Current Using ACS712
const int analogchannel = 0; //Connect current sensor with A0 of Arduino
int sensitivity = 185; // use 100 for 20A Module and 66 for 30A Module
int adcvalue= 0;
int offsetvoltage = 2500;
double Voltage = 0; //voltage measuring
double ecurrent = 0;// Current measuring
void setup() {
 //baud rate
 Serial.begin(9600);//baud rate at which arduino communicates with Laptop/PC
 // set up the LCD's number of columns and rows:
 lcd.begin(20, 4); //LCD order
 // Print a message to the LCD.
 lcd.setCursor(1,1);//Setting cursor on LCD
 lcd.print("MICROCONTROLLERSLAB");//Prints on the LCD
 lcd.setCursor(4,2);
 lcd.print(".com");
 delay(3000);//time delay for 3 sec
 lcd.clear();//clearing the LCD display
 lcd.display();//Turning on the display again
```

```
 lcd.setCursor(1,0);//setting LCD cursor
 lcd.print("Reading Values from");//prints on LCD
 lcd.setCursor(1,1);
 lcd.print("DC Current Sensor");
 lcd.setCursor(5,2);
 lcd.print("ACS 712");
 delay(2000);//delay for 2 sec
}
void loop() //method to run the source code repeatedly
{
 adcvalue = analogRead(analogchannel);//reading the value from the
analog pin
 Voltage = (adcvalue / 1024.0) * 5000; // Gets you mV
 ecurrent = ((Voltage - offsetvoltage) / sensitivity);
//Prints on the serial port
 Serial.print("Raw Value = " ); // prints on the serial monitor
 Serial.print(adcvalue); //prints the results on the serial monitor
 lcd.clear();//clears the display of LCD
 delay(1000);//delay of 1 sec
 lcd.display();
 lcd.setCursor(1,0);
 lcd.print("adc Value = ");
 lcd.setCursor(13,0);
 lcd.print(adcvalue);
 Serial.print("\t mV = "); // shows the voltage measured
 Serial.print(Voltage,3); // the '3' after voltage allows you to
display 3 digits after decimal point
 lcd.setCursor(1,1);
 lcd.print("Voltage = ");
 lcd.setCursor(11,1);
```

```
 lcd.print(Voltage,3);
 lcd.setCursor(17,1);
 lcd.print("mV");//Unit for the voltages to be measured
 Serial.print("\t ecurrent = "); // shows the voltage measured
 Serial.println(ecurrent,3);// the '3' after voltage allows you to
display 3 digits after decimal point
 lcd.setCursor(1,2);
 lcd.print("Current = ");
 lcd.setCursor(11,2);
 lcd.print(ecurrent,3);
 lcd.setCursor(16,2);
 lcd.print("A"); //unit for the current to be measured
 delay(2500); //delay of 2.5 sec
}
```

Interfacing Micro SD Card Module with Arduino

SD Card:

SD stands for secure digital and combined with card it stands for secure digital card. It is a nonvolatile memory in a small thin integrated circuit. You can store still images, videos, files etc. on it. In order to move data from one device to another using SD-card you have to remove it from one device and insert it in other. SD cards use NAND chips (one type of flash memory) to store digital files. An SD card has a series of electronic components called NAND chips. The chips allow data from the host to be written and stored on the SD card and retain the data without a power supply.

Example:-

An embedded engineer in the electronic industry need to log and store a huge amount of data that the internal memory of an Arduino can handle, examples could be like any logger project like battery energy logger, Temperature logger or GPS Tracker. The solution to this problem is to use an **SD card or micro sd card** that packs gigabytes of data and its size is smaller than a one rupee coin.

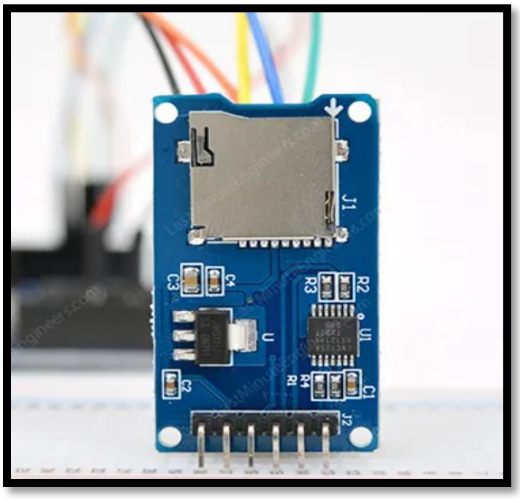


Figure 5.26 SD Card Module

Hardware Overview

This is a cheap and easy-to-use sensor that can be used for many different applications. This sensor can be used to read SD card data with microcontroller. The parts marking of the SD Card Module is shown below.

There are only three components that are significant, first is the Micro SD Card Holder Itself. This holder makes it easy for us to swap between different SD card modules. The second most important thing is the level shifter IC as the SD card module runs only on 3.3V and it has a maximum operating voltage of 3.6V so if we directly connect the SD card to 5V it will definitely kill the SD card. Also, the module has an onboard ultra-low dropout regulator that will convert the voltage from 5V to 3.3V. That is also why this module can operate on 3.3V power.

Mode of Communication in SD card – SPI vs SDIO

microSD cards can actually be interfaced in two ways: SPI mode and SDIO mode.

SDIO mode is much faster and is used in mobile phones, digital cameras, and other devices. However, it is more complicated and requires the signing of non-disclosure agreements. Because of this, hobbyists like us are unlikely to come across SDIO mode interface code.

Therefore, almost every SD card module employs the "lower speed and less overhead" SPI mode, which is simple to implement on any microcontroller.

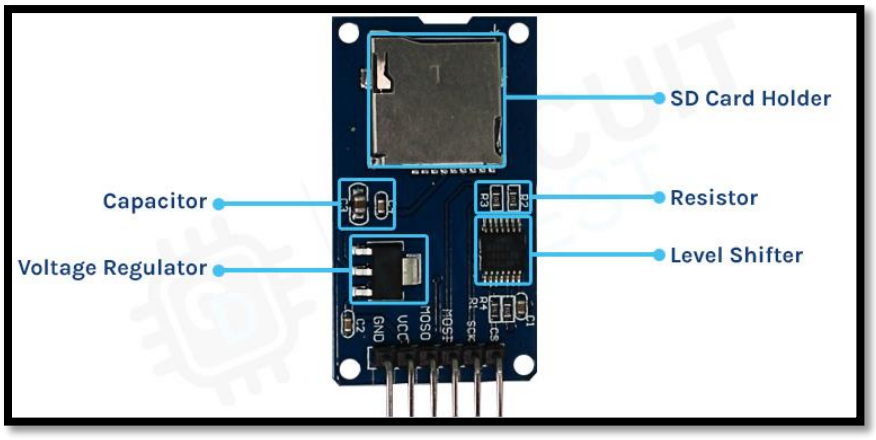


Figure 5.27 Parts of SD Card Module

MicroSD Card Module Pinout

The microSD card module is simple to connect. There are six pins on it:

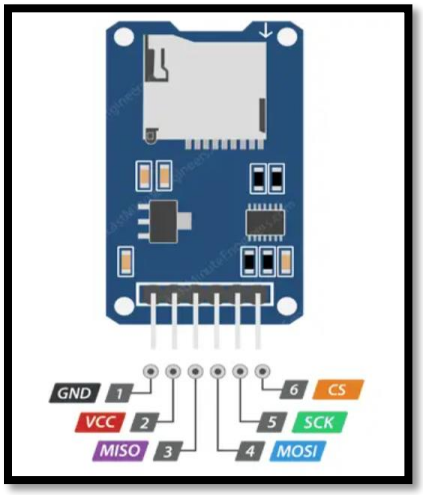


Figure 5.28 Pinout of SD Card Module

- **VCC** pin provides power to the module and should be connected to the Arduino's 5V pin.
- **GND** is a ground pin.
- **MISO (Master In Slave Out)** is the SPI output from the microSD card module.
- **MOSI (Master Out Slave In)** is the SPI input to the microSD card module.

- **SCK (Serial Clock)** pin accepts clock pulses from the master (an Arduino in our case) to synchronize data transmission.
- **CS (Chip Select)** pin is a control pin that is used to select one (or a set) of slave devices on the SPI bus.

Preparing the microSD card

Before inserting the microSD card into the module and connecting it to the Arduino, you must properly format the card to FAT16 or FAT32.

If you have a new SD card, chances are it's already pre-formatted with a FAT file system; however, you may encounter issues with how the factory formats the card. Or, if you have an old card, it needs to be formatted. In any case, it's a good idea to format the card before using it.

Wiring a microSD Card Module to an Arduino UNO

Connect the module's VCC pin to 5V on the Arduino and the GND pin to ground.

Now we are left with the pins that are used for SPI communication. Because microSD cards require a lot of data transfer, they perform best when connected to the microcontroller's hardware SPI pins.

Note that each Arduino UNO board has different SPI pins that must be connected correctly. For Arduino boards such as the UNO/Nano V3.0 those pins are digital 13 (SCK), 12 (MISO), 11 (MOSI) and 10 (CS).

The diagram below shows how to connect microSD Card Module to the Arduino UNO.

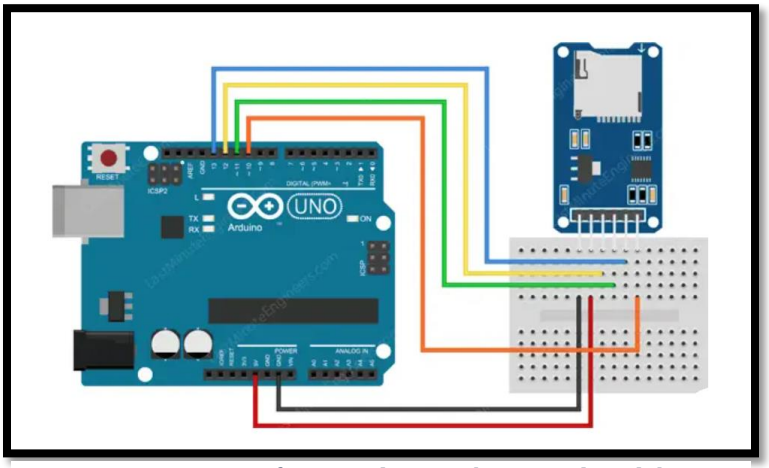


Figure 5.29 Interfacing Arduino with SD Card Module

Arduino Code – Testing the SD card module with Card Information

Communicating with an SD card is a lot of work, but luckily for us, the Arduino IDE already includes a very useful library called SD that makes reading and writing SD cards easier.

Let's start with a simple Card Information example sketch. This sketch doesn't write any data to the card. Instead, it tells us if the card is recognized and shows you some information about it. This can be very useful when determining whether or not an SD card is supported. It is therefore recommended that we run this sketch once before trying out a new card.

To open the Card Information example sketch, navigate to File > Examples > SD > Card Information**.**

Verify that the chip Select line is correctly initialized at the beginning of the sketch. In Arduino UNO case digital pin #10, so change it to 10.

Arduino Code – Reading and Writing Data

Assuming we were successful with the previous sketch, we will proceed to the next experiment. The following sketch will demonstrate how to write to a file and then verify its contents by reading it back.

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

```
#include <SD.h>
File myFile;
// change this to match your SD shield or module;
const int chipSelect = 10;
void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
  Serial.print("Initializing SD card...");
  if (!SD.begin()) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");

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

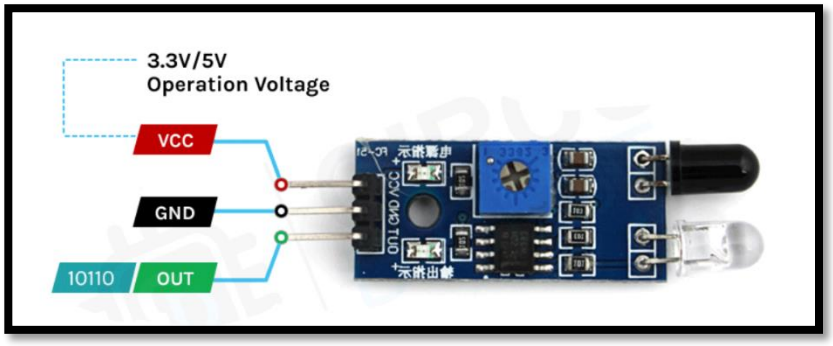
```
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
  // re-open the file for reading:
  myFile = SD.open("test.txt");
  if (myFile) {
    Serial.println("test.txt:");
    // read from the file until there's nothing else in it:
    while (myFile.available())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

Interfacing IR Sensor Module with Arduino

An **infrared proximity sensor or IR Sensor** is an electronic device that emits infrared lights to sense some aspect of the surroundings and can be employed to detect the motion of an object. As this is a passive sensor, it can only measure infrared radiation.

IR Sensor Pinout

The IR sensor has a 3 pin connector that interfaces it to the outside world. The connections are as follows:



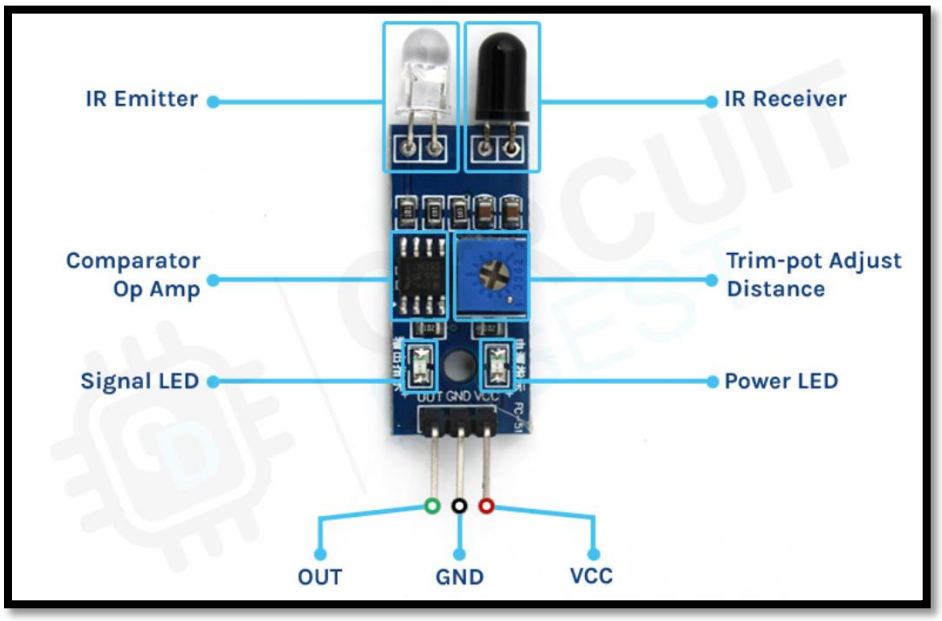
Working

The **working of the IR sensor module** is very simple, it consists of two main components: the first is the IR transmitter section and the second is the IR receiver section. In the transmitter section, **IR led** is used and in the receiver section, a **photodiode** is used to receive infrared signal.

An IR proximity sensor works by applying a voltage to the onboard **Infrared Light Emitting Diode** which in turn emits infrared light. This light propagates through the air and hits an object, after that the light gets reflected in the photodiode sensor. If the object is close, the reflected light will be stronger, if the object is far away, the reflected light will be weaker. If you look closely toward the module. When the sensor becomes active it sends a corresponding **Low signal** through the output pin that can be sensed by an Arduino or any kind of microcontroller to execute a particular task. The one good thing about this module is that it has two onboard LEDs built-in, one of which lights on when power is available and another one turns on when the circuit gets triggered.

IR Motion Sensor Module Parts

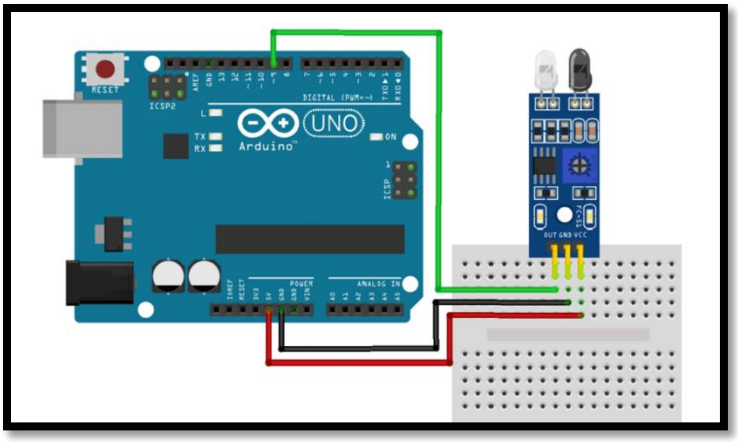
For most of the Arduino projects, this sensor is used to detect proximity or to build obstacle avoidance robots. This Sensor is popular among beginners as these are low power, low cost, rugged, and feature a wide sensing range that can be trimmed down to adjust the sensitivity.



This sensor has three pins two of which are power pins leveled VCC and GND and the other one is the sense/data pin which is shown in the diagram above. It has an onboard power LED and a signal LED the power LED turns on when power is applied to the board the signal LED turns on when the circuit is triggered. This board also has a comparator Op-amp that is responsible for converting the incoming analog signal from the photodiode to a digital signal. We also have a sensitivity adjustment potentiometer; with that, we can adjust the sensitivity of the device. Last and finally, we have the photodiode and the IR emitting LED pair which all together make the total IR Proximity Sensor Module.

IR Sensor interfacing with Arduino

Now that we have a complete understanding of how an IR sensor works, we can connect all the required wires to Arduino as shown below.



Connecting the IR sensor to any microcontroller is really simple. As we know this sensor outputs a digital signal and processing this signal is very easy. There exist two methods to do so first, you can always check the port in an infinite loop to see when the port changes its state from high to low, or the other way is to do it with an interrupt if you are making a complicated project the interrupt method is recommended. Power the IR with 5V or 3.3V and connect ground to ground. Then connect the output to a digital pin D9. Used a Male to Female Jumper wire to connect the IR sensor module with Arduino board.

Arduino Code for Interfacing IR Motion Sensor Module with Arduino

We initialize our code by declaring two global variables, the first one holds the pin value where the IR sensor is connected and the second one holds the value where the LED is connected

```
int IRSensor = 9; // connect IR sensor module to Arduino pin D9
int LED = 13; // connect LED to Arduino pin 13
```

Next, we have our setup function. In the setup function, we initialize the serial with 115200 baud. Next, we print a statement to check if the serial monitor window is properly working or not, and then we initialize the IRSensor pin as input and the LED pin as output.

```
void setup(){
  Serial.begin(115200); // Init Serial at 115200 Baud Rate.
  Serial.println("Serial Working"); // Test to check if serial is working or not
  pinMode(IRSensor, INPUT); // IR Sensor pin INPUT
  pinMode(LED, OUTPUT); // LED Pin Output
}
```

Next, we have our infinite loop. In the infinite loop, we first read the sensor pin with the **digitalRead()** function and store the value to **sensorStatus** variable. Then we check to see if the output of the sensor is high or low, if the output of the sensor is high that means no motion is detected, else motion is detected, we also print this status in the serial monitor window.

```
void loop(){
  int sensorStatus = digitalRead(IRSensor); // Set the GPIO as Input
  if (sensorStatus == 1) // Check if the pin high or not
  {
    // if the pin is high turn off the onboard Led
    digitalWrite(LED, LOW); // LED LOW
    Serial.println("Motion Detected!"); // print Motion Detected! on the serial monitor window
  }
  else  {
    //else turn on the onboard LED
    digitalWrite(LED, HIGH); // LED High
    Serial.println("Motion Ended!"); // print Motion Ended! on the serial monitor window
  }
}
```

5.3 Motor Selection Current/Voltage/Speed/Torque

For calculating Torque of a Motor:

(1)

Depend upon size and weight of the robot

Depend upon the "road" at which the robot moves, as it is flat, or does it have to go up and down hills, because need more motor power to go up hills.

Also depend upon the radius of the wheel as well.

Motor should be **D.C Geared Motor** for the high amount of torque.

Rough calculation for the Torque:-

Multiply **Coefficient of friction** (0.6 for rough ground) to **Mass of Robot** (in kg) to **Wheel radius** (in cm). You will get the total torque of the motors.

Let' say wheel radius is 7 cm. Then total torque by above formula will be 168 Kg-cm. And you have 2 motors driving it, then divide it by 2 i.e, **84 kg-cm** torque motor you need to drive. By geared motors this value gets further decreased.

(2)

To determine the torque required for the motors of robot, we would need to know some information about the specific application and operating conditions. This include factors such as the weight distribution of the robot, the terrain it will be operating on, and the maximum speed and acceleration.

In general, a rough estimate of the torque required can be calculated by multiplying the total weight of the robot **(in kg)** by the coefficient of friction between the robot's wheels and the surface it will be operating on. For example, if the coefficient of friction is **0.2,** the estimated torque required for each motor would be **40 kg x 0.2 = 8 Nm (Newton-meters).** However, this is just a rough estimate and it is important to consult with a robotics or mechanical engineer to ensure that the motors you choose have the appropriate torque and power to meet the specific requirements of your robot.

(3)

The torque of the motor depends on the **weight** of the robot, the **speed** of the robot, and the **responsiveness** of the robot. A **higher torque** means more force to move the robot, but also more **power consumption** and **less speed**. A lower torque means less force to move the robot, but also less power consumption and more speed. We need to find a balance between these factors to build a perfect line follower robot.

One way to estimate the torque required for your robot is to use this formula:

Torque (kg-cm) = (Weight of Robot (kg) x Wheel Radius (cm) x Acceleration (cm/s^2)) / (2 x Number of Motors)

For example, if our robot weighs 0.5 kg, has wheels with radius 3 cm, needs to accelerate at 100 cm/s^2, and has two motors, then the torque required is:

Torque = (0.5 x 3 x 100) / (2 x 2) = 37.5 kg-cm

This is much higher than the torque of your motor, which is 0.3 kg-cm. Therefore, we need to either increase the torque of your motor, or reduce the weight of your robot, or reduce the acceleration of your robot, or increase the number of motors.

5.4 Improvement of Line Follower Robot

How to improve line following for Robot:

It depends on the weight of the robot and the material we are using for frame (chassis). Use high powered motors with more torque and try to put ultrasonic or IR sensors on the sides for the corrections of its path.

Remember, a higher rpm motor doesn't mean the line follower will follow the line perfectly. For the Line follower robot that should be able to follow the line perfectly, will depend on 'how fast your sensors can track'.

LDR or IR sensor?

LDR (Light Dependent Resistor) has a reaction time close to 0.1s. It doesn't seem like too much. Consider that your robot is traveling at a speed of 1m/s, then the sensor will sense and send the data only once in 10cm (1m/s *0.1s). So one option here is to change the sensor to a better sensor like IR (Infra-Red) sensor.

Speed is not important for better accuracy, therefore add a rotary potentiometer to reduce the speed or in code reduce the speed. Therefore, concentrate on accuracy or sensitivity of the Robot for tracking and following the line accurately.

Chapter 6- Software/Firmware Design/Theoretical Design

6.1 Input Output Pin Outs

Inputs:

1. 2 IR Sensors
2. Ultrasonic Sensor
3. Voltage Sensor

4. Current Sensor

Outputs:

1. Left Motor
2. Right Motor
3. SD Card
4. LCD Display

6.2 Schematic Components

1. Arduino Mega 2560

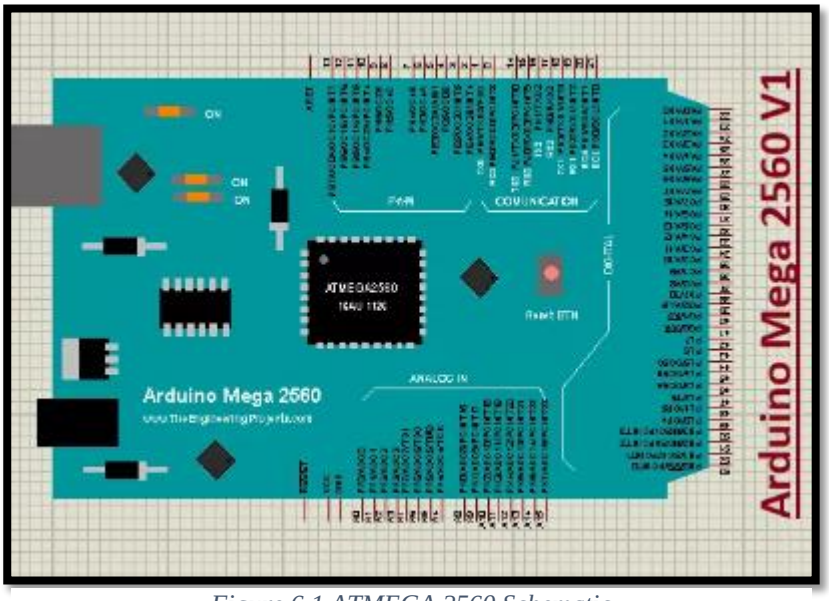


Figure 6.1 ATMEGA 2560 Schematic

2. SD Card Module

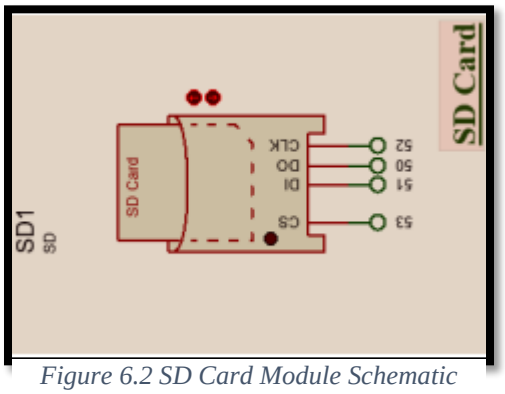


Figure 6.2 SD Card Module Schematic

3. L298N Motor Driver

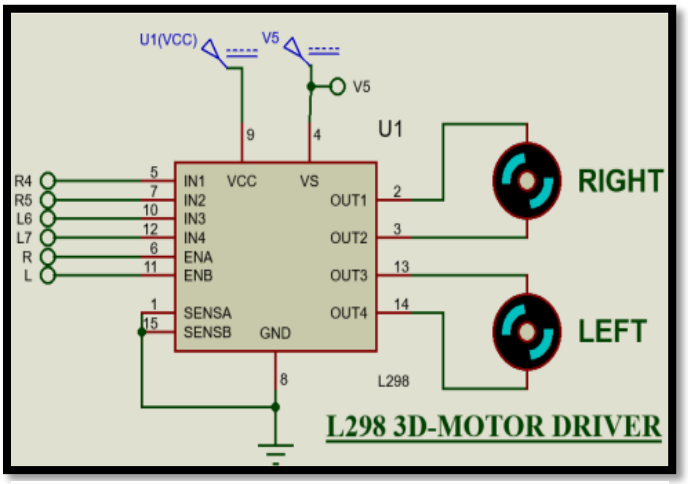


Figure 6.3 L298N Motor Driver Schematic

4. IR Sensor

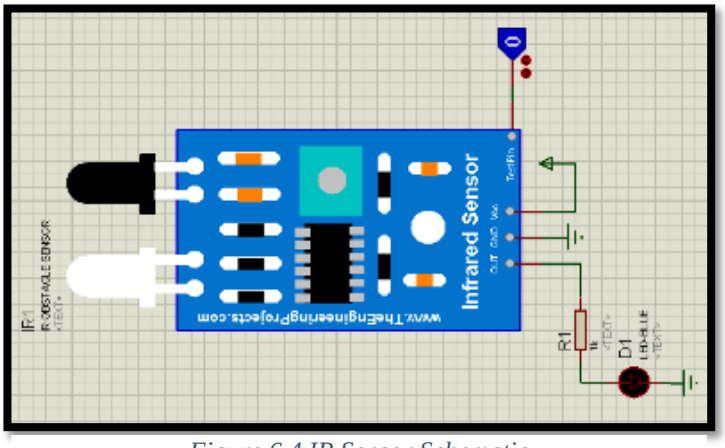


Figure 6.4 IR Sensor Schematic

5. LCD

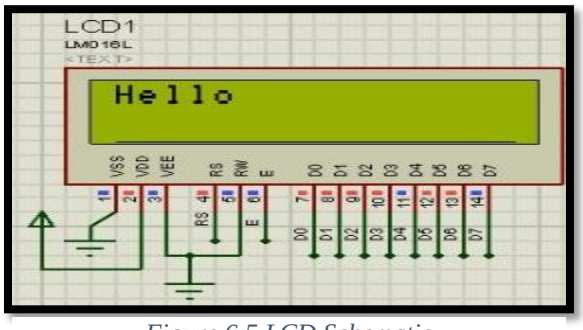


Figure 6.5 LCD Schematic

6. Ultrasonic Sensor

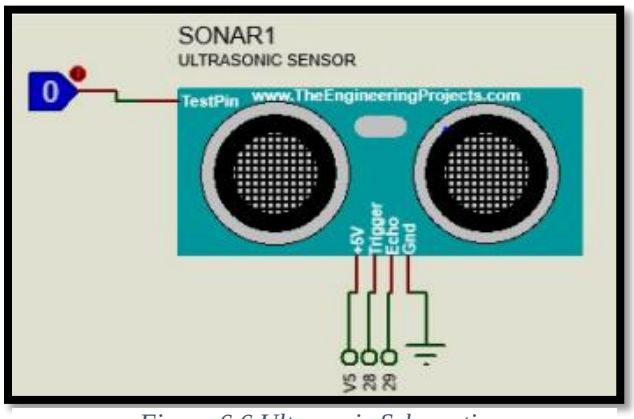


Figure 6.6 Ultrasonic Schematic

7. Potentiometer

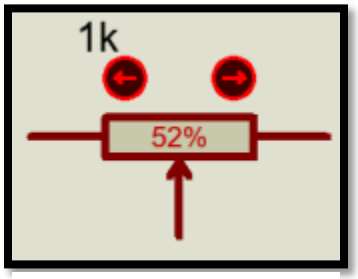


Figure 6.7 Potentiometer Schematic

6.3 Schematic Procedure

1. First, pick all the components from component library.
2. Add libraries of missing modules and components from the internet
3. Then place them onto sheet.
4. Designed the modules for Current & Voltage Sensor, LCD and L298 3D Motor Driver with motors.

5. Put a box around each set of components such as LCD Module, Motor Diver Module, Current & Voltage Sensor, Arduino Mega 2560, Ultrasonic Sensor etc. to keep schematic capture clean and understandable.
6. Search and place required Sil-headers and terminal blocks.
7. As all components are placed, connect the components with each other as required.
8. For long, complex, and intra-module connections, use Default and assigned the name according to connections
9. Then upload the hex file to At mega 2560.

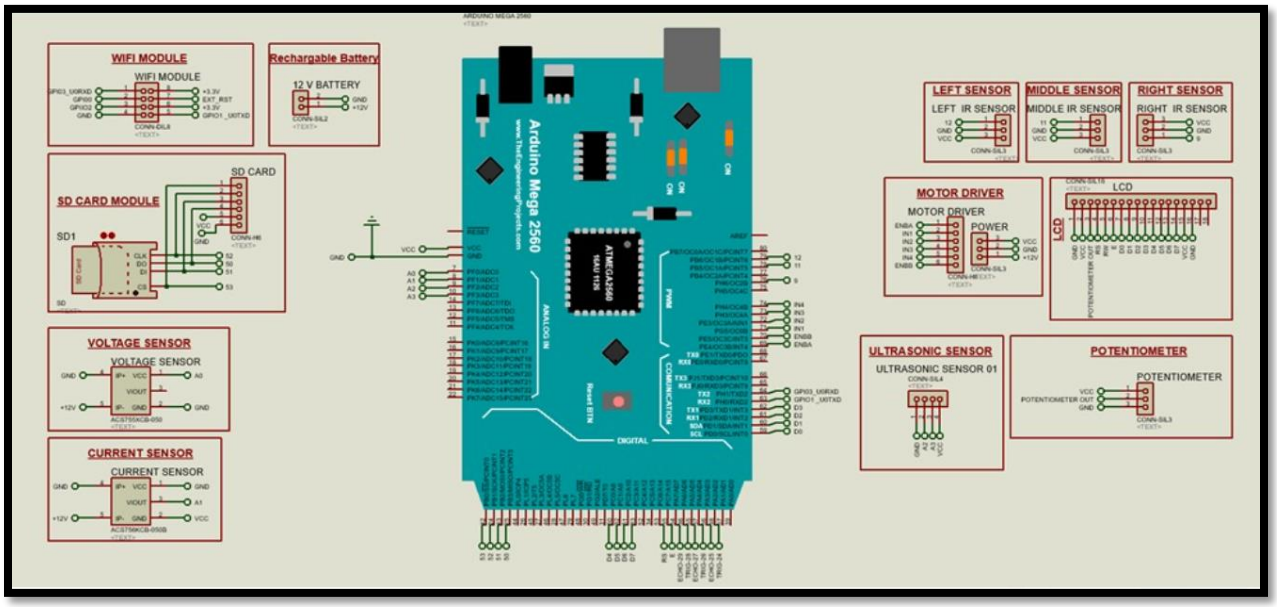


Figure 6.8 Schematic Circuit with Connectors

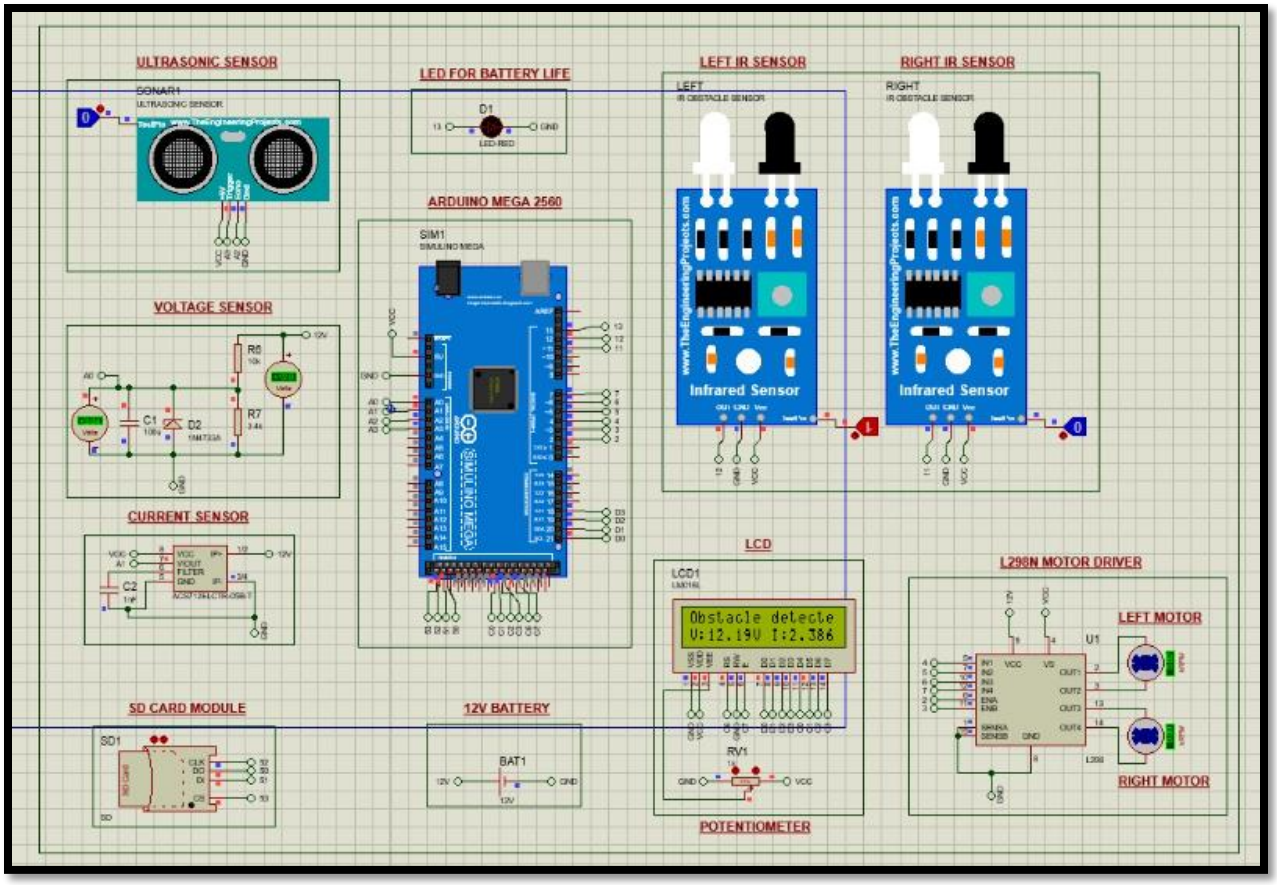


Figure 6.9 Final Schematic Circuit A

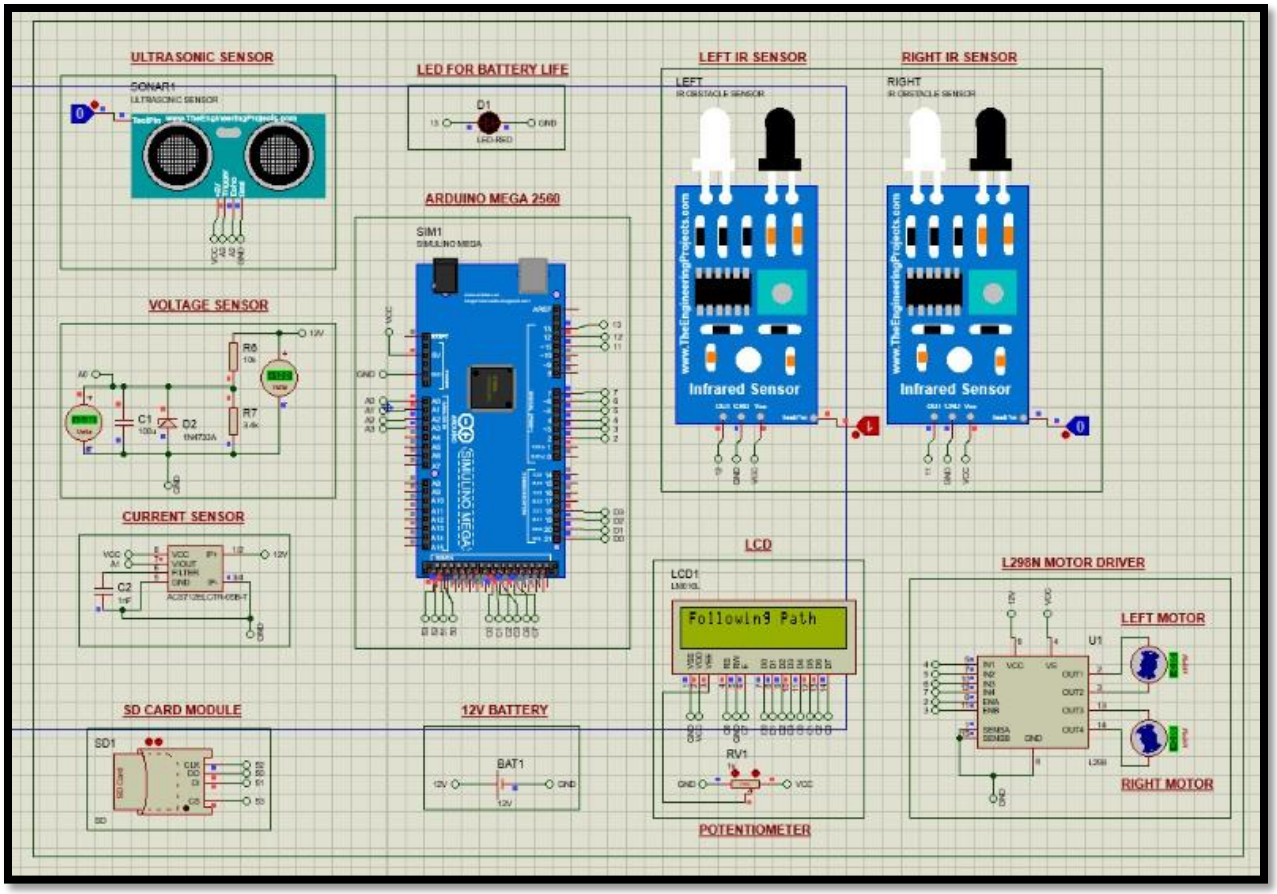


Figure 6.10 Final Schematic Circuit B

6.4 PCB Making Procedure

1. To design PCB in Proteus, there is an ARES designing tool is exist that we will use to design PCB.
2. After that we will create our design through the usage of 2D graphics box Mode.
3. Press at the select layer and now choose board edge, construct the box shape in the working area MT-359L Microcontroller and Embedded System AIR UNIVERSITY ISLAMABAD 65 and start designing PCB in the box.
4. Select Component Mode and place all the components on sheet.
5. Now design custom Modules for Voltage Boost Converter, Current & Voltage Sensors with proper dimensions between Vero pins as in hardware. Draw a box around them to further place anywhere as a module.

6. For Arduino Mega 2560, download the PCB layout from 'MY CREATIVE ENGINEERING' website. Then place this in PCB layout of project and assign pins according to pin configuration given in datasheet.
7. After that link all the components of circuit as thin green line guides.
8. Assign track width T40.
9. Give dimensions to box defining boundaries of PCB. Box has dimensions of 157.5mm X 120mm which is satisfactory according to our requirements.
10. For two-layer PCB connection and interlinking of all components can be made at both sides.
11. If there is any fault exist in our design will show in red color circles.
12. Red color track is shown for upper layer and the track of blue colored shown at lower portion among layers.
13. Use square connectors of S-70-30.
14. Then move on to 3D-visualizer to get an overview that how will circuit look in Hardware and it later helps in PCB printing, etching and components placing on etched PCB. We can analyze all joints of elements angles placements and other parameters.

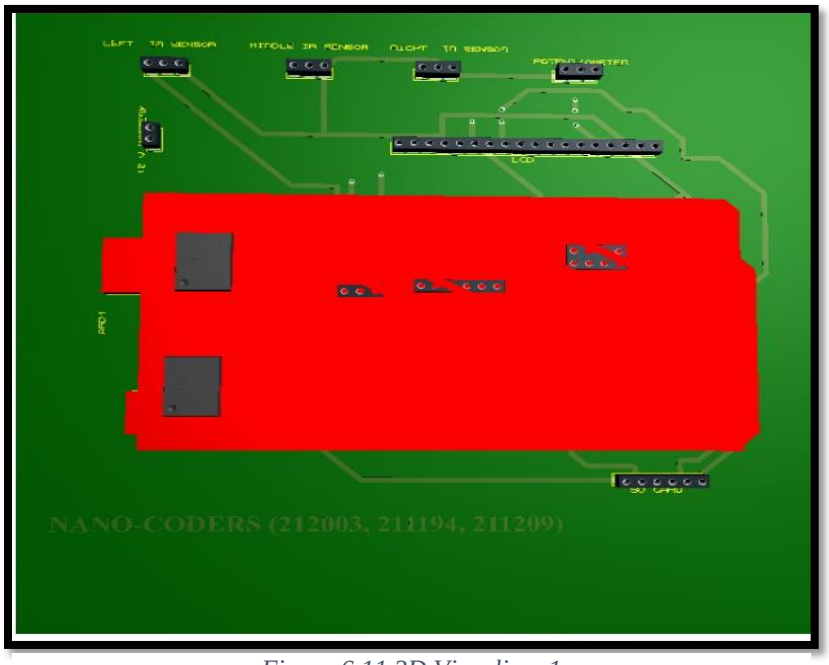


Figure 6.11 3D Visualizer 1

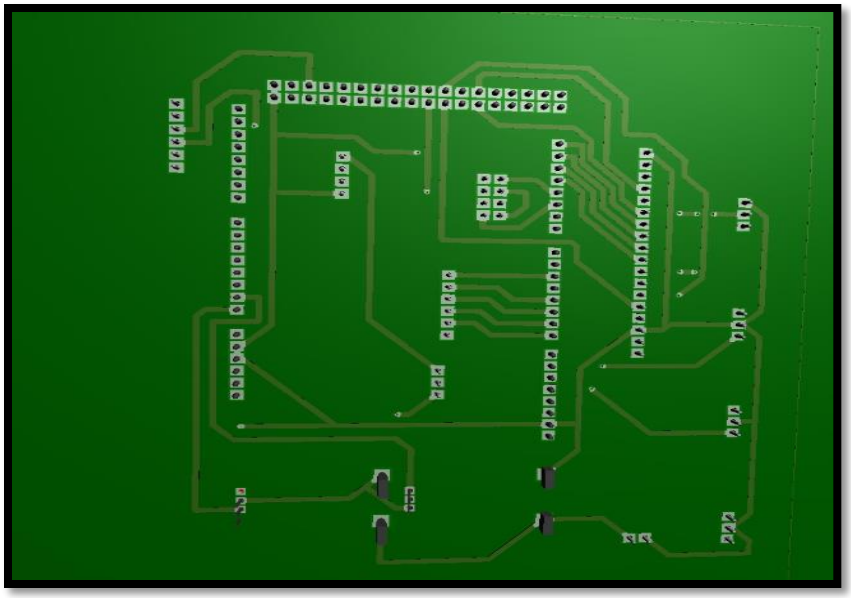


Figure 6.12 Figure 6.83 3D Visualizer 2

15. Save deigned PCB in PDF. Save front and back layers separately using Export Graphic > Export Adobe PDF File at 100% size.

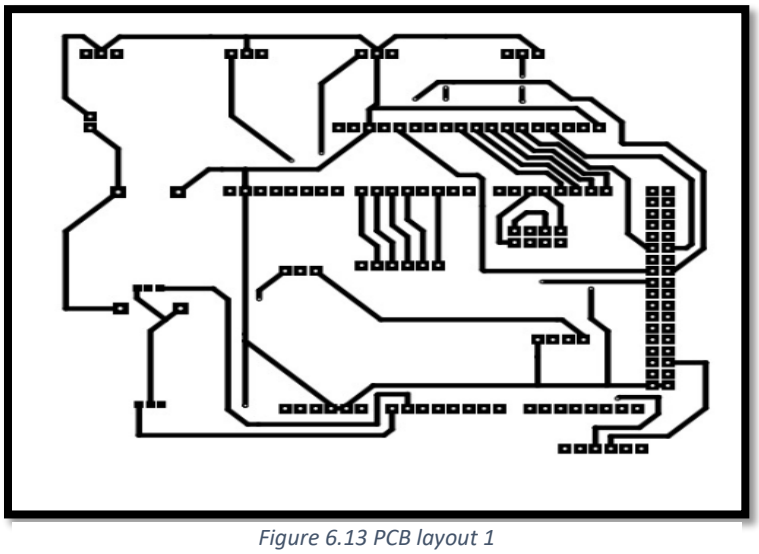


Figure 6.13 PCB layout 1

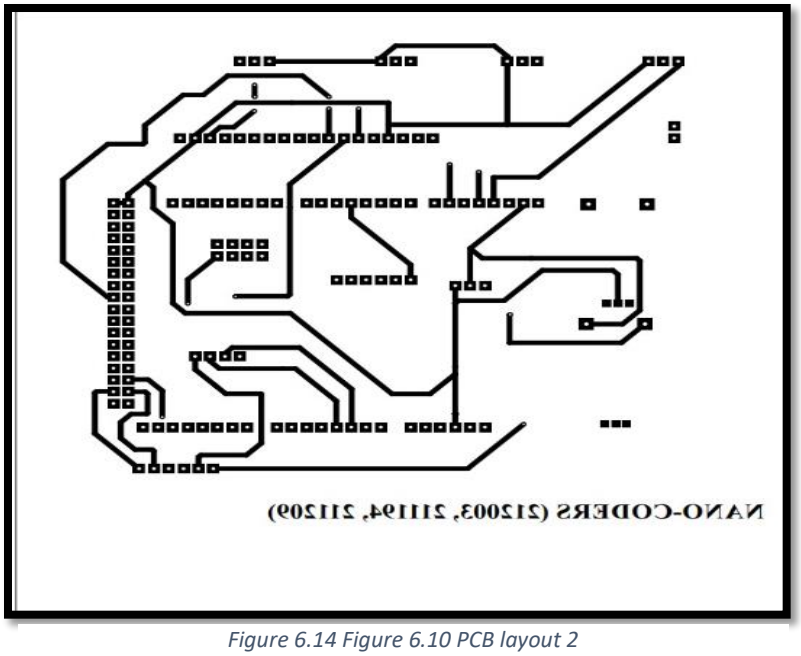


Figure 6.14 Figure 6.10 PCB layout 2

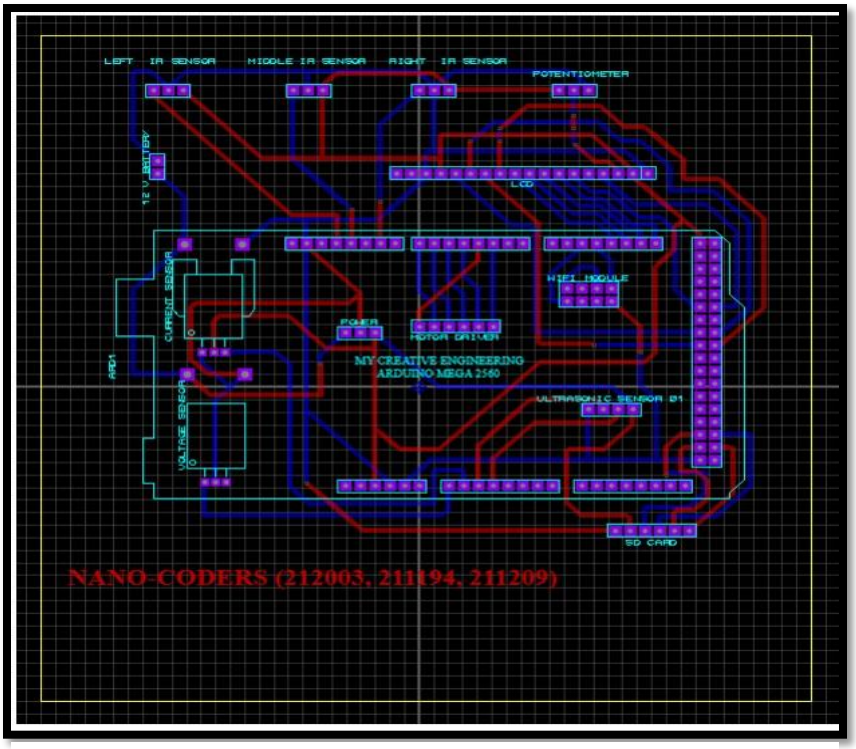


Figure 6.15 PCB Layout

6.5 Controller Selections with features

Line following robot detects and follows a line. The line is black line on a white surface and vice versa. The line is sensed by sensor, proximity sensor and IR sensor. The proximity sensor used for path detection and IR sensor used for obstacle detection. These sensors mounted at front end give

input to microcontroller (Arduino Mega 2560) which decides whether right motor or left motor will move to turn right, left, forward etc.

6.6 Programming and Sensor Interfacing

Interfacing of LCD

For the integration of the LCD module we just need to declare all the pins of the lcd and then connect it directly to the Arduino.

Code:

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(0, 1, 8, 9, 10, 11); /// REGISTER SELECT PIN,ENABLE PIN,D4 PIN,D5 PIN, D6 PIN, D7 PIN

void setup()
{
  // set up the LCD's number of columns and rows:
lcd.begin(16, 2);
}
void loop()
{
// set the cursor to column 0, line 1
lcd.print("   CIRCUIT DIGEST");//print name
lcd.setCursor(0, 1); // set the cursor to column 0, line 2
lcd.print("www.circuitdigest.com");//print name
delay(750);//delay of 0.75sec
lcd.scrollDisplayLeft();//shifting data on LCD

lcd.setCursor(0, 0);// set the cursor to column 0, line1
}
```

Interfacing of SD Card Module

For the integration of the SD card module, we need serial communication to be enabled between the controller and the module.

Code:

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

File myFile;

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

void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
  Serial.print("Initializing SD card...");

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

  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
```

```
  myFile = SD.open("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
}
```

Interfacing of Ultrasonic Sensor

Interfacing of Ultrasonic sensor with Arduino code is given below:

```
/*
* Ultrasonic Sensor HC-SR04 interfacing with Arduino.
*/
// defining the pins
const int trigPin = 9;
const int echoPin = 10;
// defining variables
long duration;
int distance;
void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); // Starts the serial communication
}
void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
```

```
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance= duration*0.034/2;
// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance);
}
```

Interfacing of IR Sensor

An infrared sensor is an electronic device, that emits to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measure only infrared radiation, rather than emitting it that is called as a passive IR sensor.

Code:

```
// Arduino IR Sensor Code
int IRSensor = 9; // connect ir sensor module to Arduino pin 9
int LED = 13; // conect LED to Arduino pin 13
void setup()
{
  Serial.begin(115200); // Init Serila at 115200 Baud
  Serial.println("Serial Working"); // Test to check if serial is working or not
  pinMode(IRSensor, INPUT); // IR Sensor pin INPUT
  pinMode(LED, OUTPUT); // LED Pin Output
}
void loop()
{
  int sensorStatus = digitalRead(IRSensor); // Set the GPIO as Input
  if (sensorStatus == 1) // Check if the pin high or not
  {
    // if the pin is high turn off the onboard Led
```

```
  digitalWrite(LED, LOW); // LED LOW
  Serial.println("Motion Ended!"); // print Motion Detected! on the serial monitor window
 }
 else
 {
  //else turn on the onboard LED
  digitalWrite(LED, HIGH); // LED High
  Serial.println("Motion Detected!"); // print Motion Ended! on the serial monitor window
 }
}
```

Interfacing of Current Sensor

ACS712 is a current sensor that can operate on both AC and DC. This sensor operates at 5V and produces an analog voltage output proportional to the measured current. This tool consists of a series of precision Hall sensors with copper lines. This sensor has an output voltage of Vcc x 0.5 = 2.5 at the input current 0A and a 5V Vcc power supply. There are three types based on the readable current range, ± 5A, ± 20A, and ± 30A with output sensitivity of each type of 185mV / A, 100mV / A, and 66mV / A respectively.

Code:

```
void
setup()
{
          Serial.begin(9600); //Start Serial Monitor to display current read value on Serial monitor
         }
         void loop() {
         unsigned int x=0;
         float AcsValue=0.0,Samples=0.0,AvgAcs=0.0,AcsValueF=0.0;
          for (int x = 0; x < 150; x++){ //Get 150 samples
          AcsValue = analogRead(A0);     //Read current sensor values
          Samples = Samples + AcsValue;  //Add samples together
          delay (3); // let ADC settle before next sample 3ms
```

```
}
AvgAcs=Samples/150.0;//Taking Average of Samples
//((AvgAcs * (5.0 / 1024.0)) is converitng the read voltage in 0-5 volts
//2.5 is offset(I assumed that arduino is working on 5v so the viout at no current comes
//out to be 2.5 which is out offset. If your arduino is working on different voltage than
//you must change the offset according to the input voltage)
//0.185v(185mV) is rise in output voltage when 1A current flows at input
AcsValueF = (2.5 - (AvgAcs * (5.0 / 1024.0)) )/0.185;
Serial.print(AcsValueF);//Print the read current on Serial monitor
delay(50);
}
```

Interfacing of Voltage Sensor

We want to check the voltage of the circuit at any point thn we will use this sensor. Voltage sensor basically measure the voltage at any certain. Voltage sensors can determine the AC voltage or DC voltage level. The input of this sensor is the voltage, whereas the output is the switches, analog voltage signal, a current signal, or an audible signal etc. Voltage Sensor Module 25V allows you to use the analog input of a microcontroller to monitor voltages much higher than it capable of sensing.

Code:

```
// Define analog input
#define ANALOG_IN_PIN A0

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

// Floats for resistor values in divider (in ohms)
float R1 = 30000.0;
float R2 = 7500.0;
```

```
// Float for Reference Voltage
float ref_voltage = 5.0;

// Integer for ADC value
int adc_value = 0;

void setup(){
  // Setup Serial Monitor
  Serial.begin(9600);
}

void loop(){
  // Read the Analog Input
  adc_value = analogRead(ANALOG_IN_PIN);

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

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

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

  // Short delay
  delay(500);
}
```

Final complete Code for Line Follower Robot

```
// Include necessary libraries

#include <LiquidCrystal.h>  // Library for interfacing with LCD

#include <SD.h>             // Library for SD card functionality

// LCD setup
```

```
const int rs = 31, en = 30, d0 = 21, d1 = 20, d2 = 19, d3 = 18, d4 = 37, d5 = 36, d6 = 35, d7 = 34;
LiquidCrystal lcd(rs, en, d0, d1, d2, d3, d4, d5, d6, d7);  // Initialize LCD object
// Definitions for motor control pins
#define IR_SENSOR_RIGHT 11   // Right IR sensor pin
#define IR_SENSOR_LEFT 12    // Left IR sensor pin
#define MOTOR_SPEED 100      // Default motor speed
int enableRightMotor = 2;    // Motor enable pin for the right motor
int rightMotorPin1 = 4;      // Right motor control pin 1
int rightMotorPin2 = 5;      // Right motor control pin 2
int enableLeftMotor = 3;     // Motor enable pin for the left motor
int leftMotorPin1 = 6;       // Left motor control pin 1
int leftMotorPin2 = 7;       // Left motor control pin 2
// Definitions for analog sensors
#define ANALOG_IN_PIN A0     // Analog input pin for voltage measurement
const int ledPin = 13;       // Pin for an LED
// Definitions for ultrasonic sensor
#define echoPin A2           // Pin connected to the echo output of the ultrasonic sensor
#define trigPin A3           // Pin connected to the trigger input of the ultrasonic sensor
// Variables for voltage and current calculations
float adc_voltage = 0.0;     // Variable to store calculated voltage from analog sensor
float in_voltage = 0.0;      // Variable to store voltage across the resistor
float R1 = 30000.0;          // Resistance R1 value
float R2 = 7500.0;           // Resistance R2 value
```

```
float ref_voltage = 4.0;      // Reference voltage for analog sensor
int adc_value = 0;            // Raw ADC value from analog sensor
const int analogchannel = 1;  // Analog channel for current measurement
int sensitivity = 185;        // Sensitivity for current measurement
int adcvalue = 0;             // Raw ADC value for current measurement
int offsetvoltage = 2500;     // Offset voltage for current measurement
double Voltage = 0;           // Calculated voltage for current measurement
double ecurrent = 0;          // Calculated current
// Variables for ultrasonic sensor
long duration;                // Variable to store the duration of the ultrasonic pulse
int distance;                 // Variable to store calculated distance from ultrasonic sensor
// Variables for timed operations
unsigned long previousMillis = 0;  // Variable to store the last time LCD was updated
const long interval = 1000;        // Interval for updating LCD (1 second)
// File for storing path information
File pathFile;
// Setup function
void setup() {
  // Set timer for analogWrite to improve PWM on certain pins
  TCCR0B = TCCR0B & B11111000 | B00000010;
  // Motor control pin configuration
  pinMode(enableRightMotor, OUTPUT);
  pinMode(rightMotorPin1, OUTPUT);
```

```
pinMode(rightMotorPin2, OUTPUT);
pinMode(enableLeftMotor, OUTPUT);
pinMode(leftMotorPin1, OUTPUT);
pinMode(leftMotorPin2, OUTPUT);
// IR sensor pin configuration
pinMode(IR_SENSOR_RIGHT, INPUT);
pinMode(IR_SENSOR_LEFT, INPUT);
// Ultrasonic sensor pin configuration
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
// Initialize motors
rotateMotor(0, 0);
// Initialize LCD
lcd.begin(16, 2);
lcd.print("Line Follower");
delay(1500);
lcd.clear();
// Analog sensor pin configuration
pinMode(ANALOG_IN_PIN, INPUT);
pinMode(ledPin, OUTPUT);
// Serial communication initialization
Serial.begin(9600);
// Initialize SD card
```

```
  if (SD.begin()) {
    Serial.println("SD card is ready to use.");
    pathFile = SD.open("path.txt", FILE_WRITE);
    if (pathFile) {
      pathFile.println("Sample Path"); // Add your initial path data here
      pathFile.close();
    } else {
      Serial.println("Error opening path file.");
    }
  } else {
    Serial.println("SD card initialization failed.");
  }
}
// Main loop function
void loop() {
  // Get current time
  unsigned long currentMillis = millis();
  // Read analog sensor values
  adc_value = analogRead(ANALOG_IN_PIN);
  adc_voltage = (adc_value * ref_voltage) / 1024.0;
  in_voltage = adc_voltage * (R1 + R2) / R2
  // Read current sensor values
  adcvalue = analogRead(analogchannel);
```

```
Voltage = (adcvalue / 1024.0) * 3000;
ecurrent = ((Voltage - offsetvoltage) / sensitivity);
// Display voltage and current on LCD
lcd.setCursor(0, 1);
lcd.print("V:");
lcd.print(in_voltage, 2);
lcd.print("V");
lcd.setCursor(9, 1);
lcd.print("I:");
lcd.print(ecurrent, 3);
lcd.print("A");
lcd.setCursor(0, 0);
// Trigger ultrasonic sensor
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = duration * 0.034 / 2;
// Check for obstacles
if (distance < 20) {
  rotateMotor(0, 0);
```

```
  lcd.print("Obstacle detected!");
  delay(3000);
  lcd.clear();
  handleObstacle();
 } else {
  // Read IR sensor values for line following
  int rightIRSensorValue = digitalRead(IR_SENSOR_RIGHT);
  int leftIRSensorValue = digitalRead(IR_SENSOR_LEFT);
  // Adjust movement based on IR sensor readings
  if (rightIRSensorValue == LOW && leftIRSensorValue == LOW) {
   rotateMotor(MOTOR_SPEED, MOTOR_SPEED);
   lcd.print("Forward");
  } else if (rightIRSensorValue == HIGH && leftIRSensorValue == LOW) {
   rotateMotor(-MOTOR_SPEED, MOTOR_SPEED);
   lcd.print("Right");
  } else if (rightIRSensorValue == LOW && leftIRSensorValue == HIGH) {
   rotateMotor(MOTOR_SPEED, -MOTOR_SPEED);
   lcd.print("Left");
  } else {
   rotateMotor(0, 0);
   lcd.print("Stop");
  }
 }
```

```
  // Display voltage and current every second
  if (currentMillis - previousMillis >= interval) {
    lcd.setCursor(0, 1);
    lcd.print("V:");
    lcd.print(in_voltage, 2);
    lcd.print("V")
    lcd.setCursor(9, 1);
    lcd.print("I:");
    lcd.print(ecurrent, 3);
    lcd.print("A");
    previousMillis = currentMillis;
  }
}
// Function to control motor movement
void rotateMotor(int rightMotorSpeed, int leftMotorSpeed) {
  // Control right motor
  if (rightMotorSpeed < 0) {
    digitalWrite(rightMotorPin1, LOW);
    digitalWrite(rightMotorPin2, HIGH);
  } else if (rightMotorSpeed > 0) {
    digitalWrite(rightMotorPin1, HIGH);
    digitalWrite(rightMotorPin2, LOW);
  } else {
```

```
    digitalWrite(rightMotorPin1, LOW);
    digitalWrite(rightMotorPin2, LOW);
  }
  // Control left motor
  if (leftMotorSpeed < 0) {
    digitalWrite(leftMotorPin1, LOW);
    digitalWrite(leftMotorPin2, HIGH);
  } else if (leftMotorSpeed > 0) {
    digitalWrite(leftMotorPin1, HIGH);
    digitalWrite(leftMotorPin2, LOW);
  } else {
    digitalWrite(leftMotorPin1, LOW);
    digitalWrite(leftMotorPin2, LOW);
  }
  // Set motor speeds
  analogWrite(enableRightMotor, abs(rightMotorSpeed));
  analogWrite(enableLeftMotor, abs(leftMotorSpeed));
}
// Function to handle obstacles
void handleObstacle() {
  // Perform a 360-degree turn
  rotateMotor(MOTOR_SPEED, -MOTOR_SPEED);
  delay(2000); // Adjust duration for a complete turn
```

```
  // Stop after the turn
  rotateMotor(0, 0);
  // Continue following the stored path
  followStoredPath();
}
// Function to follow the stored path
void followStoredPath() {
  lcd.clear();
  lcd.print("Following Path");
  // Open the path file for reading
  pathFile = SD.open("path.txt");
  if (pathFile) {
    while (pathFile.available()) {
      char character = pathFile.read();
      lcd.write(character);
      delay(100); // Adjust delay for path display speed
    }
    pathFile.close();
  } else {
    Serial.println("Error opening path file for reading.");
  }
  delay(3000); // Adjust delay after path display
  lcd.clear(); }
```

Chapter 7-Simulations and Final Integration (Test Definition Phase)

7.1 Integrations and Testing all Hardware and Software Component Separately

IR sensor:

The application circuit of the IR sensor is an obstacle detecting circuit that is shown below. This circuit can be built with a photodiode, IR LED, an Op-Amp, LED & a potentiometer, The main function of an infrared LED is to emit IR light and the photodiode is used to sense the IR light. In this circuit, an operational amplifier is used as a voltage comparator and the output of the sensor can be adjusted by the potentiometer based on the requirement. Once the light generated from the infrared LED can be dropped on the photodiode once striking an object, then the photodiode's resistance will be dropped.

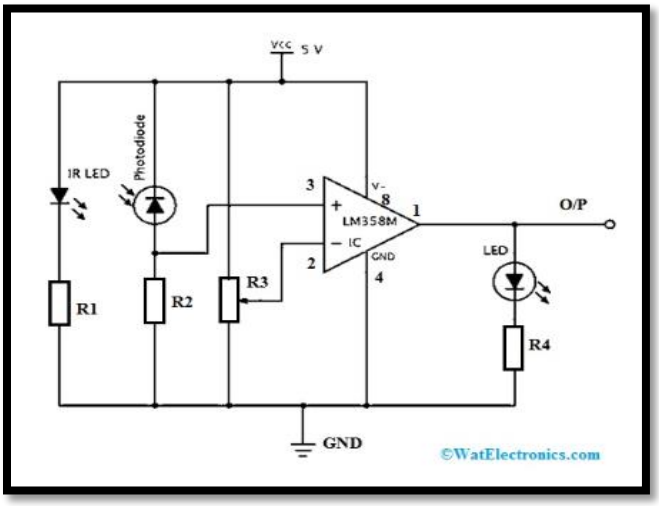


Figure 7.1 IR Sensor Circuit Diagram

IR Sensor Module

The IR sensor module includes five essential parts like IR Tx, Rx, Operational amplifier, trimmer pot (variable resistor) & output LED. The pin configuration of the IR sensor module is discussed below.

The main **specifications and features of the IR sensor** module include the following.

- The operating voltage is 5VDC
- I/O pins – 3.3V & 5V
- Mounting hole

- The range is up to 20 centimeters
- The supply current is 20mA
- The range of sensing is adjustable
- Fixed ambient light sensor

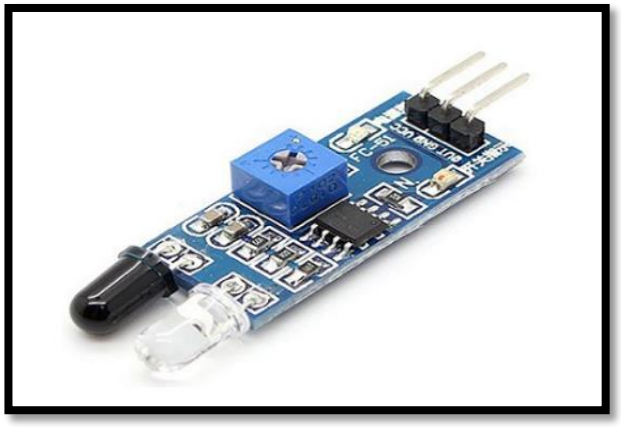


Figure 7.2 IR Sensor Module

Calibration for black/white color detection:

- Mount the IR sensor about 1 to 2 inches above a black surface.
- Turn the potentiometer fully clockwise.
- Slowly turn the potentiometer counterclockwise until the signal LED just turns off.
- Do not change the distance between the IR sensor and the black surface. If you need to change the distance then you need to re-calibrate.
- The signal LED will turn on when the sensor is above a white surface and off when it is above a black surface.

Making the connections

- Connect VCC on the IR sensor to 5V on the Arduino.
- Connect GND on the IR sensor to GND on the Arduino.
- Connect OUT on the IR sensor to pin 2 on the Arduino.

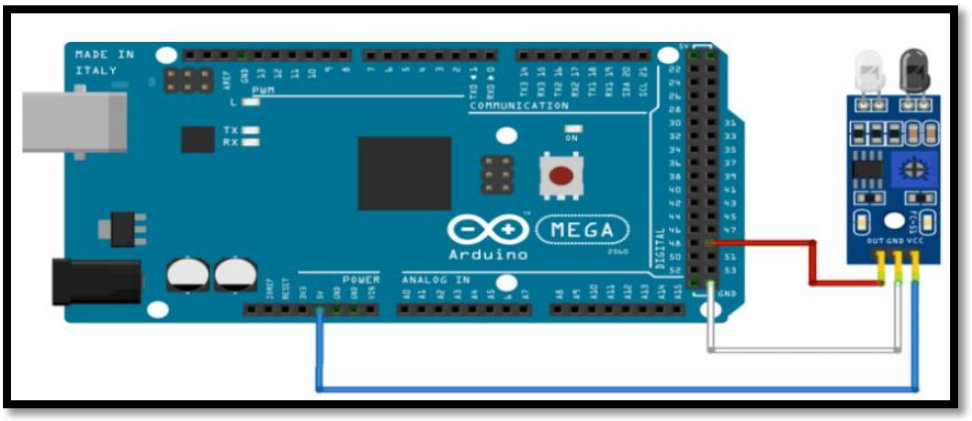


Figure 7.3 Interfacing of IR with Arduino At Mega 2560

Ultrasonic sensor:

- **What is an Ultrasonic Sensor?**
- An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves.

  An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity.

  High-frequency sound waves reflect across boundaries to produce distinct echo patterns.

- **How Ultrasonic Sensors Work**
- Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse.
- The working principle of this module is simple. It sends an ultrasonic pulse out at 40 kHz, which travels through the air, and if there is an obstacle or object, it will bounce back to the sensor. By calculating the travel time and the speed of sound, the distance can be calculated.
- Ultrasonic sensors are a great solution for the detection of clear objects. For liquid level measurement, applications that use infrared sensors, for instance, struggle

with this particular use case because of target translucence. For presence detection, ultrasonic sensors detect objects regardless of color, surface, or material (unless the material is very soft, like wool, as it would absorb sound).

To detect transparent and other items where optical technologies may fail, ultrasonic sensors are a reliable choice.

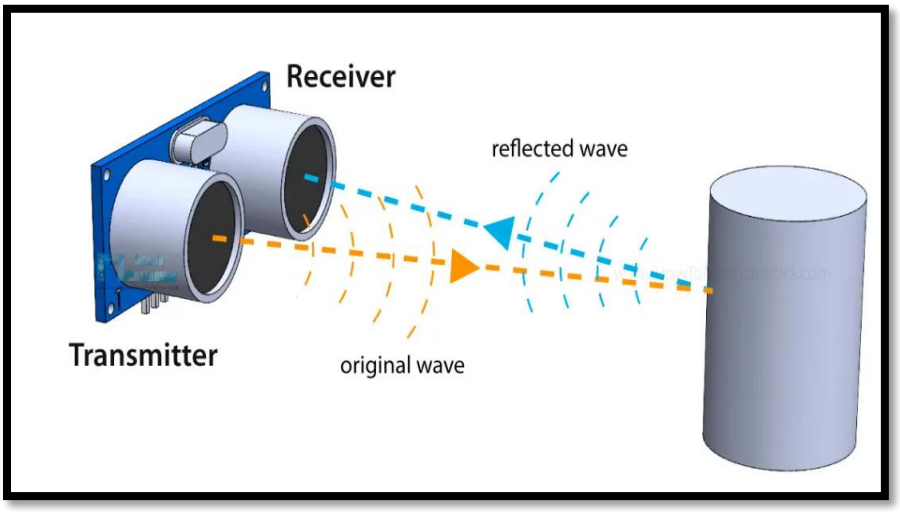


Figure 7.4 Working of Ultrasonic Sensor

Making the connections:

The connections are as follows:

- VCC to 5 Pin of Arduino
- GND to GND Pin of Arduino
- Trig to Digital Pin 9
- Echo to Digital Pin 10

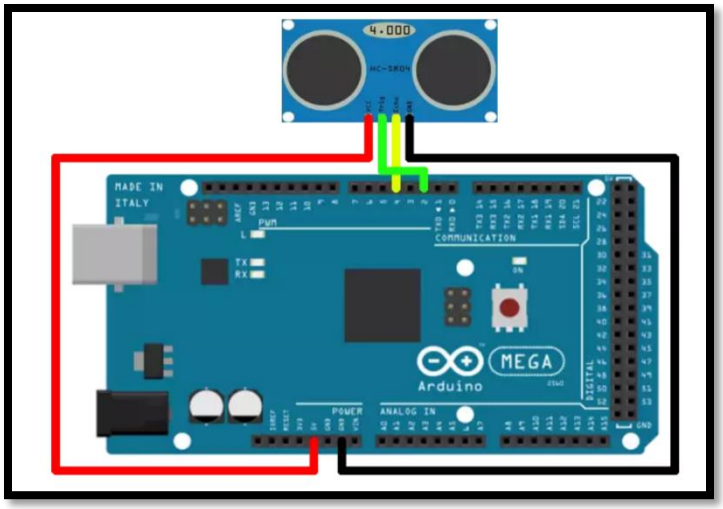


Figure 7.5 Interfacing of Ultrasonic with Arduino At Mega 2560

Voltage sensor:

The Mega 2560, like many microcontrollers, has a built-in Analog to Digital Converter (ADC) that can convert an analog voltage on a pin to a digital number. However, the maximum analog input pin voltage is limited to 5V. You may find this limit inconvenient if your project requires measuring voltages exceeding 5V. In such cases, you could create a voltage divider using discrete resistors.

But there's an easier way to measure voltages, especially if they're lower than 25V: use a Voltage Sensor. It is a pre-made voltage divider circuit that uses precision resistors to provide accurate readings.

Hardware Overview

The Voltage Sensor, in essence, is a simple voltage divider circuit composed of two resistors. There are two resistors in this circuit. The resistor (R1) closest to the input voltage, has a value of 30 KΩ, and the resistor (R2) closest to ground, has a value of 7.5 KΩ. The voltage drop across R2 is our divided voltage. This signal is broken out to a header pin labeled S.

This simple circuit divides the input voltage by a factor of 5. That's why this voltage sensor can help you measure voltages that are less than 25 volts with an Arduino.

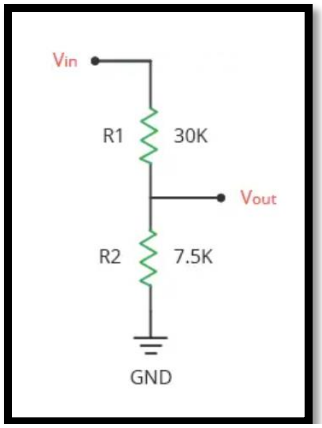


Figure 7.6 Circuit Diagram of Voltage Sensor

Voltage Sensor Pinout

Input Terminal

VCC is connected to the positive terminal of the voltage source you want to measure. The recommended voltage range for this pin is 0 to 25V.

GND is connected to the negative terminal of the input voltage source.

Output Header

S is the signal output pin of the voltage sensor module. It provides an analog voltage that is proportional to the input voltage level. It's usually connected to one of the analog input pins on the Arduino.

+ is not connected to anything.

– is the common ground pin.

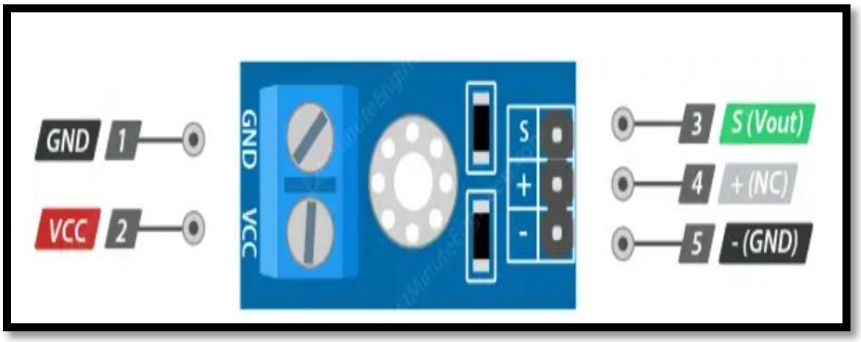


Figure 7.7 Voltage Sensor Pin Configuration

Hardware Hookup

To begin, connect the voltage source that you want to measure to the input screw terminal. Then, connect the 'S' pin on the voltage sensor to the 'A0' analog pin on the Arduino and the '-' pin to ground.

The image below shows how to connect everything.

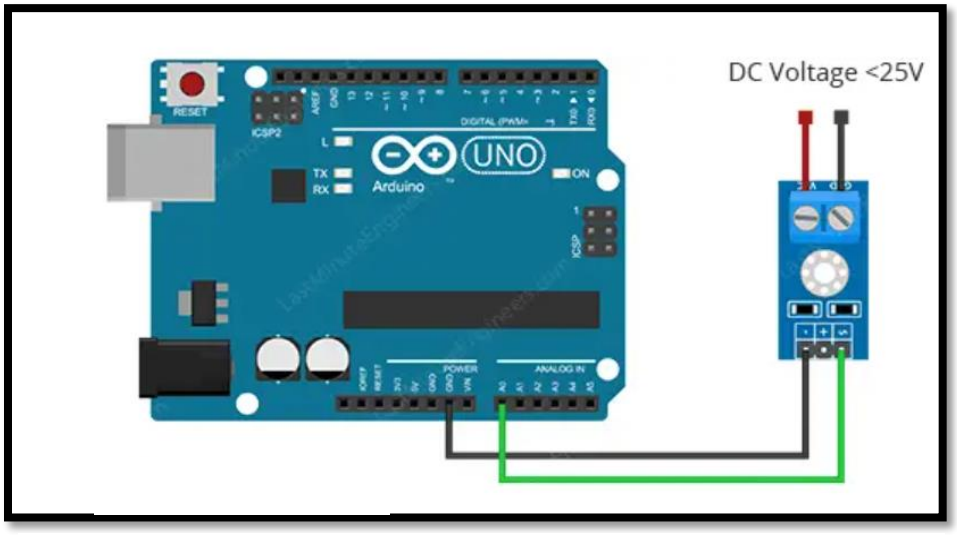


Figure 7.8 Interfacing of Voltage Sensor with Arduino At Mega 2560

Current sensor:

The ASC712 is based on Hall Effect. There is a copper strip connecting the IP+ and IP- pins internally. When some current flows through this copper conductor, a magnetic field is generated which is sensed by the Hall Effect sensor.

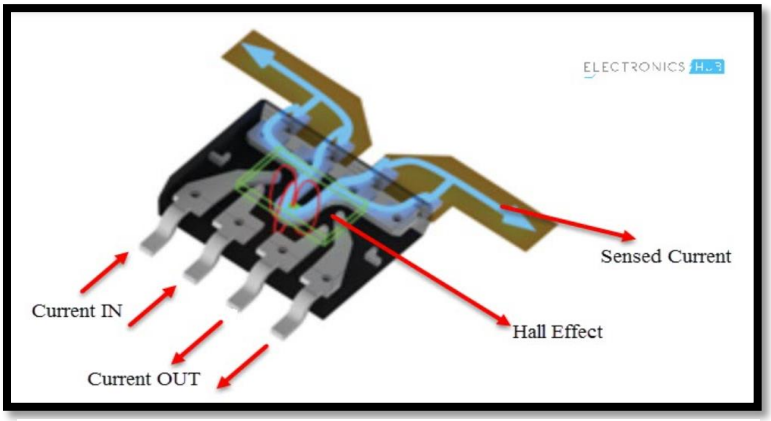


Figure 7.9 Pin Configuration of Current Sensor

ASC712 Current Sensor Application Circuit

The typical application circuit using the ASC712 Current Sensor is given in its datasheet and the following images shows the same.

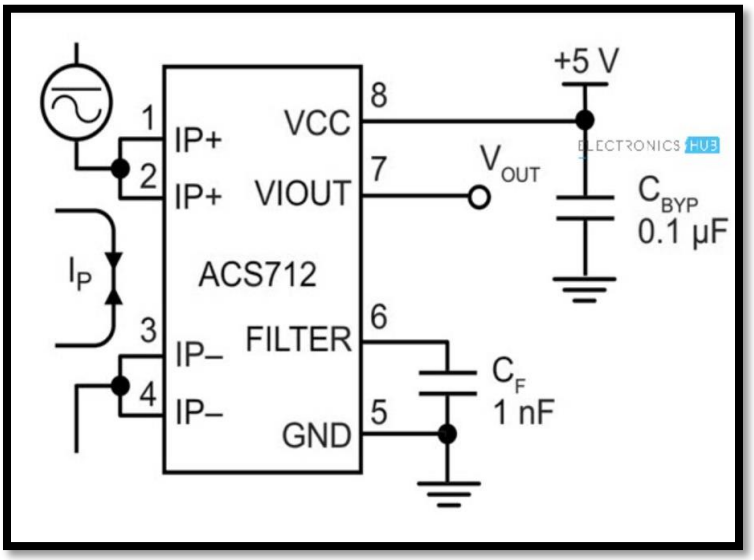


Figure 7.10 Circuit Diagram of Current Diagram

ACS712 Current Sensor Module

Using one of the variants of the ACS712 IC (5A, 20A or 30A), several manufacturers developed ASC712 Current Sensor Module boards that can be easily interfaced to a microcontroller like Arduino.

The following image shows the ASC712 Current Sensor board used in this project. This particular board consists of ASC712 ELC-30 i.e. the range of this board is +/- 30A. The following image shows the components and pins on the board.

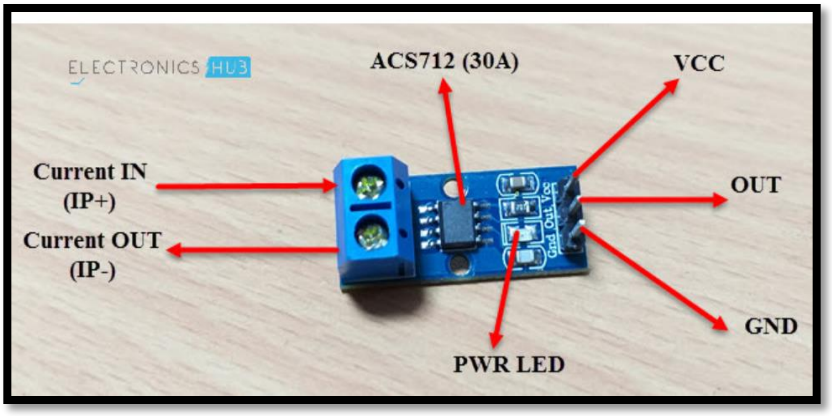


Figure 7.11 Current Sensor Module

Circuit Diagram of ASC712 Current Sensor with Arduino

The circuit diagram of interfacing ACS712 Current Sensor with Arduino is shown in the following image.

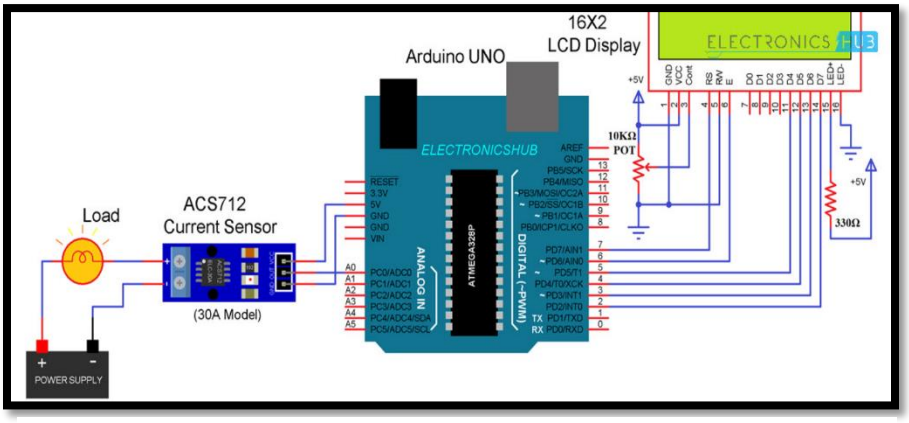


Figure 7.12 Interfacing of Current Sensor with Arduino At Mega 2560

Motor Driver

DC motors can be controlled by the L298N DC motor driver IC, which is connected to your microcontroller. L298Ns can control up to 2 DC motors. You can easily add motors through the program code. You can set DC motor speed by changing the duty-cycle of each PWM signal. The PWM is a square-wave signal which has two parameters: frequency and duty-cycle. If a PWM signal's duty-cycle is 100% than the motor spins with maximum rpm. In case of 0% the motor will stop. Motor speed and direction in your Ozeki software can be changed on each motor with the motor's own adjustable slide bar from -100% to 100%. If the slide crosses 0 than the motor will

change spin direction.

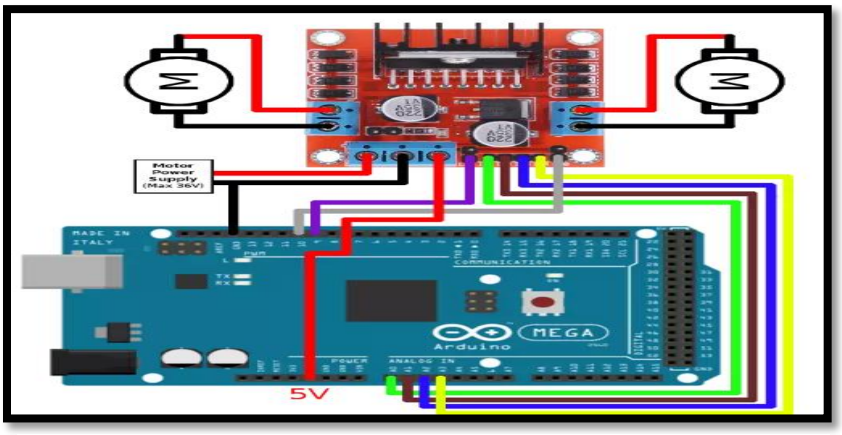


Figure 7.13 Interfacing of Motor Driver with Arduino At Mega 2560

PWM – to control speed

The speed of a DC motor can be controlled by changing its input voltage. A widely used technique to accomplish this is Pulse Width Modulation (PWM).

PWM is a technique in which the average value of the input voltage is adjusted by sending a series of ON-OFF pulses. This average voltage is proportional to the width of the pulses, which is referred to as the **Duty Cycle.**

The higher the duty cycle, the higher the average voltage applied to the DC motor, resulting in an increase in motor speed. The shorter the duty cycle, the lower the average voltage applied to the DC motor, resulting in a decrease in motor speed.

The image below shows PWM technique with various duty cycles and average voltages.

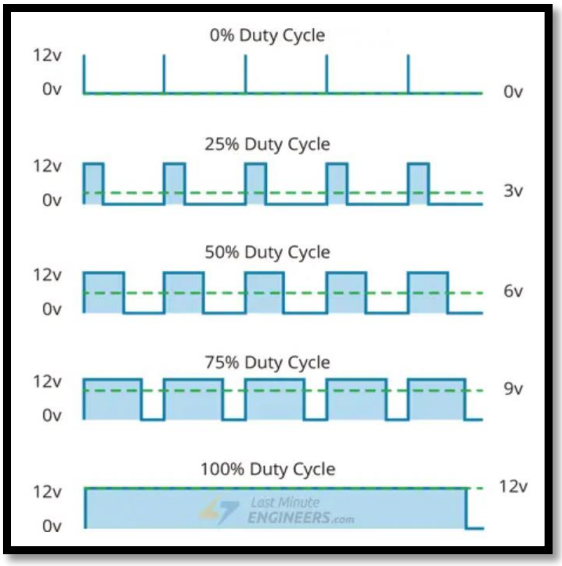


Figure 7.14 Duty Cycle for Motor Driver

H-Bridge – to control the spinning direction

The spinning direction of a DC motor can be controlled by changing the polarity of its input voltage. A widely used technique to accomplish this is to use an H-bridge.

An H-bridge circuit is made up of four switches arranged in H shape, with the motor in the center.

Closing two specific switches at the same time reverses the polarity of the voltage applied to the motor. This causes a change in the spinning direction of the motor.

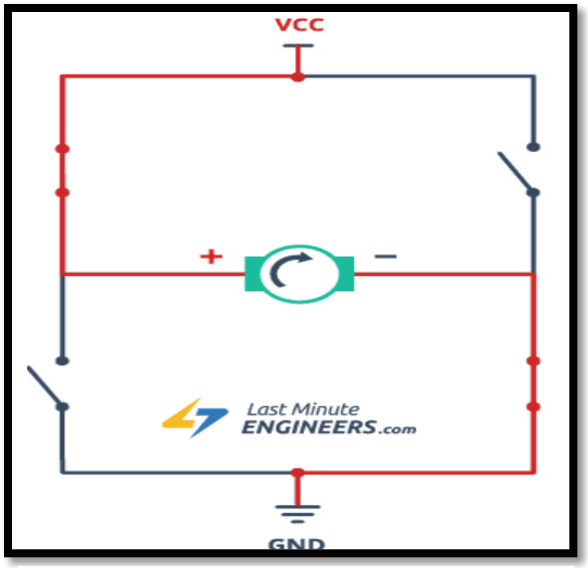


Figure 7.15 Closed Switches with Clockwise Direction Voltage

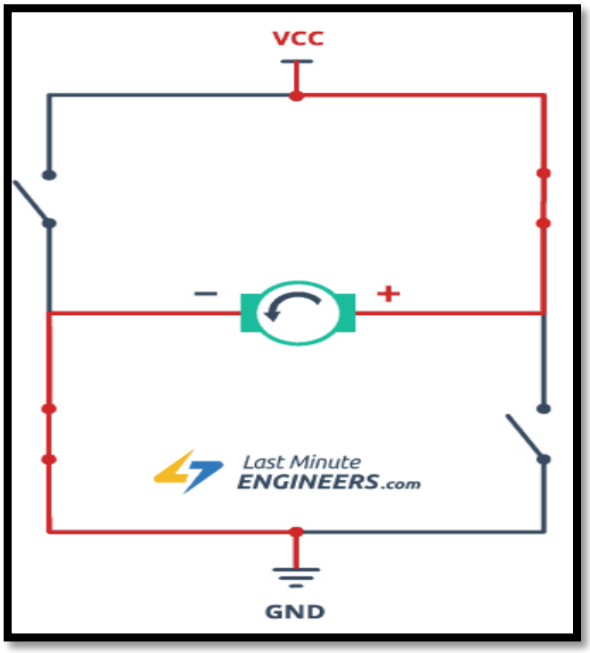


Figure 7.16 Closed Switches with Anti Clockwise Direction Voltage

L298N Motor Driver Chip

At the center of the module is a big, black chip with a chunky heat sink – the L298N. The L298N chip contains two standard H-bridges capable of driving a pair of DC motors, making it ideal for building a two-wheeled robotic platform.

The L298N motor driver has a supply range of 5V to 35V and is capable of 2A continuous current per channel, so it works very well with most of our DC motors.

Technical Specifications

Motor output voltage	5V – 35V
Motor output voltage (Recommended)	7V – 12V
Logic input voltage	5V – 7V
Continuous current per channel	2A
Max Power Dissipation	25W

L298N Motor Driver Module Pinout

The L298N module has 11 pins that allow it to communicate with the outside world. The pinout is as follows:

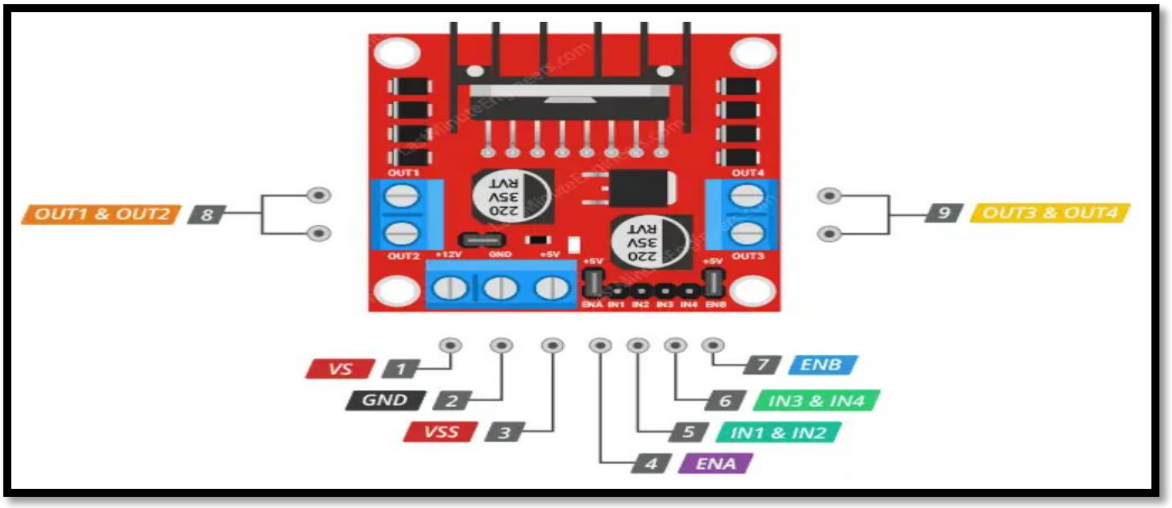


Figure 7.17 Pin Configuration of L298N Motor Driver

Power Pins

The L298N motor driver module receives power from a 3-pin, 3.5mm-pitch screw terminal.

The L298N motor driver has two input power pins: VS and VSS.

VS pin powers the IC's internal H-Bridge, which drives the motors. This pin accepts input voltages ranging from 5 to 12V.

VSS is used to power the logic circuitry within the L298N IC, and can range between 5V and 7V.

GND is the common ground pin.

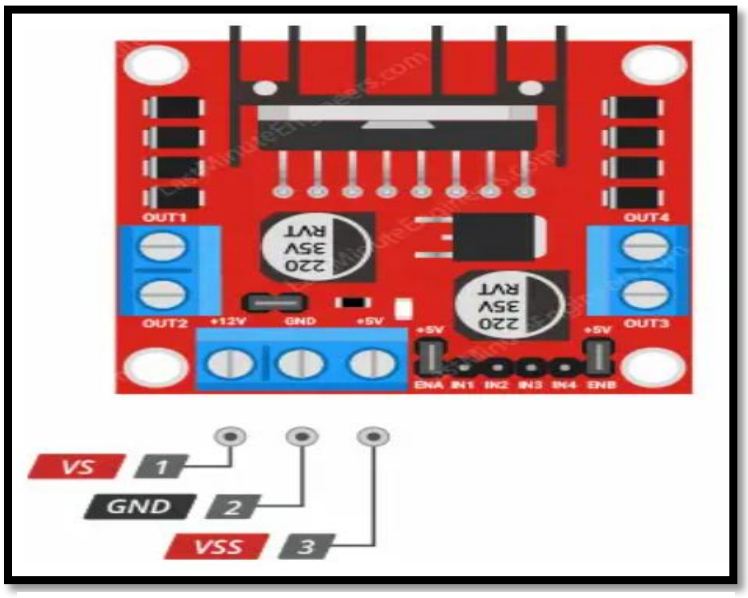


Figure 7.18 Power Pins of L298N Motor Driver

Output Pins

The output channels of the L298N motor driver, **OUT1 and OUT2** for motor A and **OUT3 and OUT4** for motor B, are broken out to the edge of the module with two 3.5mm-pitch screw terminals. You can connect two 5-12V DC motors to these terminals.

Each channel on the module can supply up to 2A to the DC motor. The amount of current supplied to the motor, however, depends on the capacity of the motor power supply.

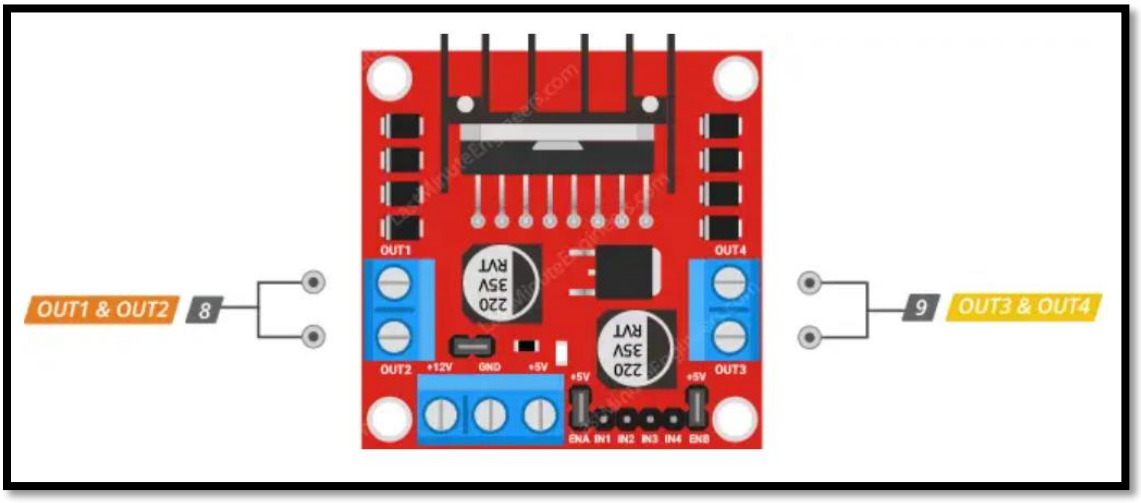


Figure 7.19 Output Pins of L298N Motor Driver

Direction Control Pins

The direction control pins allow you to control whether the motor rotates forward or backward. These pins actually control the switches of the H-Bridge circuit within the L298N chip.

The module has two direction control pins. The **IN1 and IN2** pins control the spinning direction of motor A; While **IN3 and IN4** control the spinning direction of motor B. The spinning direction of the motor can be controlled by applying logic HIGH (5V) or logic LOW (Ground) to these inputs. The chart below shows various combinations and their outcomes.

Input1	Input2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

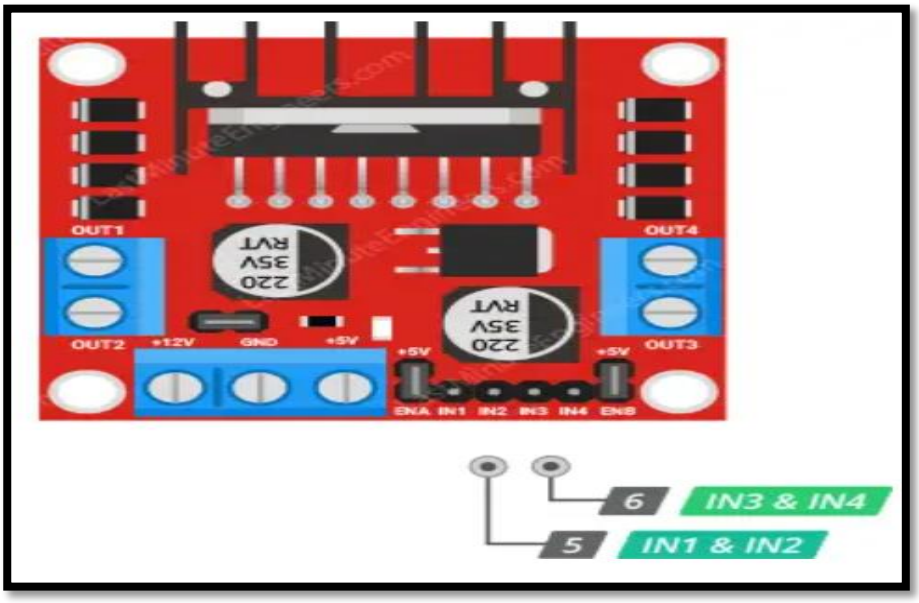


Figure 7.20 Direction Control Pins of L298N Motor Driver

Speed Control Pins

The speed control pins **ENA** and **ENB** are used to turn on/off the motors and control their speed.

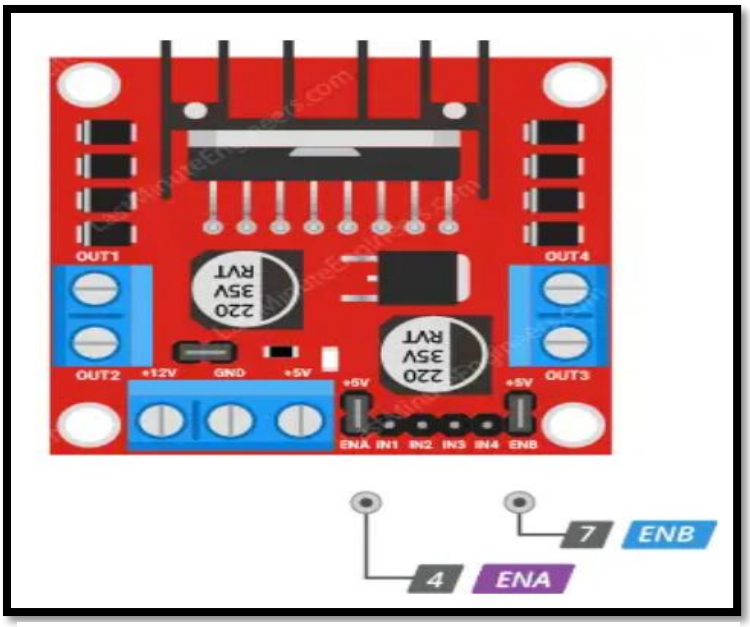


Figure 7.21 Enable Pins of Motor Driver

Pulling these pins HIGH will cause the motors to spin, while pulling them LOW will stop them. However, with Pulse Width Modulation (PWM), the speed of the motors can be controlled. Pulling these pins HIGH will cause the motors to spin, while pulling them LOW will stop them. However, with Pulse Width Modulation (PWM), the speed of the motors can be controlled.

When this jumper is in place, the 5V regulator is enabled, and the logic power supply (VSS) is derived from the motor power supply (VS). In this case, the 5V input terminal acts as the output pin, delivering 5V 0.5A. You can use it to power an Arduino or other circuitry that needs 5V power.

When the jumper is removed, the 5V regulator is disabled, and we have to supply 5V separately through the VSS pin.

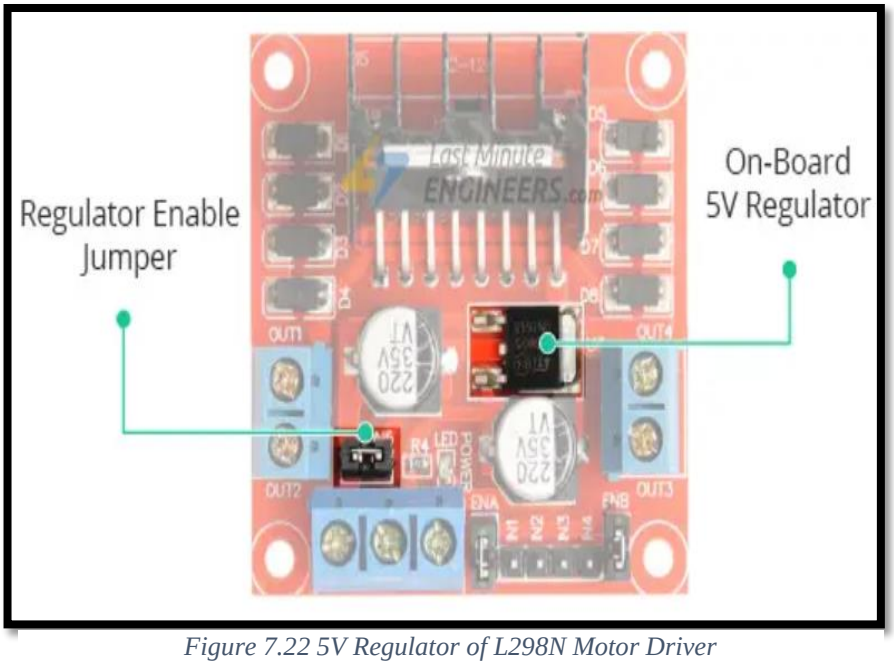


Figure 7.22 5V Regulator of L298N Motor Driver

Wiring an L298N Motor Driver Module to an Arduino

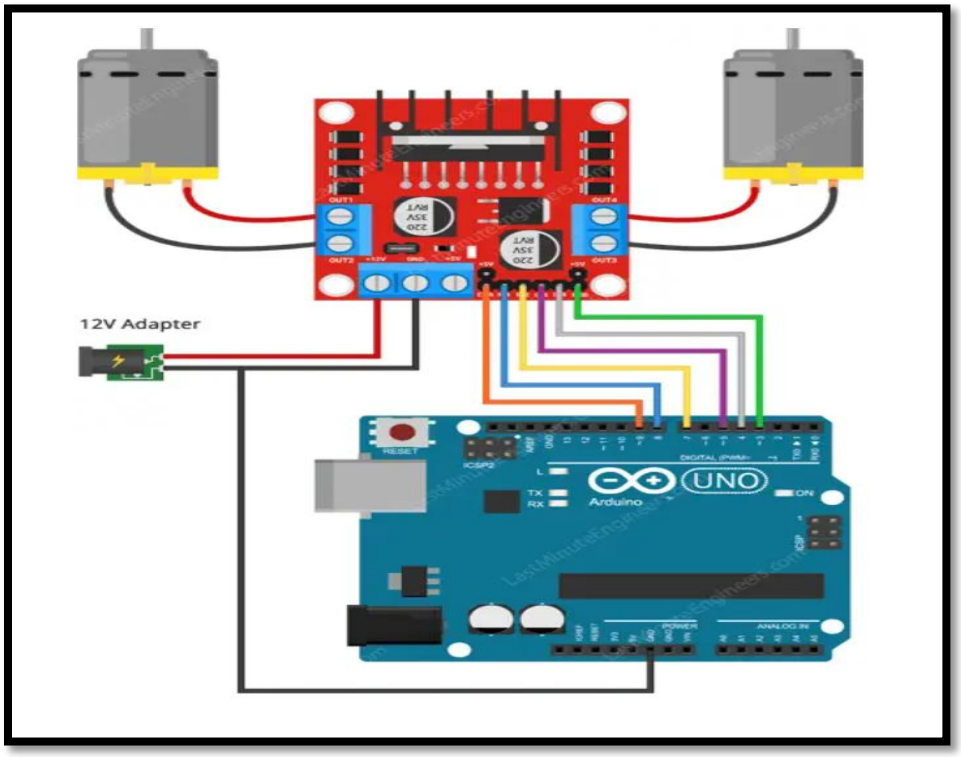


Figure 7.23 Interfacing L298N with Arduino connecting two motors

ESP8266 WiFi Module interfacing with Arduino

There are many ESP8266 WiFi modules available in marker ranging from ESP-01 to ESP-12. But in this tutorial, we are using ESP-01. AT commands are the same for all these ESP modules. The ESP8266 WIFI module consists of two rows of eight pins.

- Tx - Transmitting pin
- CH-DO – Channel Down pin
- RST – Reset
- VCC – 3.3V power supply
- GND – Power supply ground
- GPIO_2 – Not Used
- GPIO_0 – Not Used
- Rx – Receiver pin

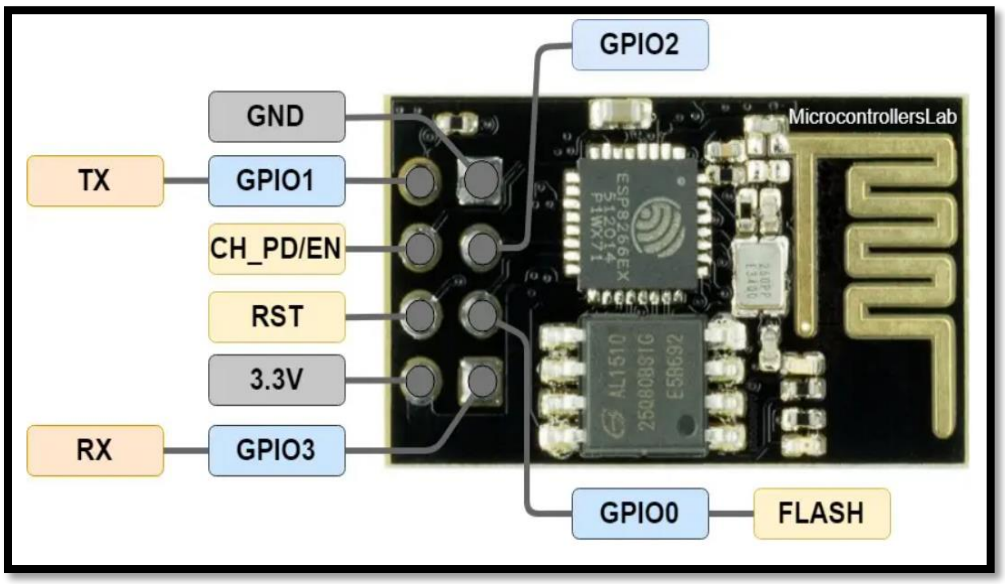


Figure 7.24 Pin Configuration of ESP8266 WIFI Module

Following is the connection diagram below to connect the two devices.

ESP-01	Arduino UNO
VCC	3.3V
EN	3.3V
GND	GND
TX	Pin 6
RX	Connect RX of ESP-01 with middle point (Junction point of series 1k and 2k resistor) of Voltage divider. The second end of 1k resistor with Pin 7 of Arduino. The second end of 2k resistor with GND pin of Arduino.

Table 7.1 Pin Configuration of WIFI Module with Arduino

The diagram below shows the connection diagram of Arduino UNO with ESP-01.

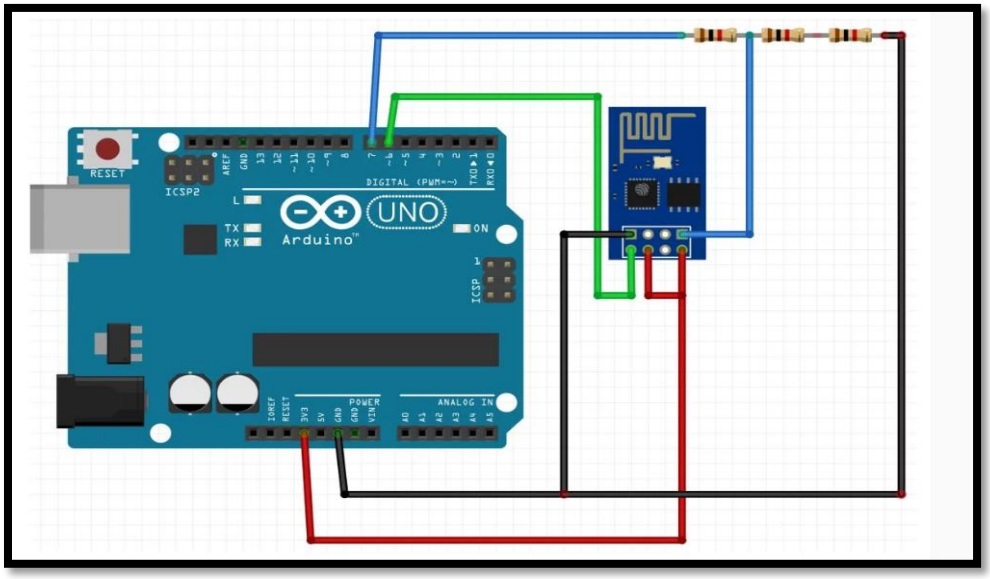


Figure 7.25 Interfacing ESP8266 WIFI Module with Arduino

LCD Interfacing with Arduino

Connecting an LCD screen was never this easy before. Connect an LCD (16*2) to Arduino MEGA 2560 without literally using any wires. You just need the two things, and you have completed this tutorial's requirement. There is no need to explain much in this tutorial here. You have to do nothing. Just take the LCD screen and connect it as follows.

Just connect VSS to A0 using the image as reference. The other pins will go in naturally.

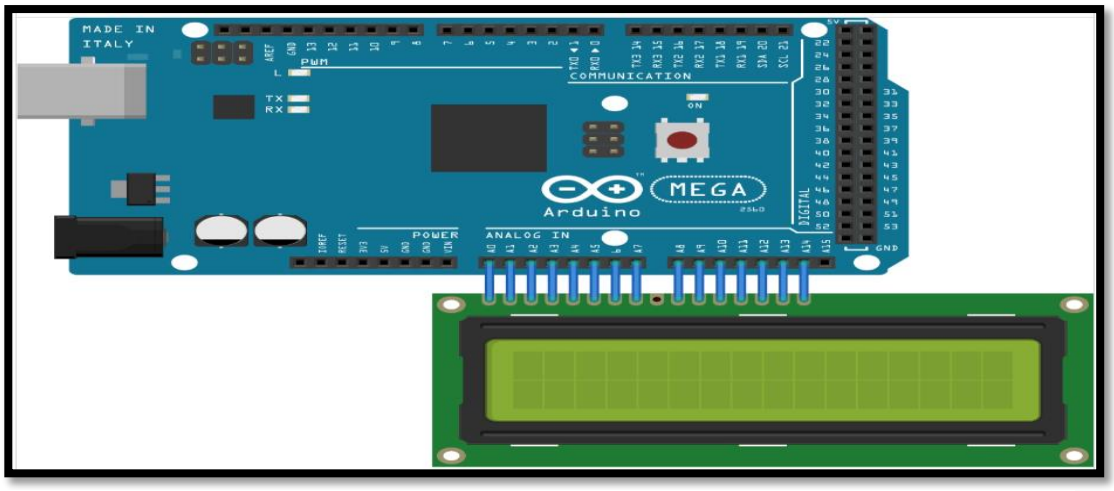


Figure 7.26 Interfacing LCD with Arduino

Interfacing SD Card with Arduino

A standard microSD card has an operating voltage of 3.3 V. As a result, we cannot connect it directly to circuits that use 5V logic; in fact, any voltages above 3.6V may permanently damage the microSD card. That is why the module includes an onboard ultra-low dropout voltage regulator capable of regulating voltage to 3.3V.

The microSD card module is simple to connect. There are six pins on it:

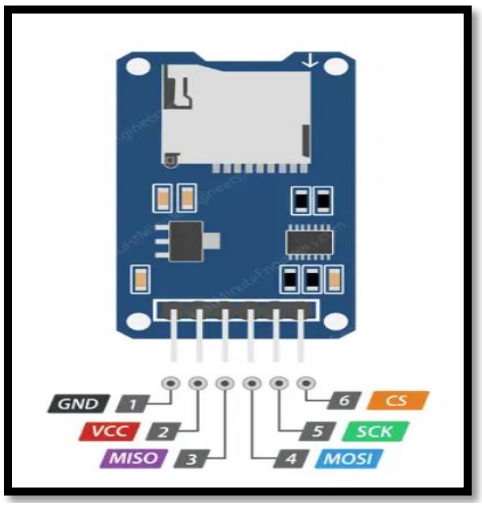


Figure 7.27 Pinout of SD Card Module

Wiring a microSD Card Module to an Arduino

Connect the module's VCC pin to 5V on the Arduino and the GND pin to ground. Now we are left with the pins that are used for SPI communication. Because microSD cards require a lot of data transfer, they perform best when connected to the microcontroller's hardware SPI pins.

Note that each Arduino board has different SPI pins that must be connected correctly. For Arduino boards such as the UNO/Nano V3.0 those pins are digital 13 (SCK), 12 (MISO), 11 (MOSI) and 10 (CS).

The following table lists the pin connections:

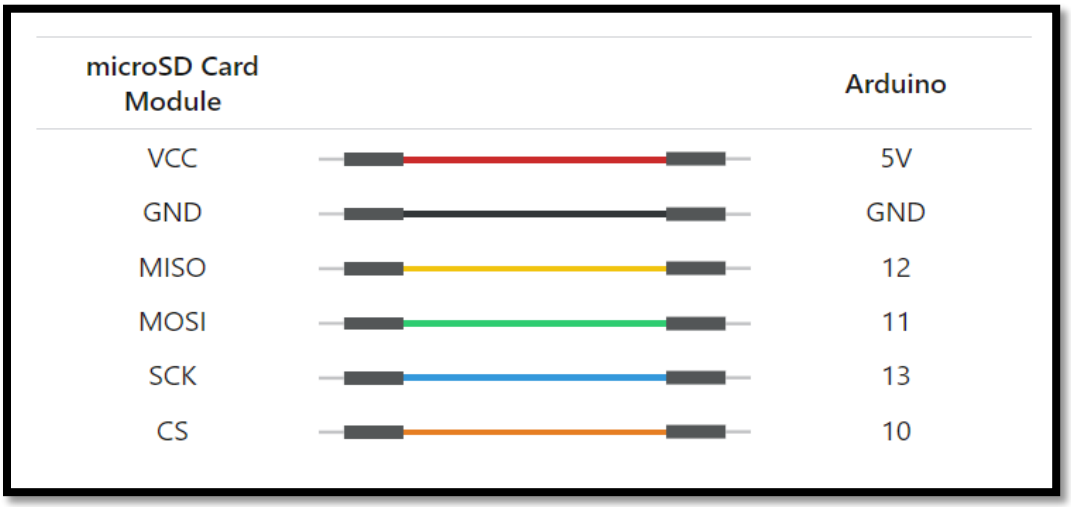


Figure 7.28 SD Card Module pins with Arduino Pins

The diagram below shows how to connect microSD Card Module to the Arduino.

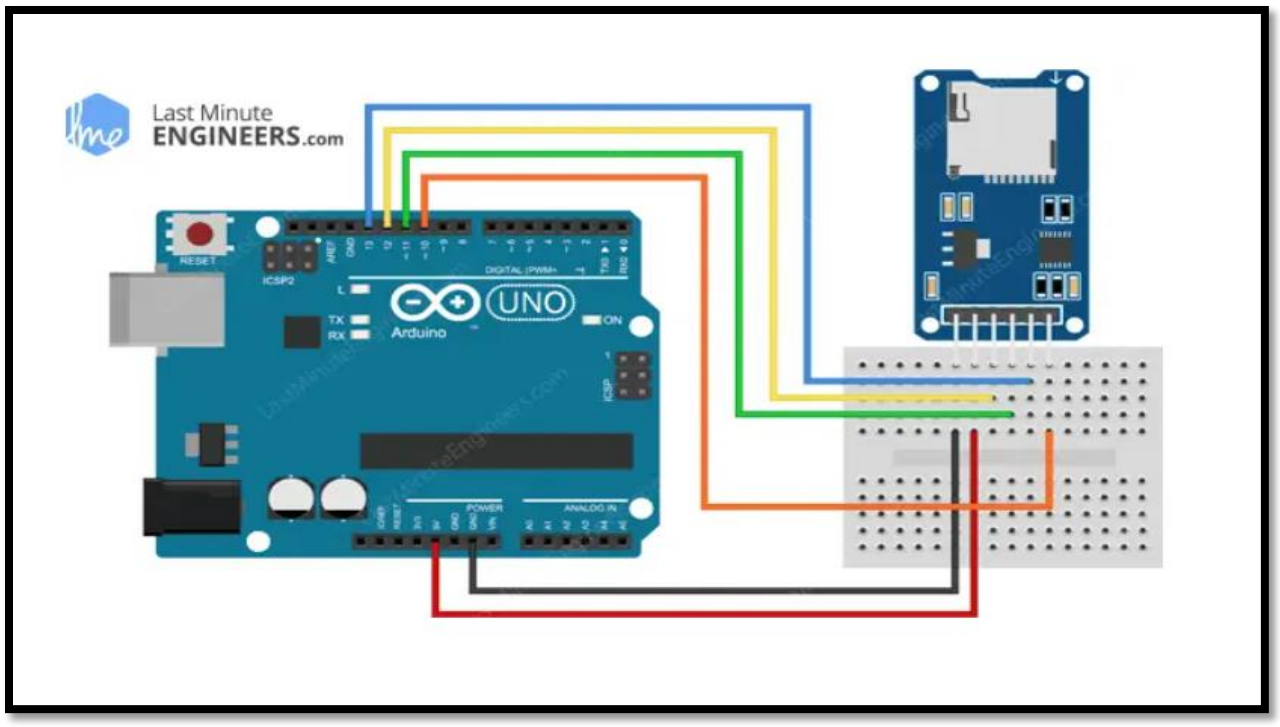


Figure 7.29 Interfacing SD Card with Arduino

7.2 PCB Implementation:

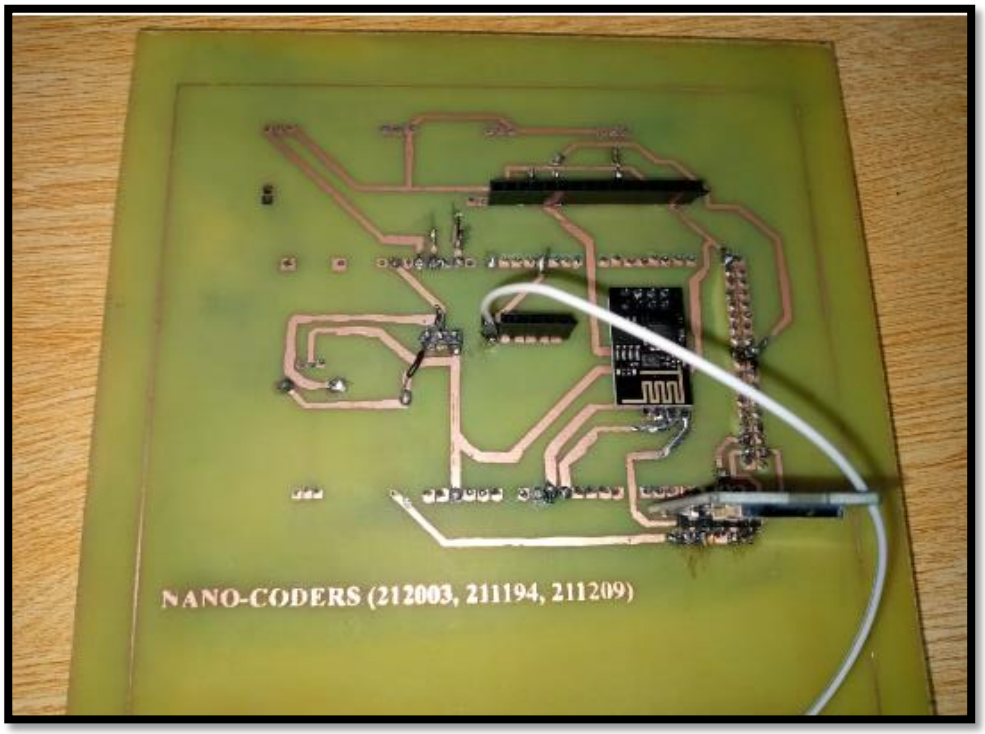


Figure 7.30 PCB Implementation

7.3 Breadboard Implementation

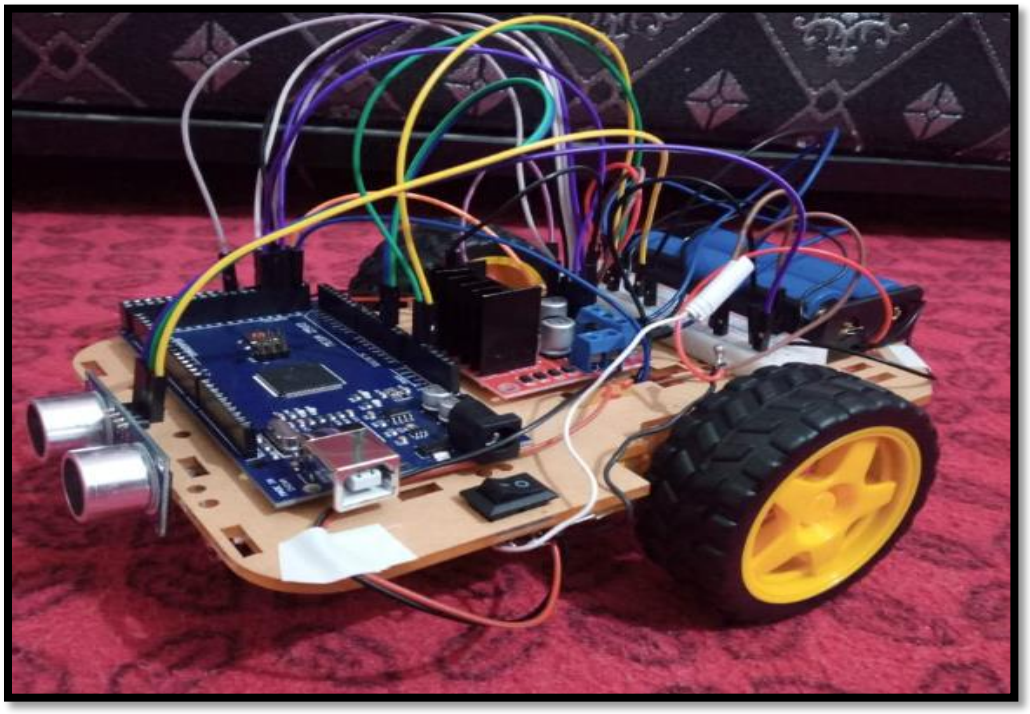


Figure 7.31 Breadboard Implementation A

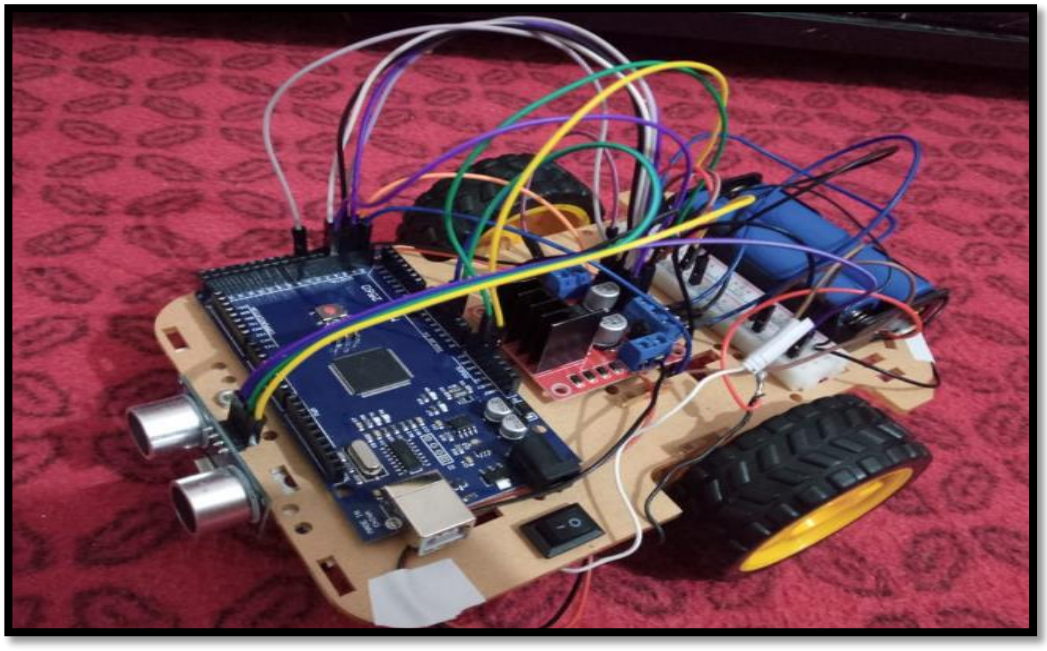


Figure 7.32 Breadboard Implementation B

7.4 State Diagram and State Table

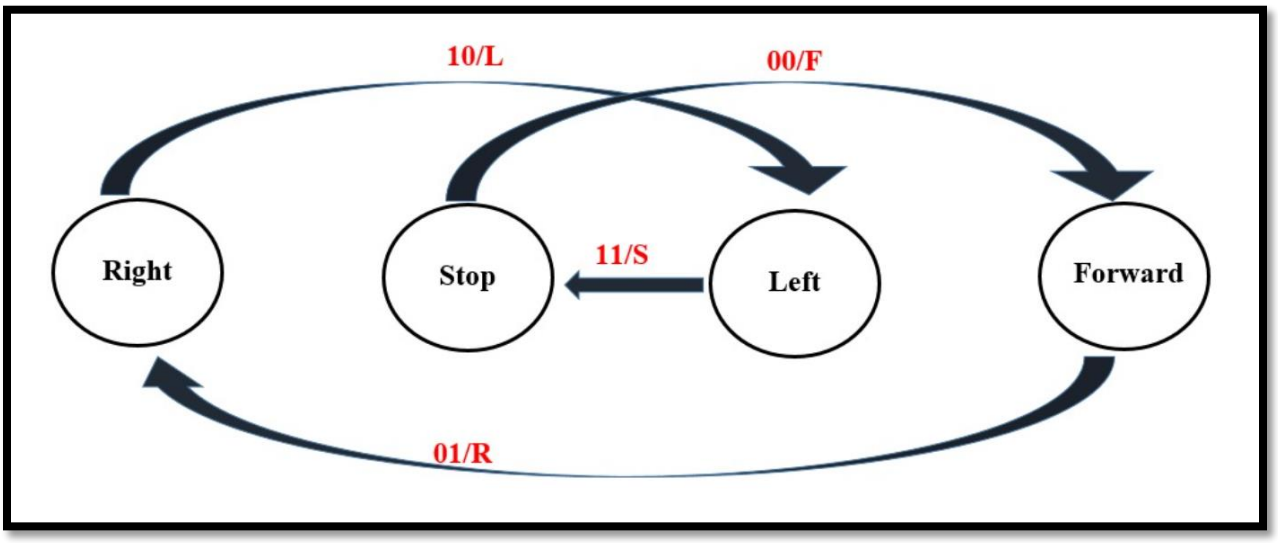


Figure 7.33 State Diagram

Current state	Left Sensor	Right Sensor	Next state	Symbols
Stop	0	0	Forward	F
Forward	0	1	Right	R
Right	1	0	Left	L
Left	1	1	Stop	S

Figure 7.34 State Table

Chapter-8 System Test Phase

8.1 Final Testing

Final testing of our project was done this arena which is shown below:

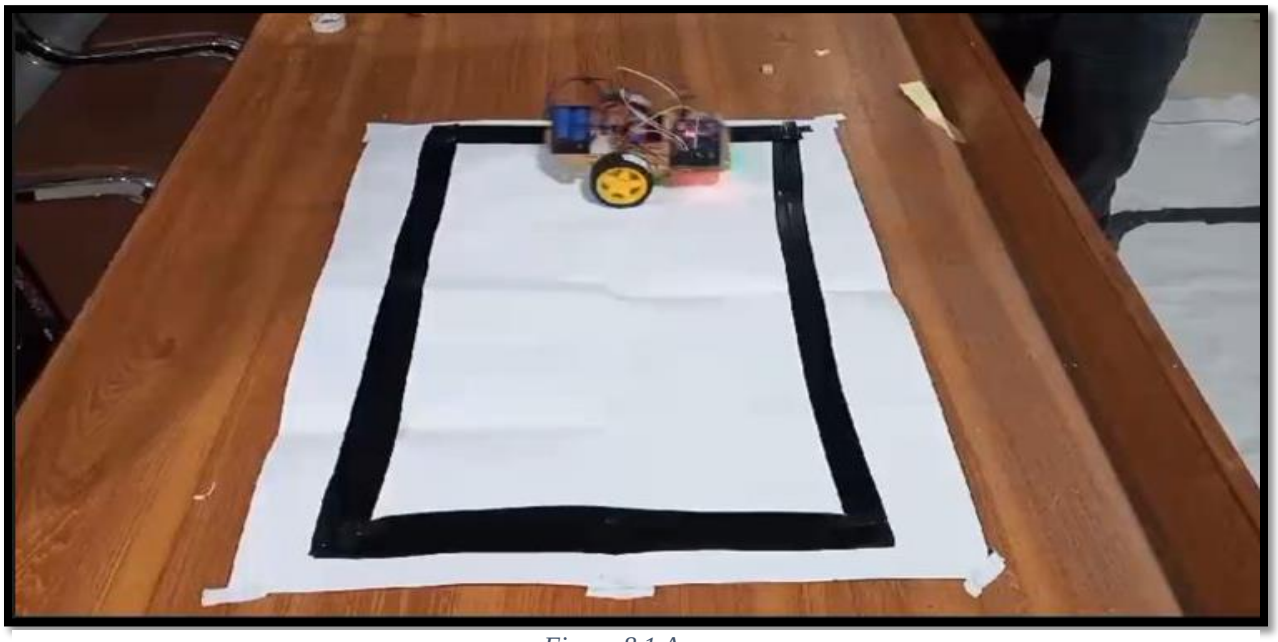


Figure 8.1 Arena

The arena path contains straight and curved path for testing we must connect our IR Sensors with 5V power supply motor driver with 15V power supply and infrared sensors with 15V power supply. After connecting all the power supplies the robot motor will start rotating. Then we have to place our robot in the start of black line. The 2 infrared sensors attached In front of robot will detect the black line and give binary output 1 to the motor driver. If the left sensor will detect black line, then it will give 1 output to motor driver. Motor driver will rotate right motor and robot will take turn towards left. But if the sensors calibration is not done properly then we have to calibrate them first according to our path.

Steps to do calibration for black and white:

1. Mount the IR sensor about 1 to 2 inches above a black surface.
2. Turn the potentiometer fully clockwise.
3. Slowly turn the potentiometer counterclockwise until the signal LED just turns off.
4. Do not change the distance between the IR sensor and the black surface. If you need to change the distance, then you need to re-calibrate.

5. The signal LED will turn on when the sensor is above a white surface and off when it is above a black surface.

   After doing proper calibration we again tested the robot, and it was following the path perfectly. The final testing was achieved successfully.

8.2 Project Actual Pictures:

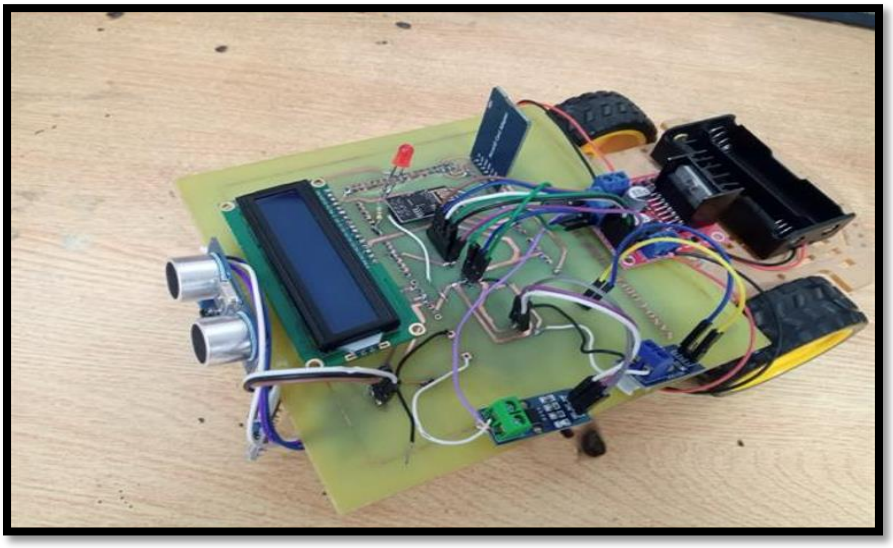


Figure 8.2 Complete Line Follower Robot A

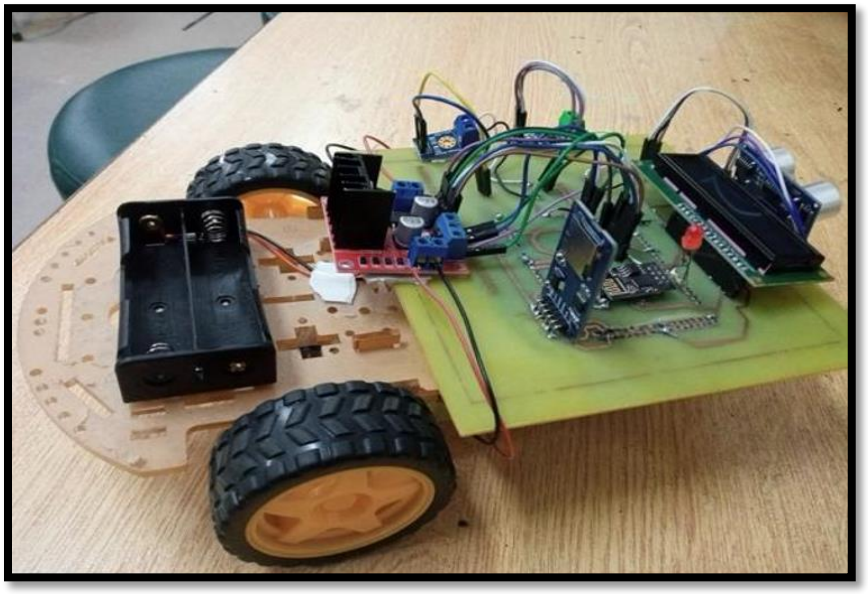


Figure 8.3Complete Line Follower Robot B

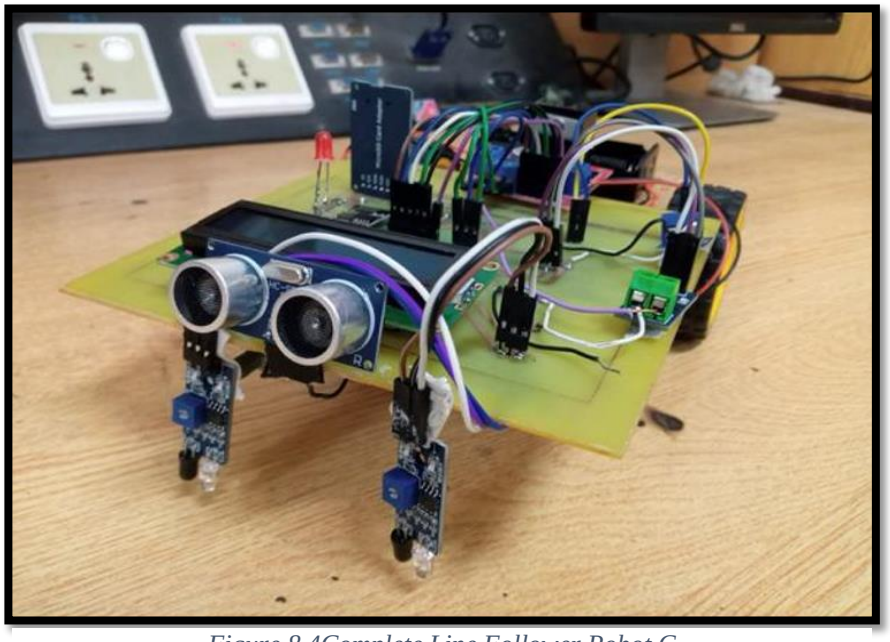


Figure 8.4Complete Line Follower Robot C

Chapter-9-Project Management

9.1 Individual Role in Project

As we have assigned to make a project based on the instructions and information and evaluation of the course in practical life, i.e., Microcontrollers and embedded systems. In this course, we have studied the very basic to the very peak and applicable procedures to solve the problem of logical circuit by creating their logical statements and codes and make them work accordingly. This is the whole background scenario on which we must make the project on the given problem that is:

Create and construct a robot which follows the line and track it along a specific path (LINE FOLLOWING ROBOT) using a microcontroller. Also use different sensors like voltage, current, speed sensors and display it on LCD display and save the data on SD card.

There are three procedures/phases of this whole project, as follows.

- Circuit designing and simulation on Proteus.
- Simulation and Proceedings of the circuit/logic on breadboard.
- PCB Designing and Finalizing procedures.

GROUP

Our group consists of three hardworking and cooperative students.

- Faizan Ahmed Siddiqui (Team Lead)
- Sania Ali (Technical Lead)
- Arooj Aftab (Project Manager)

Our group lead had assigned each task equally to everyone at every start/beginning of each phase. The role that everyone executed to fulfill his own tasks assigned by the group leader are written below. Everyone has done mixture of tasks in every phase. In ever phase, the Main support was given by the group lead but other also had done their best so that the workload was divided equally on each.

Faizan Ahmed Siddiqui:

Sania Ali:

- Interfacing sensors on breadboard
- Testing of the code on breadboard
- Patching double sided PCB
- Soldering
- Troubleshooting

Arooj Aftab:

- Detailed Report Writing
- Software interfacing module (Interfacing of IR sensor and Motor Driver with Arduino)
- Breadboard wiring

9.2 Comment on Project and Risk Management

Faizan Ahmed Siddiqui:

Project Overview:

Our 'Line Follower Robot' project has been a great learning experience throughout the semester. Tackling challenges like in sensor testing, PCB design, and wiring taught us practical skills in electronics and robotics. Working as a team highlighted the importance of collaboration, adaptability, and troubleshooting. The initial phase involved connecting sensors to a breadboard for testing, ensuring their proper functionality. Following successful tests, the sensors were

soldered onto a double-sided PCB. A meticulous check was conducted post-soldering to identify and rectify any potential short circuits that could impact the circuit's functionality.

Key Challenges Encountered:

Numerous challenges arose during the project. Fortunately, there were no instances of component short-circuits, preserving the overall circuit integrity. Calibration of the Infrared (IR) sensor posed a significant challenge due to its sensitivity to sunlight, the challenge face during making PCB and schematic designs and breadboard wiring etc.

Simulation and PCB Design:

- Designing the robot's schematics and PCB layout brought challenges, especially with the Arduino Mega's footprint missing. This meant we had to manually create the Arduino Mega's layout, a tricky task that required precision and research. During simulation, ensuring the accuracy of the manually crafted Arduino Mega footprint added another layer of complexity. I tackled the issue head-on, investing time and effort to create an accurate Arduino Mega footprint given below. Despite the initial difficulty, the final PCB design successfully integrated all components, including the Arduino Mega, ensuring a functional robot system given in the report above.

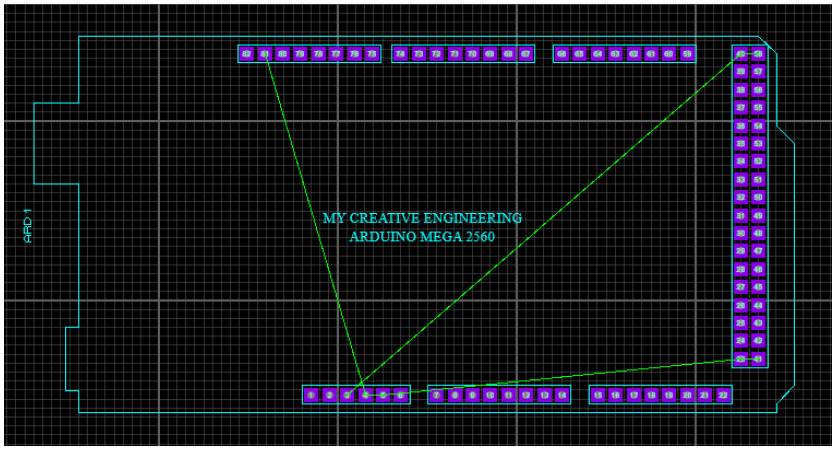


Figure 4:- Arduino MEGA 2560 missing PCB foot print.

- The another challenge with a double-sided PCB arose when connections overlapped, requiring additional wires or jumpers to bridge the gap between the top and bottom copper layers as highlighted by white circles given below. This complexity added a significant layer of difficulty to the soldering process. Managing these extra connections demanded

precision and careful planning to avoid interference and ensure the integrity of the circuit. Despite the added challenge, overcoming this hurdle contributed to a comprehensive understanding of double-sided PCB design and effective troubleshooting skills.

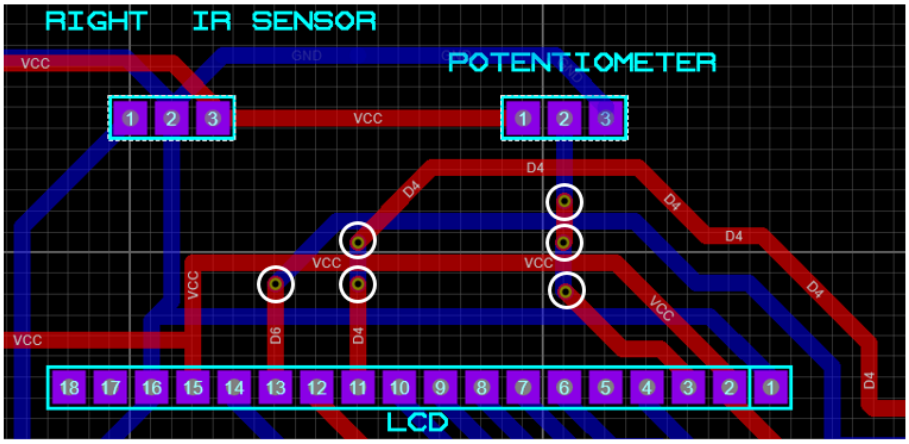


Figure 2:- The spots in routing showing the connections od top to bottom copper through wires.

- Another challenge faced during PCB design is in the routing phase due to the very small gaps between the pads of the Arduino Mega footprint. Navigating these tight spaces demanded precision to avoid short circuits. Overcoming this complexity required meticulous planning to route traces effectively.

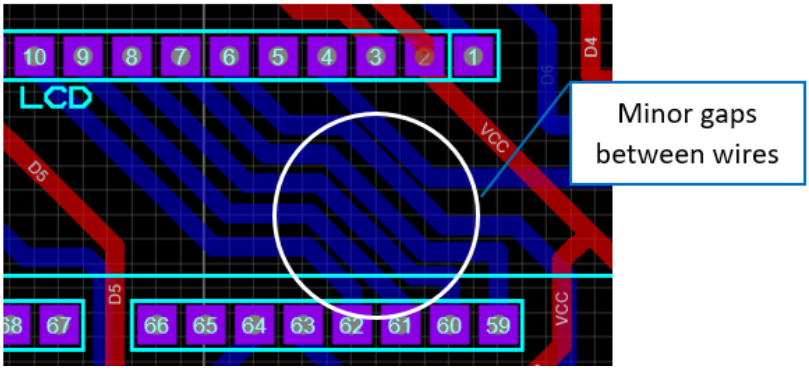


Figure 3:- Small gaps require precision to avoid short circuits *between wires.*

Wiring on Breadboard:

Wiring the robot on a breadboard posed challenges with limited space, compatibility concerns, fragile connections, and accessibility issues. Proximity-related problems included cross-talk and interference, while maneuvering within tight constraints proved challenging. Ensuring proper wire

identification for troubleshooting was crucial, and transient disruptions in temporary connections required constant attention. Overcoming these challenges demanded careful planning for a reliable robot assembly on the PCB (Printed Circuit Board).

Interfacing of IR Sensors:

The IR sensor presented a significant challenge due to its high sensitivity to sunlight, leading to unpredictable behavior. Calibrating it proved to be a demanding task, requiring meticulous adjustments to ensure accurate functionality in the presence of varying light conditions. To tackle sunlight interference with the IR sensor, we applied black tape to shield it. This straightforward solution reduced erratic behavior caused by excessive light. We also tested the robot in low-light environments to optimize the sensor's performance under different lighting conditions. These adjustments played a key role in fine-tuning the IR sensor's calibration for reliable operation.

Interfacing of SD Card Module:

Despite repeated code checks, the persistent "SD card initialization failed" message in the Arduino sketch's serial monitor remains a challenge. Resolving this issue involves revisiting the code, checking wiring, and confirming SD card module compatibility for reliable functionality. The challenge with the SD card module initially seemed like a corrupt card. We repeatedly checked and tried formatting it, confirmed connections, and even conducted beep tests. However, it took a while to realize the module itself was faulty, causing a delay in troubleshooting.

Interfacing of Ultrasonic Sensors:

The challenge with the ultrasonic sensor is occasional inconsistency in distance measurements, often influenced by obstacles or environmental factors. Troubleshooting involves adjusting sensitivity, refining the code, and considering environmental conditions for more accurate and reliable results. The sensor detects objects easily, but it tends to become overly sensitive in certain environmental conditions, leading to confusion. Instead of following the intended path during repeated rotations, it might deviate. To address this, we've been actively managing the environment and adjusting sensitivity settings to achieve better and more consistent results.

Coding:

Coding the entire robot, incorporating various modules like IR sensors, ultrasonic modules, current and voltage sensors, SD card modules, and the L298N driver, was a complex task. It involved

understanding the nuances of each module which I studied separately one by one mentioned above in the report. This is all for Resolving conflicts, optimizing data interpretation, and synchronizing operations. The challenge was to create efficient code capable of handling real-time inputs from multiple sensors for a responsive and reliable robot system. Coordination among diverse modules was crucial for achieving a cohesive and high-performing robotic functionality.

Integration of Code:

Merging code from different modules into a single script was a big challenge. Coordinating diverse components like IR sensors and motor drivers required careful synchronization. Overcoming conflicts and optimizing performance were key integration hurdles, emphasizing the need for a balanced and efficient codebase for the entire robot.

Voltage and Current Sensors Code:

The challenge in voltage and current sensor codes is to precisely measure and interpret electrical parameters. Achieving accurate readings and seamless integration into the system demands careful code calibration and troubleshooting to ensure reliable data for the robot's operation. At first, we struggled to get the right output on the Arduino-connected LCD, even after troubleshooting a lot. After many tries, I figured out the mistake, that the reference voltage was incorrect. Fixing that issue sorted out the problem.

Trouble Shooting:

The main challenge is figuring out why the robot isn't sticking to the line as it should. Troubleshooting involves checking if the sensors are working right, making sure the robot understands the signals it gets, and ensuring the motors respond correctly. Sometimes, the robot faces problems like loose connections that stop the motors, environmental changes making sensors too sensitive, and occasional mistakes from the ultrasonic sensor. Fixing these issues involves securing connections, adjusting sensor sensitivity, and refining the ultrasonic sensor for smooth robot operation in different situations.

Feedback Mechanisms:

To provide system feedback, an LCD was integrated into the circuit. However, adjusting the brightness using a potentiometer proved challenging. Subsequently, an LED was incorporated to

serve as an indicator for voltage checks. Adding an LED to indicate low battery and coding it to blink when voltage drops below 6V faced a challenge. Despite extensive troubleshooting, the LED didn't blink, and robot interruption didn't occur as expected. Through persistent efforts, it was discovered using a digital multi-meter that the complete voltage wasn't reaching the end of the LED circuit, indicating a shortfall in meeting the LED voltage requirement. This realization led to a solution, emphasizing the importance of ensuring the full voltage supply for the proper functioning of the LED indicator.

Circuit Complexity:

The circuit's complexity led to initial oversights in connections during the soldering process. Identified and trouble shooted post-implementation, these issues were rectified to ensure the circuit operated as intended. The addition of an LED for voltage checking enhanced the system's reliability.

Overall Impact:

The 'Line Follower Robot' project not only demonstrates technical excellence but also underscores effective collaboration and adaptability. The team's commitment to overcoming challenges makes this project a notable accomplishment.

Sania Ali:

Sensor Connection on Breadboard:

The first step involved connecting each sensor to a breadboard. This was essential for testing the sensors and making sure they functioned correctly.

Patching Double-Sided PCB:

After testing, the next step was to solder the sensors onto a double-sided Printed Circuit Board (PCB). Following the soldering, a thorough check was performed to identify any short circuits in the wires to ensure the proper functionality of the circuit.

Challenges Faced:

Throughout the process, various challenges were encountered. However, it was fortunate that no components were short-circuited, maintaining the integrity of the circuit.

Calibration of IR Sensor:

One of the significant challenges involved calibrating the Infrared (IR) sensor. It exhibited erratic behavior in the presence of sunlight due to its sensitivity. This required special attention to ensure the sensor worked accurately under all conditions.

LCD for Feedback:

To provide feedback on the system, an LCD was incorporated into the circuit. However, adjusting the brightness using a potentiometer proved to be challenging.

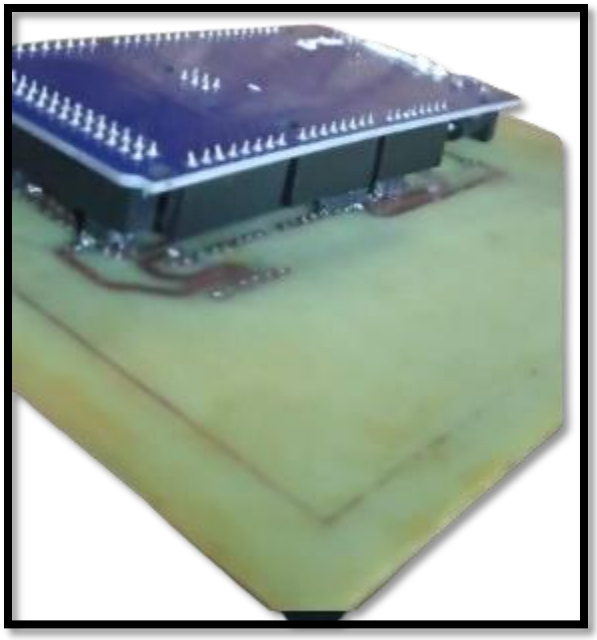
LED for Voltage Check:

To address the difficulty in controlling brightness, an LED was added. This served as an indicator to check the voltage. If the voltage dropped below 6V, the LED would glow, indicating a potential issue; otherwise, it remained off.

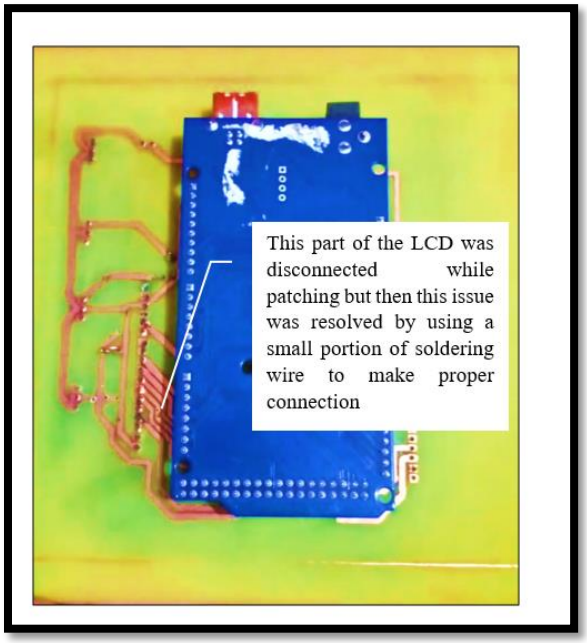
Circuit Complexity and Missed Connections:

Due to the complexity of the circuit, some connections were initially overlooked during the soldering process. However, these issues were identified and troubleshooted afterward to ensure the circuit operated as intended.

In summary, the process involved testing sensors on a breadboard, soldering them onto a double-sided PCB, addressing challenges such as sunlight interference with the IR sensor, incorporating an LCD for feedback, and overcoming issues with circuit complexity and missed connections. The addition of an LED for voltage checking enhanced the reliability of the overall system.



Above figure shows the successful patching of the Arduino with zero alignment issues in first attempt. The Mega2560 was patched with the help of male to male headers.



Arooj Aftab:

This project was a big challenge for us. We have learned many things minor and major from this project. We have implemented our project on breadboard and PCB as well. On PCB we have faced a lot of challenges. But with trouble shooting we have found the problems faced in our project.

Also we have made the double layered PCB which was very big challenge for us. But with careful attempts we have successfully made our project.

9.3 Group Members Review

Faizan Ahmed

Sania Ali

Sania Ali played a crucial role in our project, demonstrating exceptional hard work, talent, and sharp-mindedness. Her effectiveness in completing tasks, particularly in hardware-related aspects like troubleshooting and soldering, significantly contributed to our project's success. Sania consistently delivered her work on time, showcasing reliability and dedication. Her cooperation in adopting and achieving project goals was instrumental, especially in solving numerous problems we encountered. The challenge of working with a double-sided PCB, a task we tackled for the first time, was navigated skillfully by Sania. Her adept soldering skills proved crucial in handling complex situations, especially in tight spaces between wires. Overall, Sania's contribution played a key role in overcoming challenges and achieving our project goals.

Arooj Aftab

Arooj Aftab has shown proficiency in her role, particularly in detailed report writing and setting up GitHub accounts for our project. While her work is commendable, there's room for improvement in terms of meeting deadlines. Arooj excels in providing comprehensive and well-documented reports that align with the teacher's requirements. However, there's potential for even greater contribution if she shows more interest and involvement in both hardware and software aspects of the project. Encouraging her to explore these areas further could enhance her overall impact on our collaborative efforts. I suggests that enhanced project management and regular progress updates could have further improved collaboration.

Sania Ali

The work distribution was done equally by the leader but the project management could have been done better if the progress have been shown throughout the project.

Arooj Aftab

Both of my group members were best person. They worked very hard in this project. Also they have faced every challenges and problems in the project and solved them in a very excellent way.

Faizan Ahmed and Sania Ali both are best software developer and excellent in hardware implementation. They both are best researcher not only for this project but overall in every subject. Also they helped me a lot for learning many things from this project.

Overall, they did a great job in our project i.e. **'Line Follower Robot'**.

9.4 Final Bill of Materials

Components	Price- (Rs)	Quantity
ATMEGA2560	4000	1
Wires (Male to female, male to male, female to female)	650	-
Sticker Sheet	50	5
Ultrasonic Sensor	300	1
glue	120	1
Voltage Sensor	200	1
Current Sensor	200	1
Double sided tape	150	1
Ferric chloride	250	200g
IR sensors	240	2
L298N motor driver	370	1
Robotic chassis (2 tires, 2 motors, bolts, nuts, and base)	1300	1

PCB	700	1
Bread board	250	1
18650 battery cells	350	1(Pair)
LCD Display	1200	1
SD Card module	250	1
Power bank module	650	1
Total	11230	

Table 9.1 Final Bill of Material

9.5 Word Count

Group Members	Word Count for Each Member
Faizan Ahmed Siddiqui (212003)	7134
Sania Ali (211194)	6032
Arooj Aftab (211209)	10159

Chapter-10 Feedback for Project and Course

10.1Feedback for Project

The project taught us a lot. Working with circuits isn't easy, even though it might seem simple at first. Through practice and understanding how to troubleshoot problems, it became more manageable. Overall, the experience was a great learning opportunity, showing that dealing with circuits requires hands-on practice and a good grasp of troubleshooting techniques.

10.2Feedback for Course

This course has provided us in-depth knowledge of the microcontrollers and how different sensor interface with them in real world. This subject has wider applications in the real world problems.

Conclusion

In conclusion, the line follower robot project proved to be highly successful both in terms of its functionality and the valuable learning experiences it provided. Throughout the course of the project, we encountered numerous challenges, with troubleshooting emerging as a central task. One significant hurdle was the sensitivity of the Infrared (IR) sensors to white light, which required meticulous problem-solving and adjustments in the design. As design engineers, this project afforded us a deep and comprehensive understanding of sensor technologies and their integration with Arduino, a crucial aspect of modern robotics. The process not only enhanced our technical skills but also sharpened our ability to analyze and overcome unexpected obstacles. This hands-on experience in problem-solving and the intricacies of sensor calibration has undoubtedly contributed to our growth as engineers, providing valuable insights that extend beyond the immediate scope of the project. Overall, the project was a triumph in terms of both its educational value and the successful development of a functional line follower robot.

Chapter-11 GitHub

GitHub Link

Appendix

Data sheets and similar Products that we inspired from are given below in following website:

https://lastminuteengineers.com

References

[1]https://www.seeedstudio.com/blog/2019/11/13/atmega2560-features-comparisons-and-arduino-mega-review/

[2]https://store.arduino.cc/products/arduino-mega-2560-rev3

[3]https://www.robotpark.com/SD-Card-Module#:~:text=SD%20Card%20Module%20is%20a,need%20to%20use%20an%20adapter.

[4]https://components101.com/modules/micro-sd-card-module-pinout-features-datasheet-

alternatives

[5]https://www.electroduino.com/introduction-to-l298n-motor-driver-how-its-work/

[6]https://components101.com/modules/l293n-motor-driver-modulehttps://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/#:~:text=The%20module%20has%20two%20direction,(Ground)%20to%20these%20inputs.

[6]https://www.fenixlighting.com/blogs/news/the-ultimate-guide-to-the-18650-battery#:~:text=An%2018650%20battery%20is%20a,mili%2Damp%2Dhours).

[7]https://robocraze.com/blogs/post/what-is-ultrasonic-sensor

[8]https://www.fierceelectronics.com/sensors/sensors-sports-gear-someday-could-include-voice-tech

[9]https://fiixsoftware.com/glossary/what-is-a-voltage-sensor/#:~:text=A%20voltage%20sensor%20is%20a,industrial%20controls%20and%20power%20systems.

[10]https://components101.com/sensors/voltage-sensor-module

[11]https://fiixsoftware.com/glossary/what-is-a-voltage-sensor/#:~:text=It%20induces%20currents%20in%20nearby,%2Ddigital%20converters%20(ADCs).

[12]https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/

[13]https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

[14]https://www.majju.pk/product/esp-01-esp8266-serial-wifi-transceiver-module-esp-01-4mb-flash/

[15]https://lastminuteengineers.com/voltage-sensor-arduino-tutorial/#:~:text=Connecting%20a%20voltage%20sensor%20to,shows%20how%20to%20connect%20everything.

[16]https://microcontrollerslab.com/acs712-current-sensor-interfacing-arduino/

[17]https://circuitdigest.com/microcontroller-projects/interfacing-micro-sd-card-module-with-arduino

[18]https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/