Arduino Manual

Prepared by: Maedin Seid and Betemariam Assaminew

About this book

This book is intended for people who are new to microcontrollers as well as those who have some knowledge of microcontroller programming and electronics and want to expand their knowledge. This manual focuses on the Arduino Uno, a board based on the Atmega 32 microcontroller. We chose Arduino Uno over other boards because it is easily accessible; you can find it in any electronics store.

The book is divided into three sections: the first is an introduction to Arduino, followed by the components section, which includes sensors, modules, and motors that will give you a general idea of components that you will use in the final section. The final section is the project section, where you will complete projects that are categorized by difficulty, from beginner to advanced.

-Keep learning

Table of Content

1.Arduino Engineering Basics

1.1. Exploring the Arduino Ecosystem

What is an Arduino ?

Arduino is a free and open-source platform for creating electronic projects. Arduino is made up of a physical programmable circuit board (also known as a microcontroller) and a piece of software called an IDE (Integrated Development Environment) that runs on your computer and is used to write and upload computer code to the physical board.

Why Arduino Boards ?

The Arduino board has been used in a variety of engineering projects and applications. The Arduino software is easy to use for beginners while remaining flexible enough for advanced users. It is compatible with Windows, Linux, and Mac. Teachers and students in schools use it to create low-cost scientific instruments to test physics and chemistry principles.

Arduino also makes the working process of microcontroller, but it gives some advantages over other systems for teachers, students, and beginners.

- Inexpensive
- Cross-platform
- The simple, clear programming environment
- Open source and extensible software
- Open source and extensible hardware

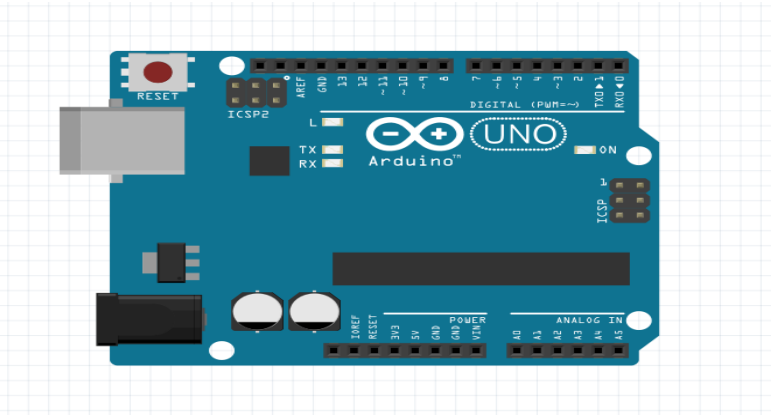
There are different Types of Arduino Boards

- Arduino Uno (R3)
- Arduino Nano
- Arduino Micro
- Arduino Due
- LilyPad Arduino Board
- Arduino Bluetooth
- Arduino Diecimila
- RedBoard Arduino Board
- Arduino Mega (R3) Board
- Arduino Leonardo Board
- Arduino Robot
- Arduino Esplora
- Arduino Pro Mic
- Arduino Ethernet
- Arduino Zero
- Fastest Arduino Board

In this project we mainly see Arduino UNO (R3) Controllers .

Uno Arduino (R3) The Uno is a fantastic choice for your first Arduino. This Arduino board is powered by an ATmega328P microcontroller. When compared to other types of arduino boards, such as the Arduino Mega, it is very simple to use. It has 14 digital I/O pins, six of which can be used as PWM (pulse width modulation outputs), six analog inputs, a reset button, a power jack, a USB connection, an In-Circuit Serial Programming header (ICSP), and so on. It includes everything needed to support the microcontroller; simply connect it to a PC via a USB cable to power it up .

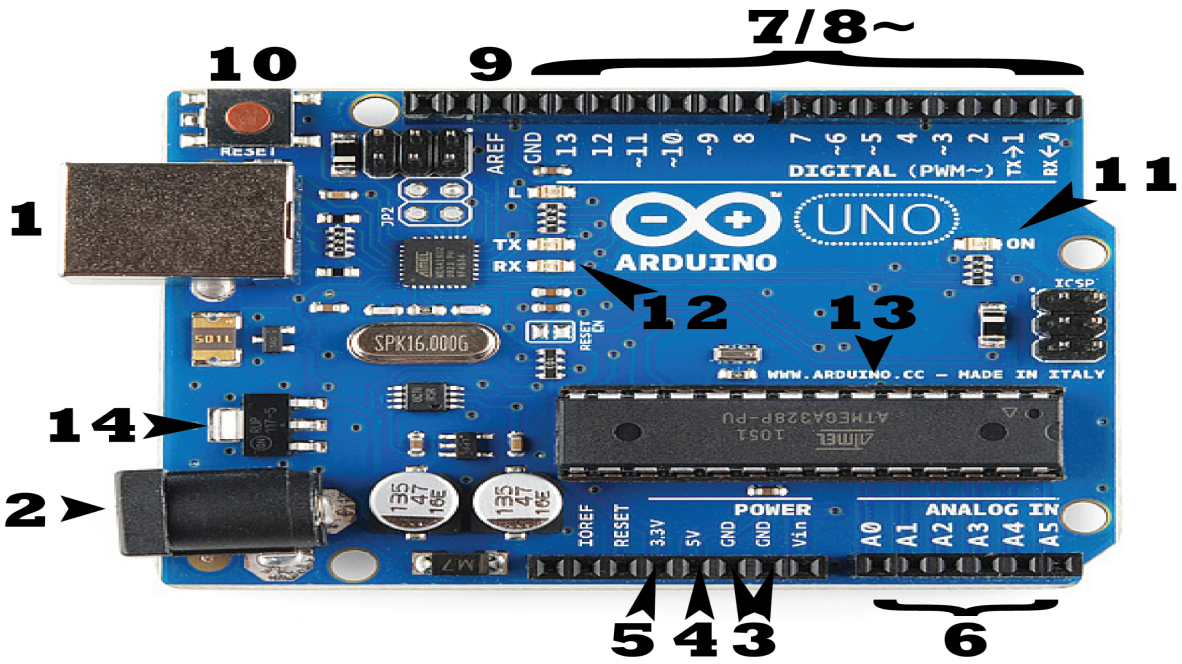
Fig 1.1.



What does it Do? The Arduino hardware and software were developed for artists, designers, hobbyists, hackers, newcomers, and anyone else interested in creating interactive objects or environments. Buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and even your phone or TV can communicate with Arduino! This adaptability, combined with the fact that the Arduino software is free, the hardware boards are reasonably priced, and both the software and hardware are simple to use, has resulted in a large user community that has contributed code and released instructions for a wide range of Arduino-based projects.

1.2.What is on board?

Fig 1.2



1 . Power USB

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer .The USB connection is also how you will load code onto your Arduino board. More on how to program with Arduino can be found in our creating your first program on the Arduino Section.

2. Power Barrel Jack

Specifically made for powering an arduino using an external source .

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

3. Ground(GND)

There are several GND pins on the Arduino, any of which can be used to ground your circuit.

4. 5v

The pin supplies 5 volts of power.

5. 3.3v

The pin supplies 3.3 volts of power.

6 .Analog

The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor and convert it into a digital value that we can read.

7.Digital

Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

8.PMW

You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). About **PMW**, you can find more about it **Basic electronics** , but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).

9.AREF

Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

10 Reset Button

Pushing this will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times.

11.Power LED Indicator

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON' . This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

12 .Tx Rx LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for Serial Communication. In our case, there are two places on the Arduino UNO where TX and RX appear -- once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs . These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

13. Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit . Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

14.Voltage Regulator

The voltage regulator is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says -- it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

1.3. Creating Your First program on Arduino

How to Install Arduino IDE ?

Go to the Arduino website at arduino.cc and click the Software tab to display the Software page shown below. From there, you can download the newest version of the IDE that corresponds to your operating system.

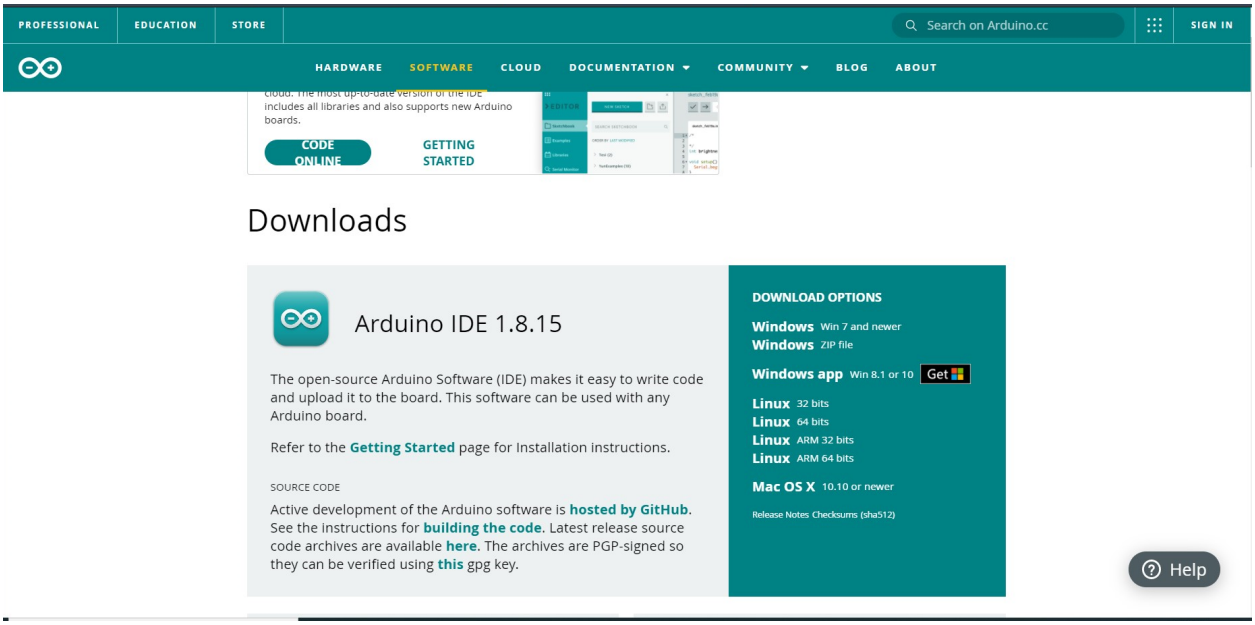


Fig 1.3

if you're on Windows, download the installer instead of the Zip file. The installer will handle loading the necessary drivers for you. Run the installer and follow the onscreen directions. All the default options should be fine. For macOS or Linux, download the compressed folder and extract it. On Mac OS X, simply drag the application into your Applications fold

How to Connect the Arduino to your Computer and Run your First code ?

Now that you have the IDE downloaded and ready to run, you can connect the Arduino to your computer via USB, as shown in below .

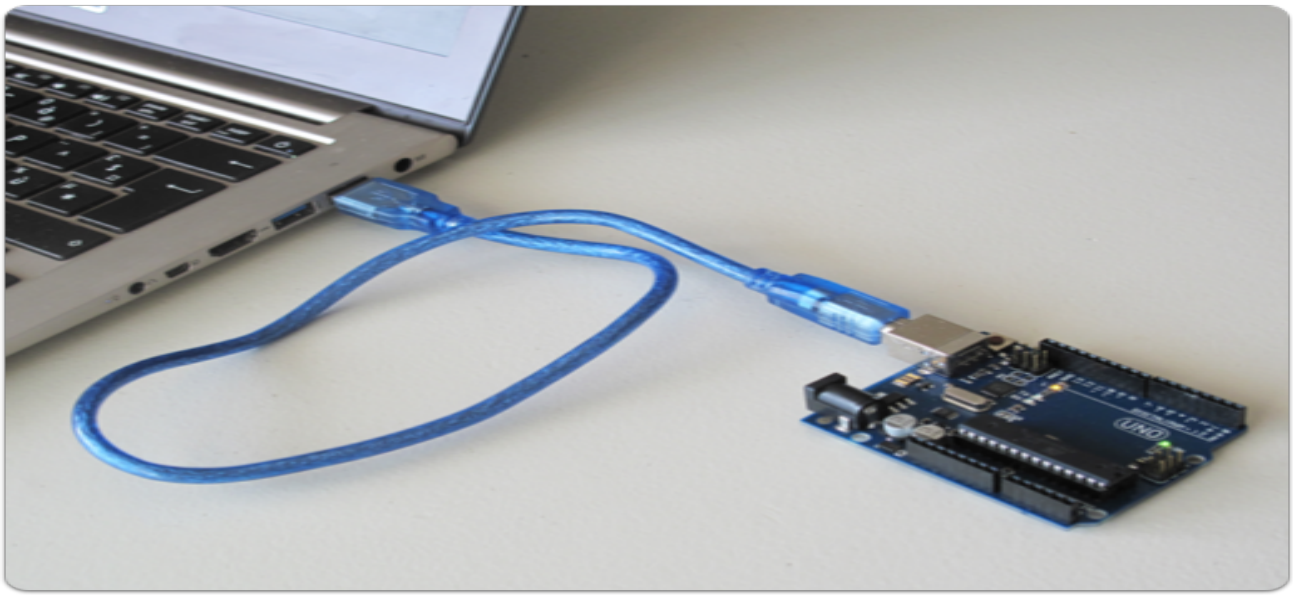


Fig 1.4

the next step is installing the board drivers

If you use the Installer, Windows - from XP up to 10 - will install drivers automatically as soon as you connect your board.

If you downloaded and expanded the Zip package or, for some reason, the board wasn't properly recognized, please follow the procedure below.

- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)". If there is no COM & LPT section, look under "Other Devices" for "Unknown Device".
- Right click on the "Arduino UNO (COmxx)" port and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.

- Finally, navigate to and select the driver file named "arduino.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named "Arduino UNO.inf"
- Windows will finish up the driver installation from there.

NOTE Having trouble getting the IDE installed, or connecting to your board? Arduino.cc provides great troubleshooting instructions for all operating systems and Arduino hardware.

Now, launch the Arduino IDE. You're ready to load your first program onto your Arduino. To ensure that everything is working as expected, you'll load the Blink example program, which will blink the onboard LED. Most Arduinos have an onboard LED **Figure 1.5**: Arduino Uno connected to a computer via USB (connected to pin 13 in the case of the Arduino Uno). Navigate to File ➢ Examples ➢ Basic, and click the Blink program. This opens a new IDE window with the Blink program already written for you. First, you'll program the Arduino with this example sketch, and then you'll analyze the program to understand the important components so that you can start to write your own programs in the next chapter. Before you send the program to your Arduino board, you need to tell the IDE what kind of Arduino you have connected and what port it is connected to. Go to Tools ➢ Board and ensure that the right board is selected. This example uses the Uno, but if you are using a different board, select that one (assuming that it also has an onboard LED—most do). The last step before programming is to tell the IDE what port your board is connected to. Navigate to Tools ➢ Serial Port and select the appropriate port. On Windows machines, this will be COM* , where * is some number representing the serial port number.

You're finally ready to load your first program. Click the Upload button (==>) in the top-left corner of the IDE. The status bar at the bottom of the IDE shows a progress bar as it compiles and uploads your program. The TX/RX LEDs on your Arduino will flash as it is programming. These LEDs show that data is being transferred to the board from your computer. When the upload completes, the onboard LED on your Arduino should be blinking once per second. Congratulations! You've just uploaded your first Arduino program.

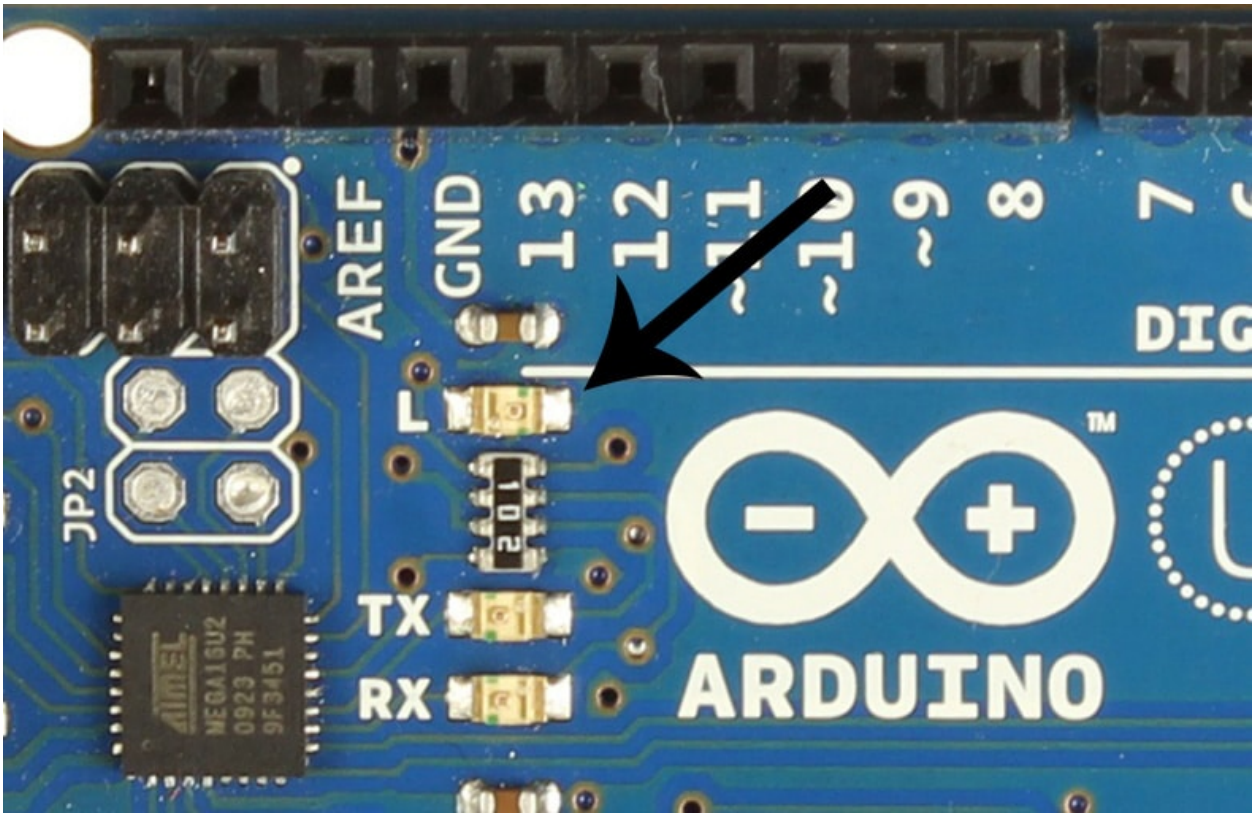


Fig 1.5

1.4.Basic Electronics

The goal of this chapter is to provide some basic information about electronic circuits. We make the assumption that you have no prior knowledge of electronics, electricity, or circuits, and start from the basics.

What is A Circuit ?

Before you design an electronic project, you need to know what a circuit is and how to create one properly.

An electronic circuit is a circular path of conductors by which electric current can flow. A closed circuit is like a circle because it starts and ends at the same point forming a complete loop. Furthermore, a closed circuit allows electricity to flow from the (+) power to the (-) ground uninterrupted.

In contrast, if there is any break in the flow of electricity, this is known as an open circuit. As shown below, a switch in a circuit can cause it to be open or closed depending on it's position.

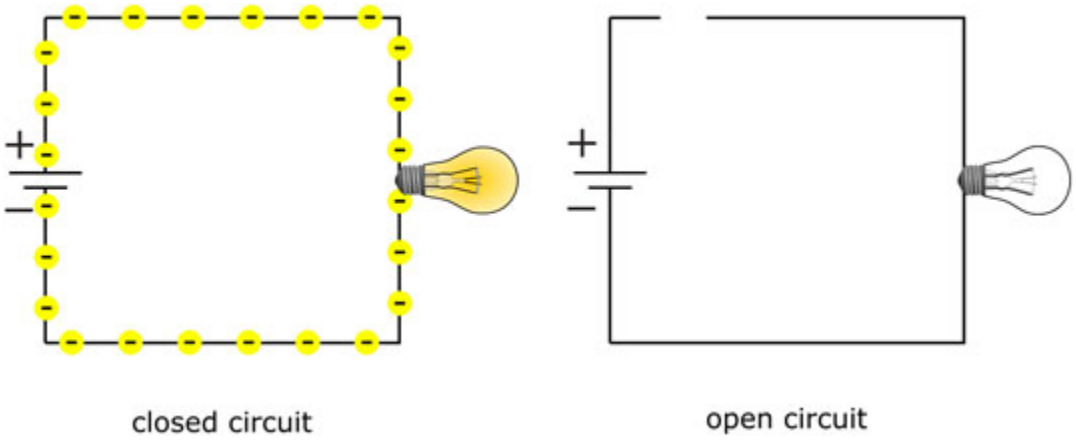


Fig 1.6

All circuits need to have three basic elements. These elements are a voltage source, conductive path and a load.

The voltage source, such as a battery, is needed in order to cause the current to flow through the circuit. In addition, there needs to be a conductive path that provides a route for the electricity to flow. Finally, a proper circuit needs a load that consumes the power. The load in the above circuit is the light bulb.

To implement closed circuits using breadboards The diagram below shows how. To know how to use it, read about them in the components section.

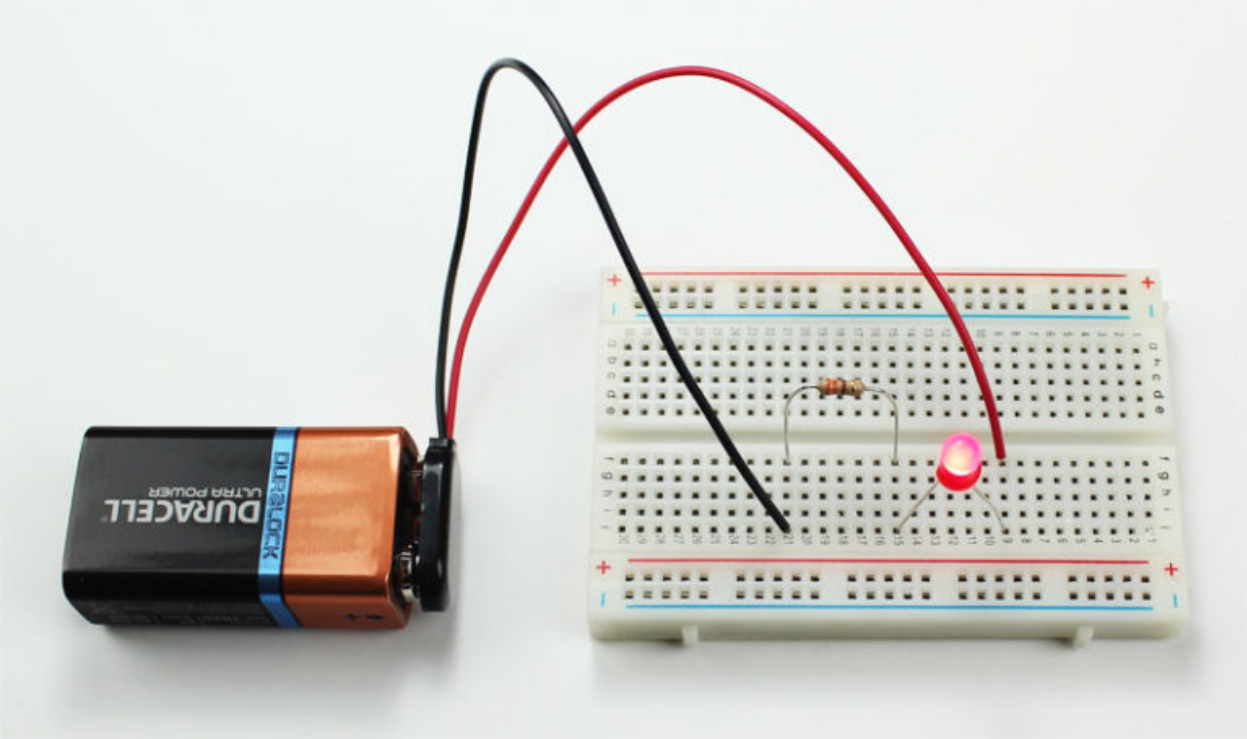


Fig 1.7

Parallel and Series circuit

In electrical and electronics engineering it is very important to know the differences between series and parallel circuits. They are the two most basic forms of electrical circuit and the other one being the series-parallel circuit, which is the combination of both, and can be understood by applying the same rules.

Series connection

A series connection between components is when two or more than two components are connected together in a cascaded form or the tail of the 1st component is connected to the head of the 2nd component and so on.or Series connected components form a chain-like structure in a single line.

Fig 1.8

Parallel Connection

A connection is said to be parallel if two or more than two components are connected together side by side or their heads are connected together and their tails are connected together or it's when two or more two-terminal elements are connected to the same two nodes. The parallel-connected components form multiple loops.

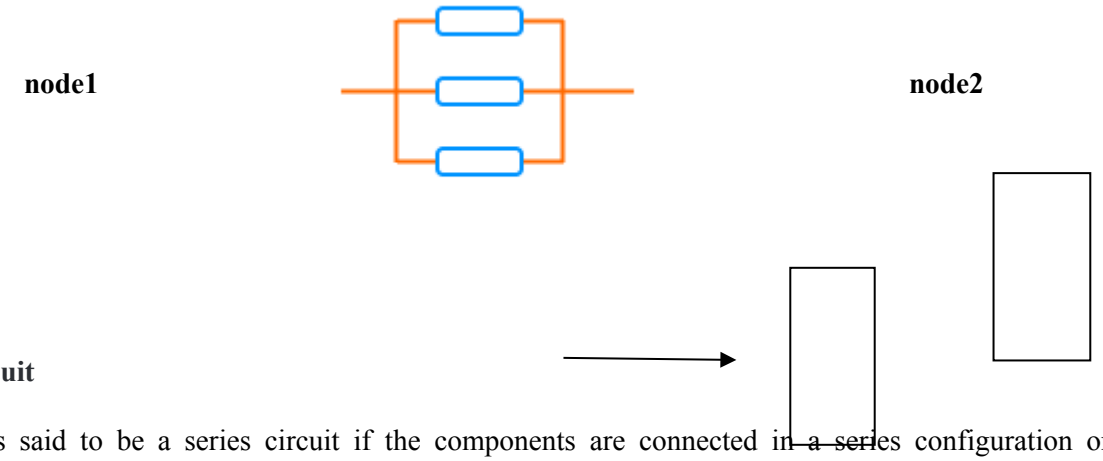


Fig 1.9

Series Circuit

A circuit is said to be a series circuit if the components are connected in a series configuration or cascaded formation in a single line. A series circuit forms a pathway that has only one loop, therefore, the current flowing through components is the same and the voltage divides depending on the resistance of each component.

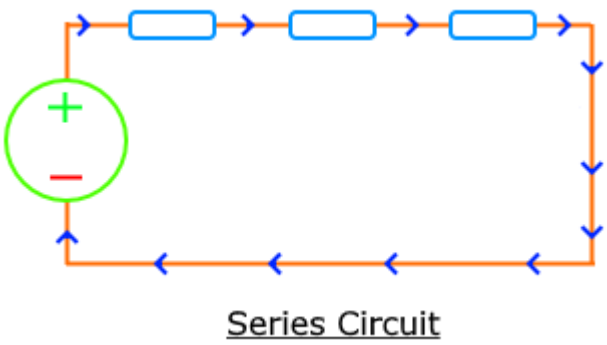
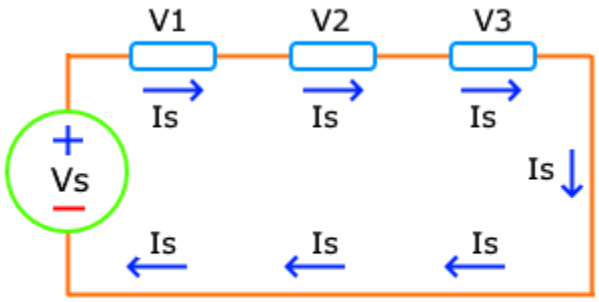


Fig 1.10

Current in Series Circuit

The current through each component in a series circuit remains the same and it is equal to the current supplied by the power source. Since there is only one path for the current flow, the current does not divide.

$I_T = I_1 += I_2 = I_3 + \ldots I_n$



Current & Voltage in Series Circuit

Fig 1.11

Voltage in Series Circuit

The sum of the voltage drops across each component in a series circuit is equal to the supply voltage. The voltage is a series circuit divided among the components based on their resistance. Therefore, the voltage drop across each component is different and depends on the value of resistance of the components.

Resistance in Series Circuit:

When resistors are connected in a series configuration their total resistance adds up and it is the sum of individual resistance of each resistor.

$Req = R1 + R2 + R3 + \ldots Rn$

The total resistance in a series circuit is always greater than its individual resistance.

Capacitor in a Series Circuit:

When capacitors are connected in series their total or equivalent capacitance decreases because the voltage difference across each capacitor decreases and the charge stored due to this voltage also decreases.

$1/Ceq = 1/C1 + 1/C2 + 1/C3 + \ldots 1/Cn$

The total capacitance in a series circuit is always less than the individual capacitance.

Inductor is Series Circuit:

The total inductance of two or more than two inductors in a series circuit is the sum of individual inductance.

$Leq = L1 + L2 + L3 + \ldots Ln$

The total inductance increases and it is always greater than the individual inductance in a series circuit.

Fault in Series Circuit:

If there is a fault in any component in a series circuit, the whole circuit does not work because the current flow breaks and there is no other path for the current to flow. So a fault in a single component will cause the whole circuit to deactivate. In order to troubleshoot a series circuit, you have to check each component. Thus, the troubleshooting series circuit is difficult than a parallel circuit.

A common example of the series circuit would be the Christmas lights, they are connected in series. If one of them stops working, the whole string does not light up and it is very difficult to spot the defective light.

Power Supplies in Series Circuit:

If two or more than two power supplies are connected in series their total or equivalent voltage will be the sum of individual voltages while the total current supplied will remain the same as the current supplied by individual supply.

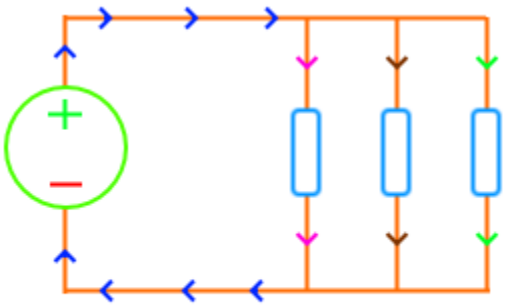
You may use the following electrical formulas to calculate the power in a series circuit:

or

So if you want to increase the voltage of the power supply connected then in series. For example, two 6v batteries connected in series will provide 12V of the total voltage.

Parallel Circuit

A circuit is said to be a parallel circuit if the electrical components are connected in a parallel configuration or their ends are connected to a common point. It forms multiple loops or pathways for the current to flow.



Parallel Circuit

Fig 1.12

Current in Parallel Circuit

The current in a parallel circuit divides and branches through each pathway. There the total current or supply current is equal to the sum of the currents through individual components and it depends on the value of their resistance.

I-T = I1 + I2 + I3 + … In

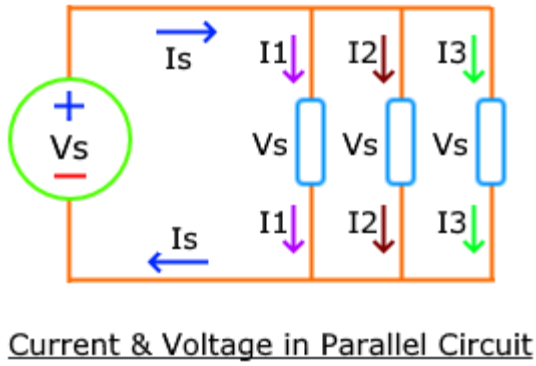


Fig 1.13

Voltage in Parallel Circuit

The voltage in a parallel circuit remains the same across each path or component because each component is connected to the source at the same points.

VT = V1 = V2 = V3 = … Vn

Resistors in Parallel Circuit

The total resistance of multiple resistors in a parallel circuit decreases and it is less than the individual resistance.

1/Req = 1/R1 + 1/R2 + 1/R3 + … 1/Rn

Capacitance in Parallel Circuit

The total capacitance of capacitors connected in parallel circuit increases and it is the sum of the capacitance of the individual capacitors.

$$Ceq = C1 + C2 + C3 + \dots Cn$$

The total or equivalent capacitance is always greater than individual capacitance

Inductance in Parallel Circuit

The total inductance of the inductors connected in a parallel circuit decreases and it is always less than the inductance of any individual inductors.

$$1/Leq = 1/L1 + 1/L2 + 1/L3 + \dots 1/Ln$$

Power supplies in Parallel Circuit

When power supplies are connected in a parallel configuration, their total voltage remains the same while their total current supplied is the sum of current supplied by an individual power supply.

The following formulas can be used to calculate the power in a parallel circuit:

or

Fault in Parallel Circuit

If there is a fault in any component connected in a parallel circuit, the other components still work because there are multiple pathways for the current to flow. It is easier to troubleshoot components in a parallel circuit because it is easier to identify the branch having the problem.

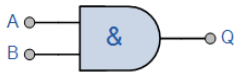
The appliances in our homes are connected in a parallel circuit. Because the voltage does not divide among the appliances and if any of them stop working, it does not affect any other appliances. It is easier to spot where the problem is.

1.5.DIGITAL LOGIC GATES

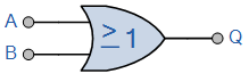
Digital Logic Gates can be made from discrete components such as Resistors, Transistors and Diodes to form RTL (resistor-transistor logic) or DTL (diode-transistor logic) circuits, but today's modern digital 74xxx series integrated circuits are manufactured using TTL (transistor-transistor logic) based on NPN bipolar transistor technology or the much faster and low power CMOS based MOSFET transistor logic used in the 74Cxxx, 74HCxxx, 74ACxxx and the 4000 series logic chips.

Standard Logic Gates

AND gate

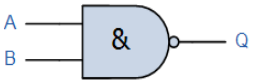
Symbol	Truth Table		
 2-input AND Digital Logic Gate	B	A	Q
	0	0	0
	0	1	0
	1	0	0
	1	1	1
Boolean Expression **Q = A.B**	Read as A **AND** B gives Q		

OR gate

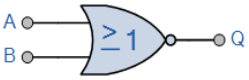
Symbol	Truth Table		
	B	A	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	1
Boolean Expression Q = A + B	Read as A **OR** B gives Q		

Inverting Logic Gates

NAND gate

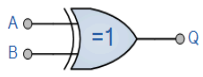
Symbol	Truth Table		
 2-input NAND Gate	A	B	Q
	0	0	1
	0	1	1
	1	0	1
	1	1	0
Boolean Expression Q = $\overline{A.B}$	Read as A AND B gives NOT-Q		

NOR gate

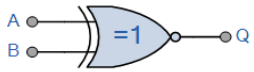
Symbol	Truth Table		
	B	A	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	0
Boolean Expression Q = $\overline{A + B}$	Read as A **OR** B gives **NOT** Q		

Exclusive Logic Gates

Exclusive-OR (Ex-OR / XOR) gate

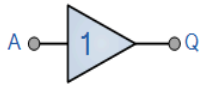
Symbol	Truth Table		
	B	A	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	0
Boolean Expression $Q = A \oplus B$	Read as A **OR** B but not **BOTH** gives Q (odd)		

Exclusive-NOR (Ex-NOR/XNOR) gate

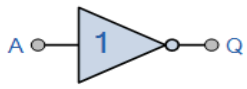
Symbol	Truth Table		
	B	A	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	1
Boolean Expression $Q = \overline{A \oplus B}$	Read if A **AND** B the **SAME** gives Q (even)		

Single Input Logic Gates

The Hex Buffer

Symbol	Truth Table	
	A	Q
	0	0
	1	1
Boolean Expression Q = A	Read as **A** gives **Q**	

NOT (Inverter) gate

Symbol	Truth Table	
	A	Q
	0	1
	1	0
Boolean Expression Q = not A or $\overline{A}$	Read as inverse of **A** gives Q	

Digital Logic Gate Truth Table Summary

The following logic gates truth table compares the logical functions of the 2-input logic gates detailed above.

Inputs		Truth Table Outputs For Each Gate					
B	A	AND	NAND	OR	NOR	EX-OR	EX-NOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

Truth Table Output for Single-input Gates		
A	NOT	Buffer
0	1	0
1	0	1

2. Components

2.1. Bread board

Bread boards are essential for testing and prototyping circuits because no soldering is needed. Components and wires are pushed into the holes to form a temporary circuit. Because it's not permanent, you can experiment and make changes until the desired outcome is reached.

Below the holes of each row are metal clips that connect the holes to each other. The middle rows run vertically as shown while the exterior columns are connected horizontally. These exterior columns are called power rails and are used to receive and provide power to the board.

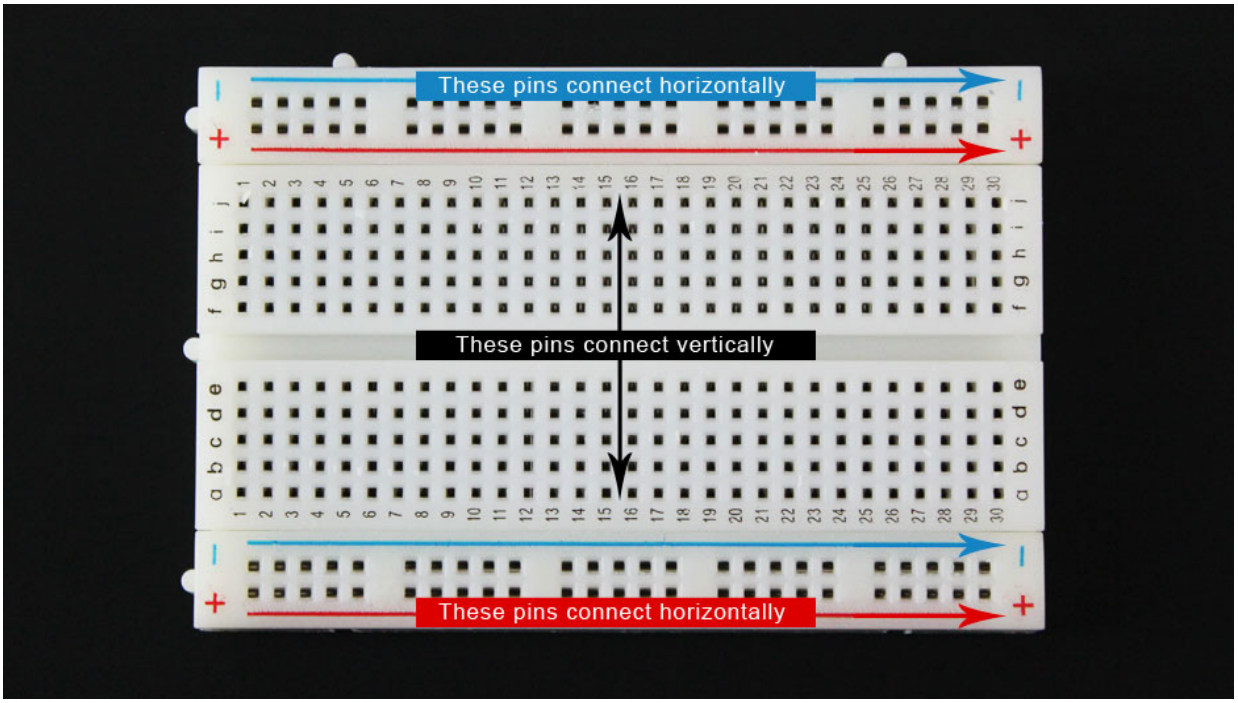


Fig 2.1

2.2. Push button

A Push Button is a type of switch that works on a simple mechanism called “Push-to-make”. Initially, it remains in off state or normally open state but when it is pressed, it allows the current to pass through it or we can say it makes the circuit when pressed. Normally their body is made up of plastic or metal in some types.

Push Button structure has four legs, two on one side and other two on another side. So, we can operate two lines of the circuit by single Push Button. Two legs on both the sides are internally connected as shown in the figure below.

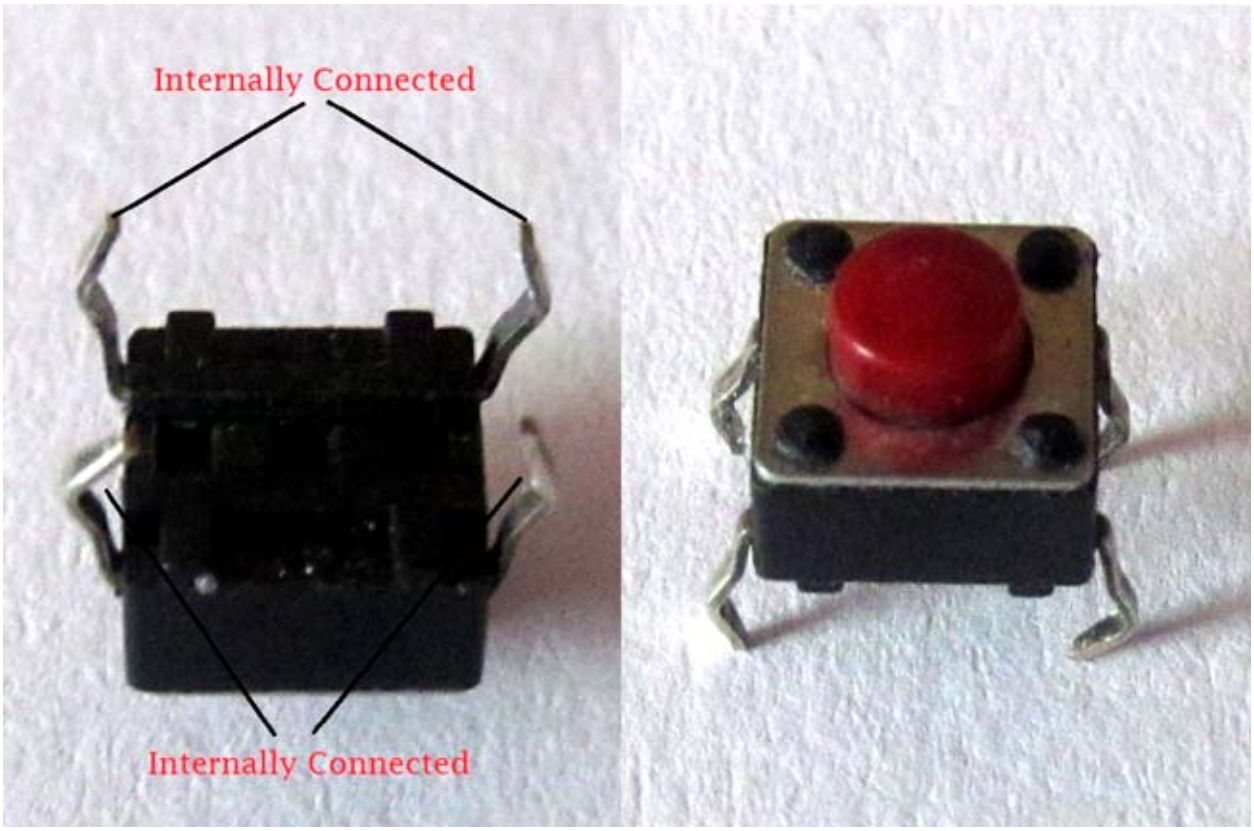


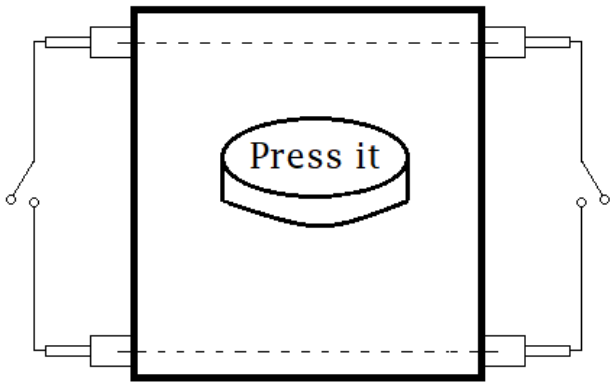
Fig 2.2

Tip

To check internally connection

Use a multimeter in continuity mode and connect the probe to each leg to determine which legs are internally connected. If the multimeter makes a sound, the legs are internally connected.

The working concept of Push Button is given above, till the button pressed it conducts current through it



or makes the circuit. As the button is released it breaks the circuit again. The figure below shows how .

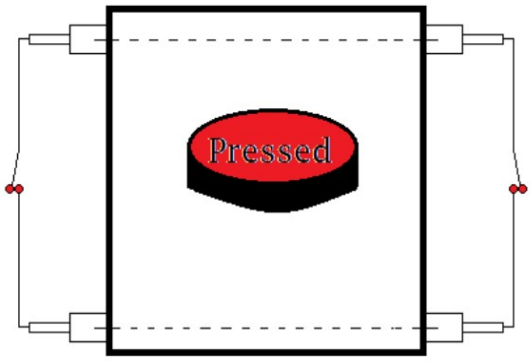


Fig 2.3

2.3. Buzzer

A buzzer, also known as a beeper, is an audio signaling device that can be mechanical, electromechanical, or piezoelectric (piezo for short). We only see Piezo Buzzers in this manual. Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

The main working principle is based on the theory that, whenever an electric potential is applied across a piezoelectric material, a pressure variation is generated.When current is applied to the buzzer it causes the ceramic disk to contract or expand. Changing the This then causes the surrounding disc to vibrate. That's the sound that you hear. By changing the frequency of the buzzer, the speed of the vibrations changes, which changes the pitch of the resulting sound.

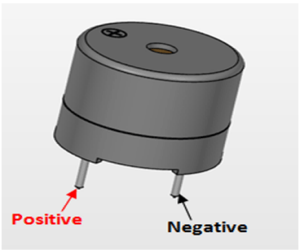


Fig 2.4

2.4.PHOTORESISTOR (LDR)

A photoresistor (or light-dependent resistor, LDR, or photo-conductive cell) is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits, and light-activated and dark-activated switching circuits.

A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several MegaOhms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms.

Photo resistor should not be confused with photoresist

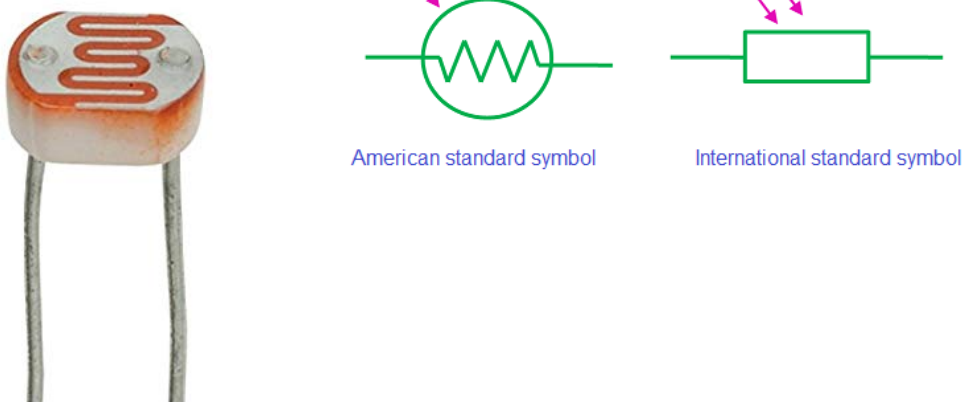


Fig 2.5

2.5. POTENTIOMETER

A potentiometer is a three terminal resistor in which the resistance is manually varied to control the flow of electric current. Among the three terminals two are fixed and one is variable. The two fixed terminals of the potentiometer are connected to both ends of the resistive element called track and the third terminal is connected to the sliding wiper. The wiper that moves along the resistive element varies the resistance of the potentiometer. The resistance of the potentiometer is changed when the wiper is moved over the resistive path.

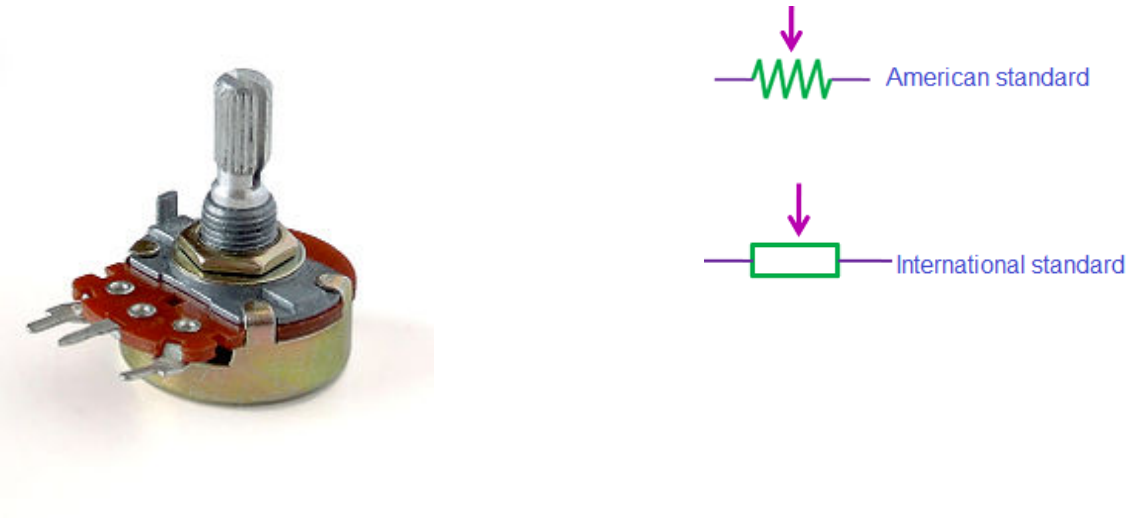


Fig 2.6

2.6. RESISTOR

Resist the flow of electrical energy in a circuit, changing the voltage and current as a result. Resistor values are measured in ohms. The colored stripes on the sides of resistors indicate their value (see resistor color code table).

Fig 2.7

How to read a color code of a resistor ?

The color code is given by several bands. Together they specify the resistance value, the tolerance, and sometimes the reliability or failure rate. The number of bands varies from three to six. The following figure shows the range of the bands and the color code .

Color	Signficant figures			Multiply	Tolerance (%)	Temp. Coeff. (ppm/K)	Fail Rate (%)
black	0	0	0	x 1		250 (U)	
brown	1	1	1	x 10	1 (F)	100 (S)	1
red	2	2	2	x 100	2 (G)	50 (R)	0.1
orange	3	3	3	x 1K		15 (P)	0.01
yellow	4	4	4	x 10K		25 (Q)	0.001
green	5	5	5	x 100K	0.5 (D)	20 (Z)	
blue	6	6	6	x 1M	0.25 (C)	10 (Z)	
violet	7	7	7	x 10M	0.1 (B)	5 (M)	
grey	8	8	8	x 100M	0.05 (A)	1(K)	
white	9	9	9	x 1G			
gold			3th digit only for 5 and 6 bands	x 0.1	5 (J)		
silver				x 0.01	10 (K)		
none					20 (M)		

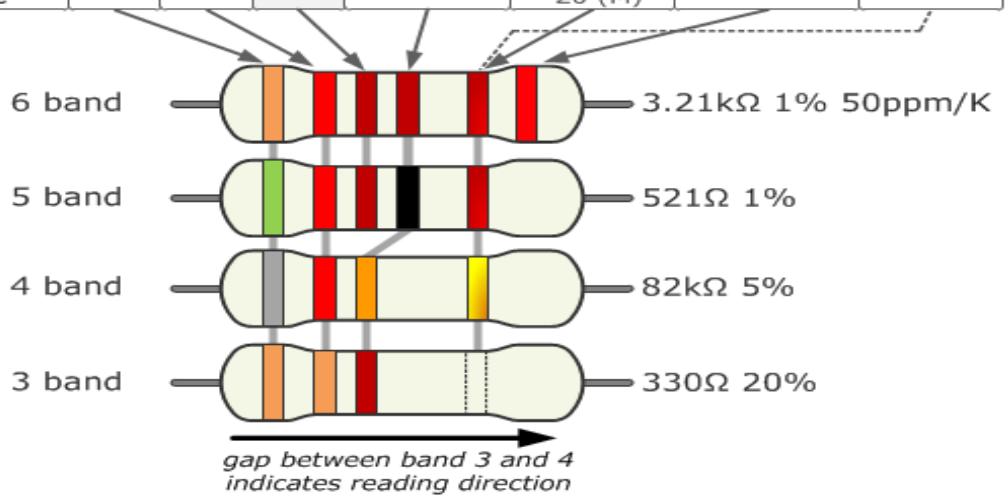


Fig 2.8

3 band resistor

For a 3-band resistor color code, the first two bands always denote the first two digits of the resistance value while the third band represents the multiplier.In the figure shown in the resistor color code table above , the 3 bands are orange, orange and brown. By using the color code chart, one finds that orange for 3. The third band is the multiplier, with brown representing a multiplier value of 1(). Therefore, the value of this resistor is 33 =330

4 band resistor

The four band color code is the most common variation. These resistors have two bands for the resistance value, one multiplier and one tolerance band.In the figure shown in the resistor color code table above , the 4 bands are silver, red, orange and gold. By using the color code chart, one finds that silver stands for 8 and red for 2. The third band is the multiplier, with orange representing a multiplier value of 3(). Therefore, the value of this resistor is 82 =82000

5 band resistor

Resistors with high precision have an extra band to indicate a third significant digit. Therefore, the first three bands indicate the significant digits, the fourth band is the multiplication factor, and the fifth band represents the tolerance. For the example shown here: green (5), red(2), brown (1), black (x 10_0 = x1), brown (1%) represents a 147 Ω resistor with a 0.5% tolerance.

6 band resistor

Resistors with 6 bands are usually for high precision resistors that have an additional band to specify the temperature coefficient (ppm/˚C = ppm/K). The most common color for the sixth band is brown (100 ppm/˚C). This means that for a temperature change of 10 ˚C, the resistance value can change 1000 ppm = 0.1%. For the 6 band resistor example shown above: orange (3), red (2), brown (1), brown (x10), brown (1%), red (50 ppm/°C) represents a 3.21 kΩ resistor with a 1% tolerance and a 50 ppm/°C temperature coefficient.

2.7. DIODE

A diode is a two terminal semiconductor device that essentially acts as a one-way switch for current. It allows current to flow easily in one direction, but severely restricts current from flowing in the opposite direction.

Diodes have polarity, determined by an anode (positive lead) and cathode (negative lead). Most diodes allow current to flow only when positive voltage is applied to the anode.

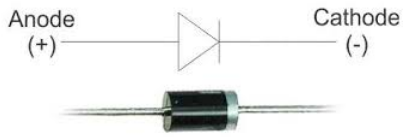


Fig 2.9

There are several types of diodes available for use in electronics design, namely; BARITT diode, Gunn Diode, Laser diode, Light emitting diodes (LED), Photodiode, PIN diode, Schottky diodes, Step recovery diode, Tunnel diode, Varactor diode and a Zener diode.

From this collection of diodes the ones specific for our purpose are:-

Light Emitting Diode (LED)

The term LED stands for light emitting diode, is one of the most standard types of the diode. When the diode is connected in forwarding bias, then the current flows through the junction and generates the light.

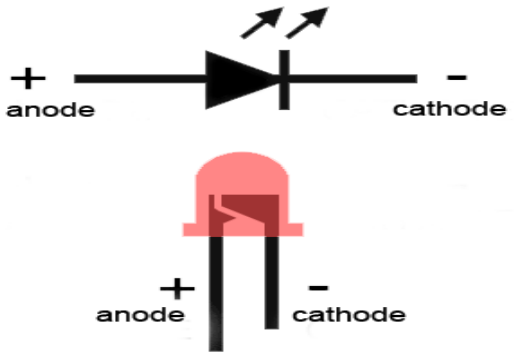
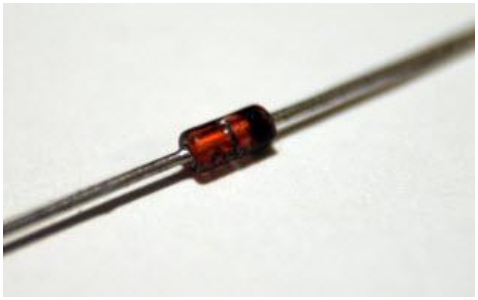


Fig 2.10

Zener Diode

The Zener diode is used to provide a stable reference voltage. As a result, it is used in vast amounts. It works under reverse bias condition and found that when a particular voltage is reached it breaks down. If the flow of current is limited by a resistor, it activates a stable voltage to be generated. This type of diode is widely used to offer a reference voltage in power supplies



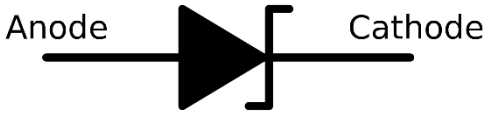


Fig 2.11

2.8. ULTRASONIC SENSOR

As the name indicates, ultrasonic sensors measure distance by using ultrasonic waves.
The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception. Considering the fact that the time spent is for both go and return.

Ultrasonic sensors use sound to determine the distance between the sensor and the closest object in its path. This sensors are essentially sound sensors, but they operate at frequency above human hearing.

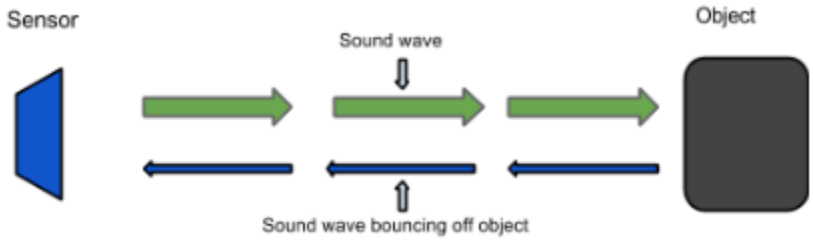


Fig 2.12

The sensor sends out a sound wave at a specific frequency. It then listens for that specific sound wave to bounce off of an object and come back. The sensor keeps track of the time between sending the wave and the sound wave returning. If you know how fast something is going and how long it is traveling you can find the distance traveled as follows

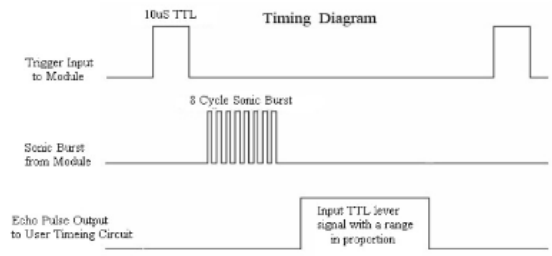
D (distance) = v (speed) * t (time) ---------- equation 1

This HC-SR04 (ultrasonic sensor) has four pins which are VCC, GND, TRIG, and ECHO. The VCC and GND pins power the HC-SR04. These pins need to be attached to a +5 volt source and ground respectively. There is a single control pin: the TRIG pin. The TRIG pin is responsible for sending the ultrasonic burst. This pin should be set too HIGH for 10µs, at which point the HC-SR04 will send out an eight cycle sonic burst at 40 kHz. After a sonic burst has been sent the ECHO pin will go HIGH. The ECHO pin is the data pin; it is used in taking distance measurements. After an ultrasonic burst is sent the pin will go HIGH, it will stay high until an ultrasonic burst is detected back, at which point it will go LOW.

To interface an ultrasonic sensor use the following steps

1) Set TRIG to HIGH
2) Set a timer when ECHO goes to HIGH

3) Keep the timer running until ECHO goes to LOW
4) Save that time
5) Use equation 1 to determine the distance traveled



More details could be found from HC-SR04 Datasheet

HC-SR04

Fig 2.13

2.9. RGB

An RGB LED is basically an LED package that can produce almost any color. It can be used in different applications such as outdoor decoration lighting, stage lighting designs, home decoration lighting, LED matrix display, and more.

RGB LEDs have three internal LEDs (Red, Green, and Blue) that can be combined to produce almost any color output. In order to produce different kinds of colors, we need to set the intensity of each internal LED and combine the three color outputs. In this tutorial, we are going to use PWM to adjust the intensity of the red, green, and blue LEDs individually and the trick here is that our eyes will see the combination of the colors, instead of the individual colors because the LEDs are very close to each other inside.

RGB types and structures

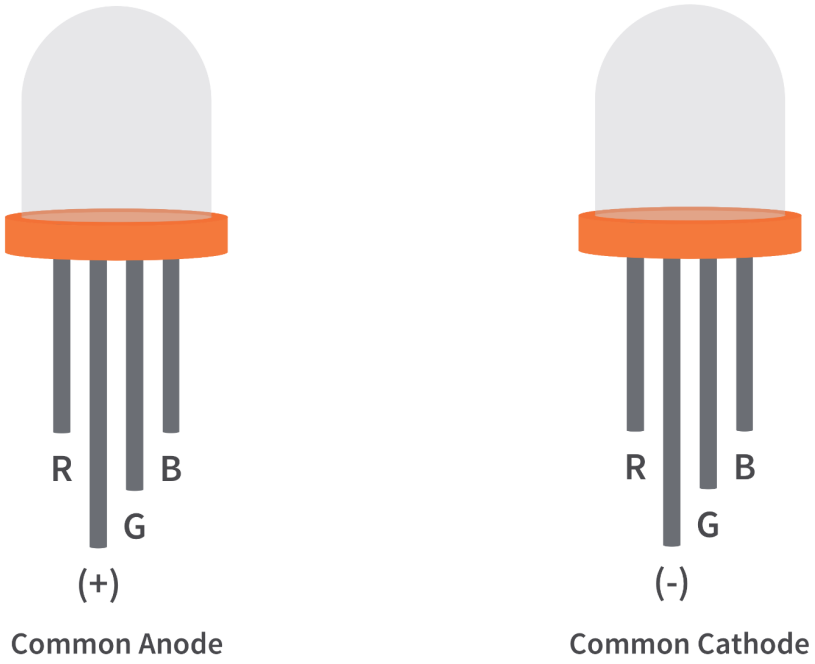


Fig 2.14

Common Anode

In a common anode RGB LED, the anode of the internal LEDs are all connected to the external anode lead. To control each color, you need to apply a LOW signal or ground to the red, green, and blue leads and connect the anode lead to the positive terminal of the power supply.

Common Anode RGB LED Pinout

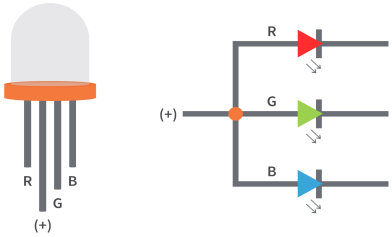


Fig 2.15

Common Cathode

In a common cathode RGB LED, the cathode of the internal LEDs are all connected to the external cathode lead. To control each color, you need to apply a HIGH signal or VCC to the red, green, and blue leads and connect the anode lead to the negative terminal of the power supply

Common Cathode RGB LED Pinout

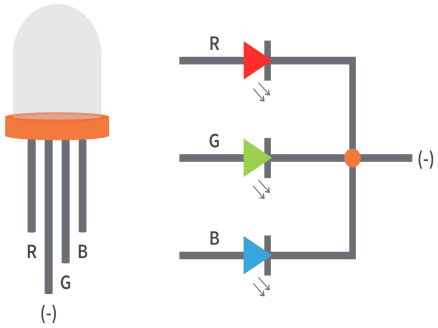


Fig 2.16

2.10. LCD

The term LCD stands for liquid crystal display. 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. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.

Hardware overview for 16x2 character LCD

These LCDs are ideal for displaying text/characters only, hence the name ‘Character LCD’. The display has an LED backlight and can display 32 ASCII characters in two rows with 16 characters on each row.

16

2

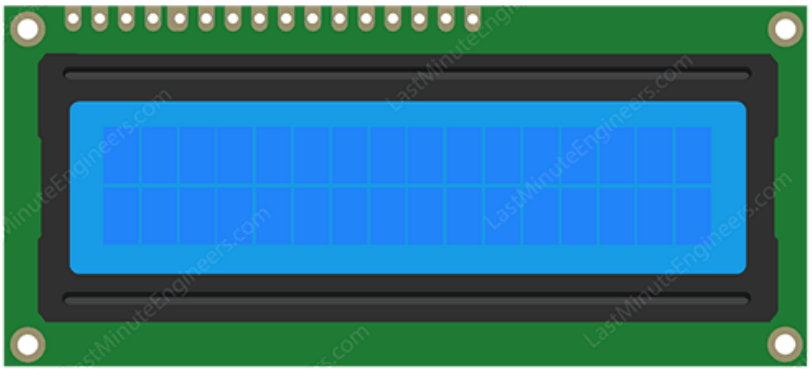


Fig 2.17

If you look closely, you can actually see the little rectangles for each character on the display and the pixels that make up a character. Each of these rectangles is a grid of 5×8 pixels.

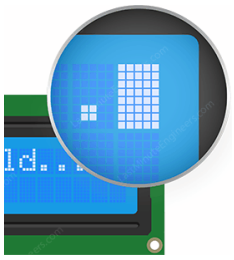


Fig 2.18

16×2 Character LCD Pinout

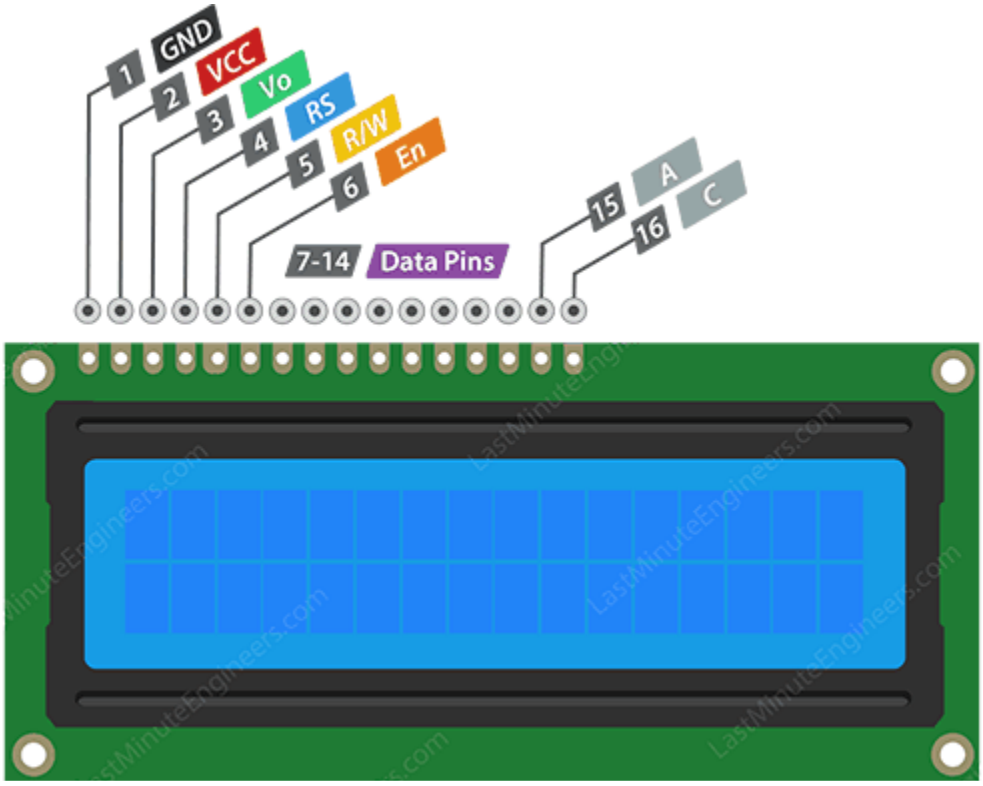


Fig 2.19

GND should be connected to the ground of the Arduino.

VCC is the power supply for the LCD which we connect to the 5 volts pin on the Arduino.

V0 controls the contrast and brightness of the LCD. Using a simple voltage divider with a potentiometer, we can make fine adjustments to the contrast.

RS pin lets the Arduino tell the LCD whether it is sending commands or the data. Basically this pin is used to differentiate commands from the data.

For example, when the RS pin is set to LOW, then we are sending commands to the LCD (like set the cursor to a specific location, clear the display, scroll the display to the right and so on). And when RS pin is set on HIGH we are sending data/characters to the LCD

R/W pin on the LCD is to control whether or not you're reading data from the LCD or writing data to the LCD. Since we're just using this LCD as an OUTPUT device, we're going to tie this pin LOW. This forces it into the WRITE mode.

En pin is used to enable the display. Meaning, when this pin is set to LOW, the LCD does not care what is happening with R/W, RS, and the data bus lines; when this pin is set to HIGH, the LCD is processing the incoming data.

(D7-D14) Data Pins are the pins that carry the 8 bit data we send to the display. For example, if we want to see the uppercase 'A' character on the display we will set these pins to 0100 0001(according to the ASCII table) to the LCD.

A-k (Anode and Cathode) pins are used to control the backlight of the LCD.

2.11. Matrix keypad

Matrix keypads are the kind of keypads you see on cell phones, calculators, microwaves ovens, door locks, etc. They're practically everywhere.

However, in DIY electronics, they are a great way to let users interact with your project and are often needed to navigate menus, punch in passwords and control robots

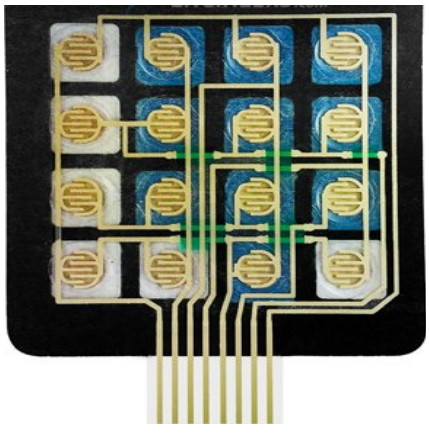


Fig 2.20

Hardware Overview – Membrane Keypad

Membrane keypads are made of a thin, flexible membrane material. They do come in may sizes 4×3, 4×4, 4×1 etc. Regardless of their size, they all work in the same way.

One of the great things about them is that they come with an adhesive backing so you can attach it to nearly anything. You just have to peel the paper backing off.

Fig 2.21

Pinout of 4×3 & 4×4 Membrane Keypad

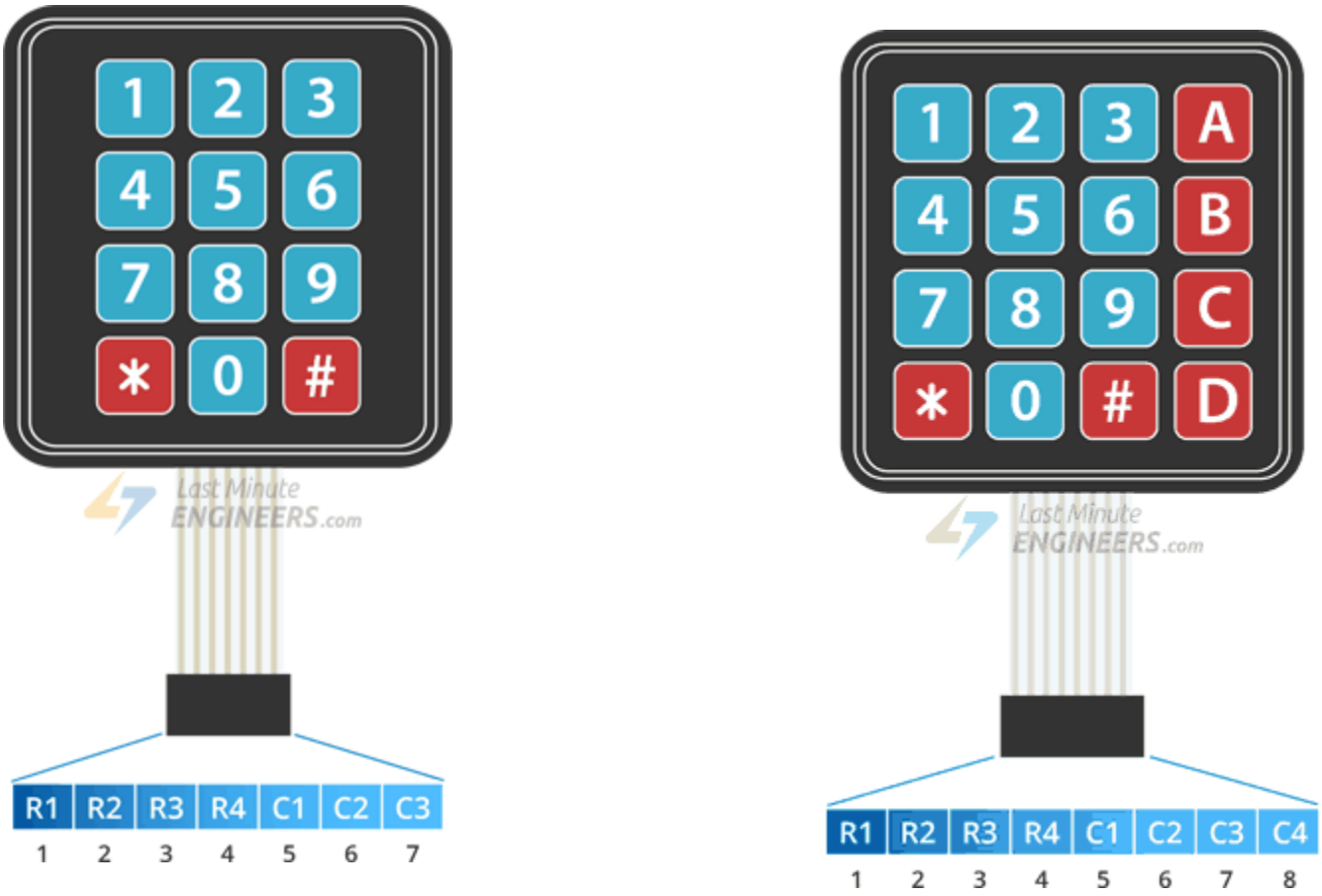


Fig 2.22

How keypad works & How to scan them?

The working principle is very simple. Pressing a button shorts one of the row lines to one of the column lines, allowing current to flow between them. For example, when key '4' is pressed, column 1 and row 2 are shorted.

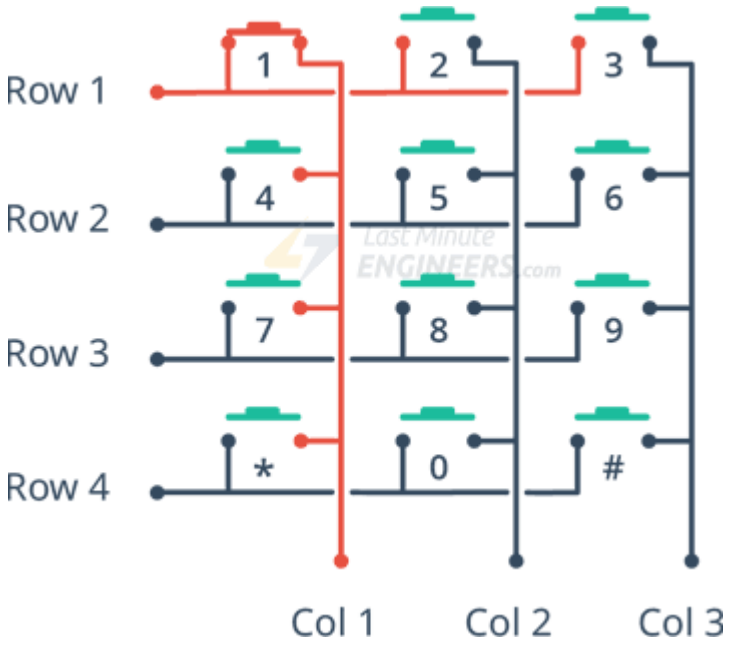


Fig 2.23

A microcontroller can scan these lines for a button-pressed state. To do this, it follows the procedure below.

1. Microcontroller sets all the column and row lines to input.
2. Then, it picks a row and sets it HIGH.
3. After that, it checks the column lines one at a time.
4. If the column connection stays LOW, the button on the row has not been pressed.
5. If it goes HIGH, the microcontroller knows which row was set HIGH, and which column was detected HIGH when checked.
6. Finally, it knows which button was pressed that corresponds to the detected row & column.

2.12. Temperature and Humidity sensor (DTT11 Sensor)

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). Its fairly simple to use, but requires careful timing to grab data.

DHT11 Module Hardware Overview

At the heart of the module is the digital temperature & humidity sensor manufactured by AOSONG – DHT11.DHT11 can measure temperature from 0°C to 50°C with ±2.0°C accuracy, and humidity from 20 to 80% with 5% accuracy.

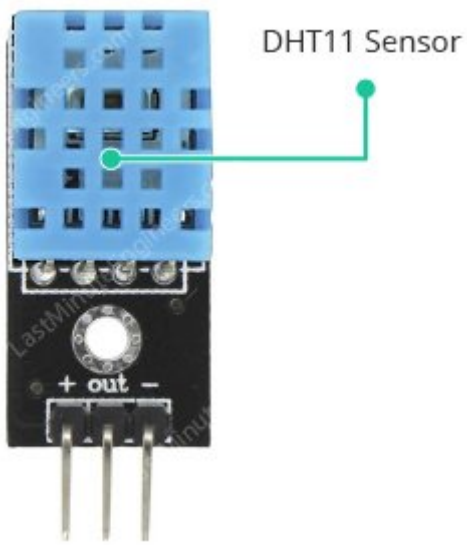


Fig 2.23

Note that the sampling rate of the DHT11 is 1Hz, meaning you can get new data from it once every second.

Supporting Circuitry

The module comes with all the essential supporting circuitry, so it should be ready to run without any extra components.

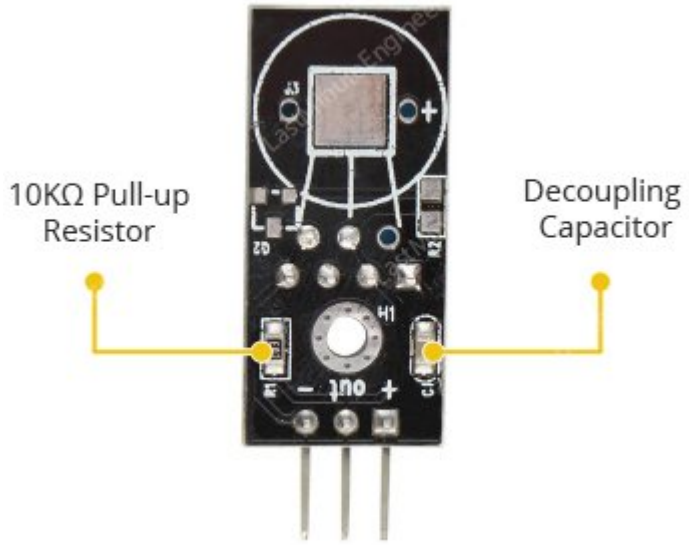


Fig 2.24

The DHT11 sensors usually require an external pull-up resistor of 10KΩ between VCC and Out pin for proper communication between sensor and the Arduino. However, the module has a built-in pull-up resistor, so you need not add it.

The module also has a decoupling capacitor for filtering noise on the power supply.

DHT11 Module Pinout

The DHT11 module is fairly easy to connect. It has only three pins:

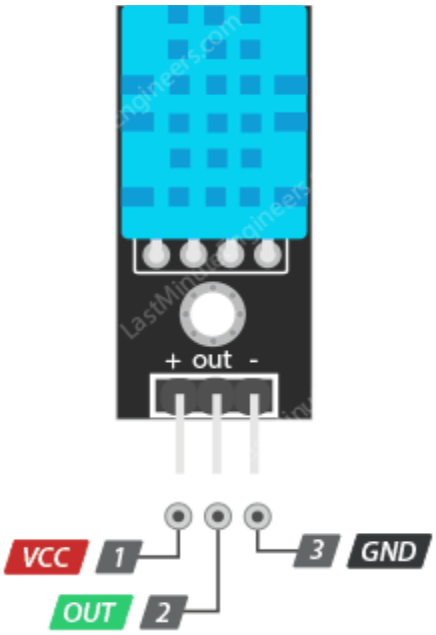


Fig 2.25

+ (VCC) pin supplies power for the sensor. 5V supply is recommended, although the supply voltage ranges from 3.3V to 5.5V. In case of 5V power supply, you can keep the sensor as long as 20 meters. However, with 3.3V supply voltage, cable length shall not be greater than 1 meter. Otherwise, the line voltage drop will lead to errors in measurement.

Out pin is used to communicate between the sensor and the Arduino.

2.13. Stepper Motor

Stepper motors are great motors for position control. They are a special type of brushless motors that divides a full rotation into a number of equal “steps”. They are usually found in desktop printers, 3D printers, CNC milling machines, and anything else that requires precise positioning control.

The 28BYJ-48 Stepper Motor

The 28BYJ-48 is a 5-wire unipolar stepper motor that runs on 5 volts. The interesting thing about this motor is that people have been using it in countless applications over the last few decades. It is used in air-conditioners, vending machines and many other applications.

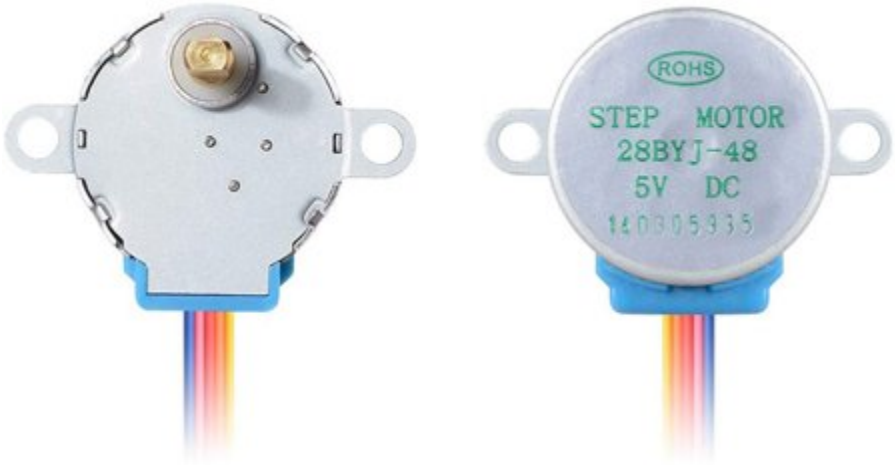


Fig 2.26

One of the best things about these motors is that they can be positioned accurately, one ‘step’ at a time.

The other advantage is that they are relatively precise in their movement and they are quite reliable since the motor does not use contact brushes.

They generally give good torque even in the stand-still state which is maintained as long as power is supplied to the motor.

The only downside is that they are a bit hungry for power and consume power even when they are not moving.

According to the data sheet, when the 28BYJ-48 motor runs in full step mode, each step corresponds to a rotation of 11.25°. That means there are 32 steps per revolution (360°/11.25° = 32).

Power Consumption

The power consumption of the motor is around 240mA.Because the motor draws too much power, it is best to power it directly from an external 5V power supply rather than drawing that power from the Arduino.The motor consumes power, even in the stand still state, to maintain its position.The motor usually comes with a ULN2003 based driver board.

The ULN2003 is one of the most common motor driver ICs, consisting of an array of 7 Darlington transistor pairs, each pair is capable of driving loads of up to 500mA and 50V. Four out of seven pairs are used on this board.

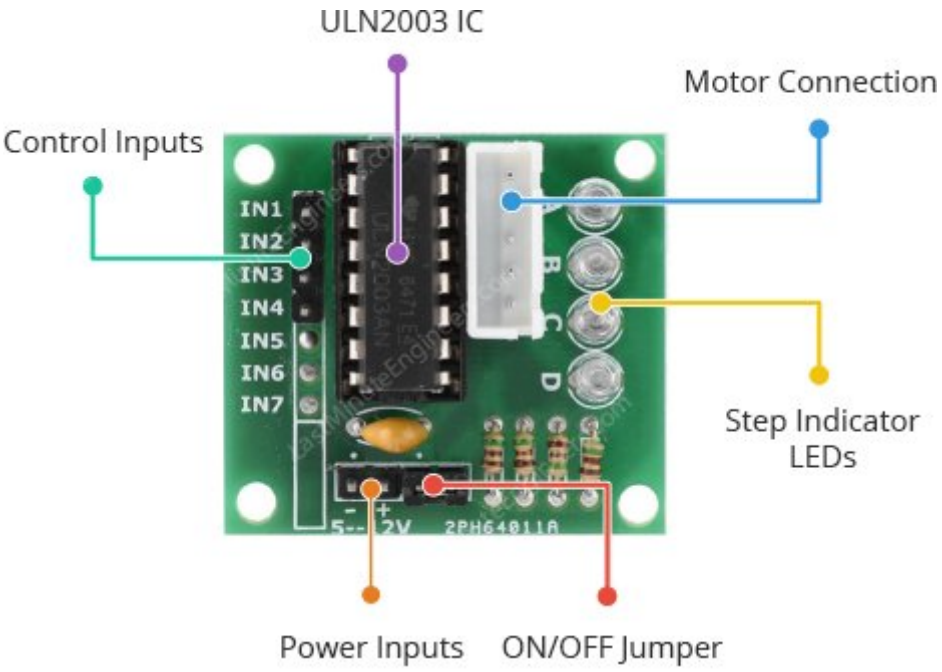


Fig 2.27

The board has a connector that mates the motor wires perfectly which makes it very easy to connect the motor to the board. There are also connections for four control inputs as well as power supply connections.

The board has four LEDs that show activity on the four control input lines (to indicate stepping state). They provide a nice visual when stepping.

The board also comes with an ON/OFF jumper to isolate power to the stepper Motor.

ULN2003 Stepper Driver Board Pinout

The pinouts of the ULN2003 stepper driver board are as follows:

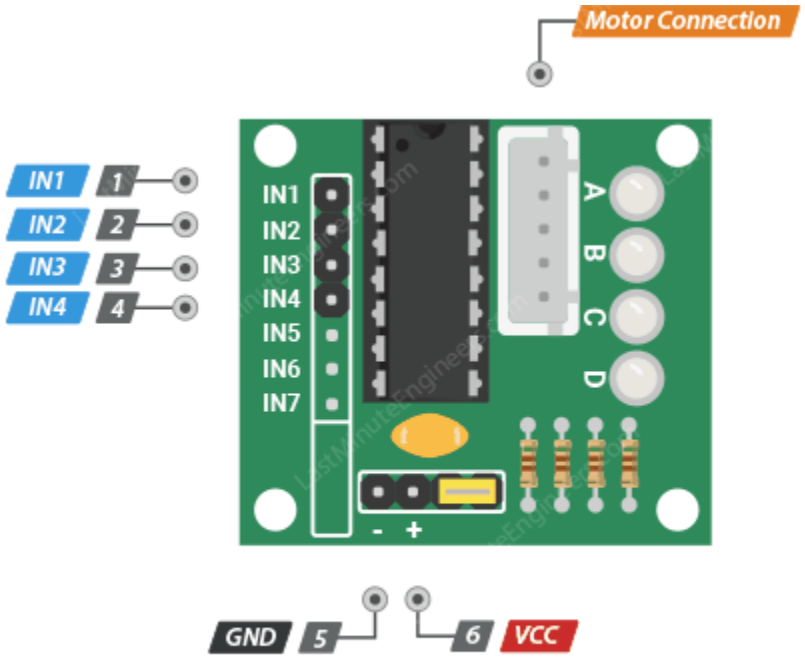


Fig 2.28

IN1 – IN4 pins are used to drive the motor. Connect them to a digital output pins on the Arduino.

GND is a common ground pin.

VDD pin supplies power for the motor. Connect it to an external 5V power supply. Because the motor draws too much power, you should NEVER use the 5V power from your Arduino to power this stepper motor.

Motor Connector This is where the motor plugs into. The connector is keyed, so it only goes in one way.

2.14. Servo Motor

Servo is a general term for a closed loop control system.A closed loop system uses the feedback signal to adjust the speed and direction of the motor to achieve the desired result.

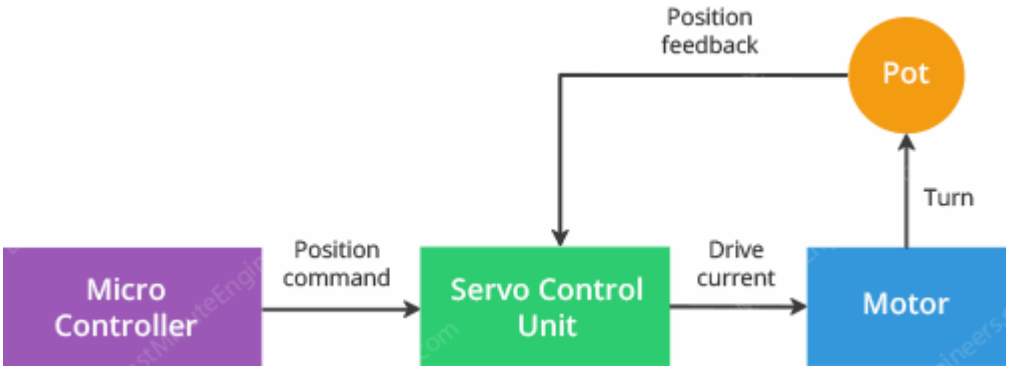


Fig 2.29

RC servo motor works on the same principal. It contains a small DC motor connected to the output shaft through the gears.

The output shaft drives a servo arm and is also connected to a potentiometer (pot).

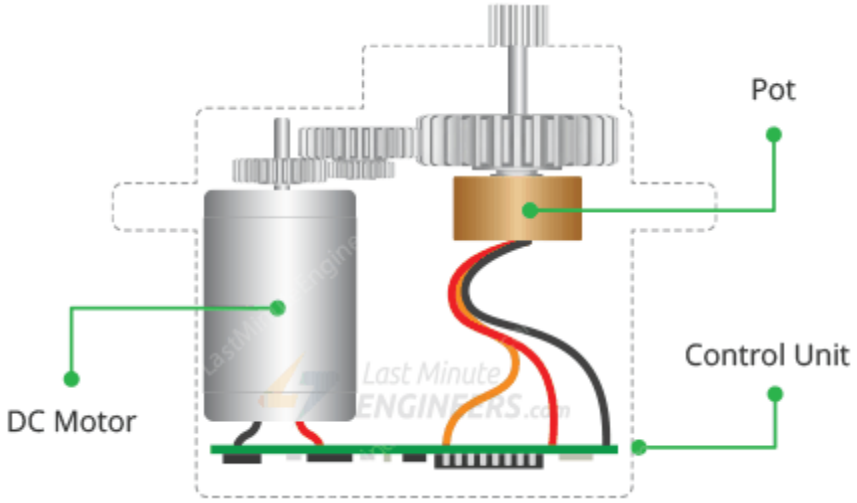


Fig 2.30

The potentiometer provides position feedback to the servo control unit where the current position of the motor is compared to the target position.

According to the error, the control unit corrects the actual position of the motor so that it matches the target position.

How Servo Motors Work?

You can control the servo motor by sending a series of pulses to the signal line. A conventional analog servo motor expects to receive a pulse roughly every 20 milliseconds (i.e. signal should be 50Hz).The length of the pulse determines the position of the servo motor.

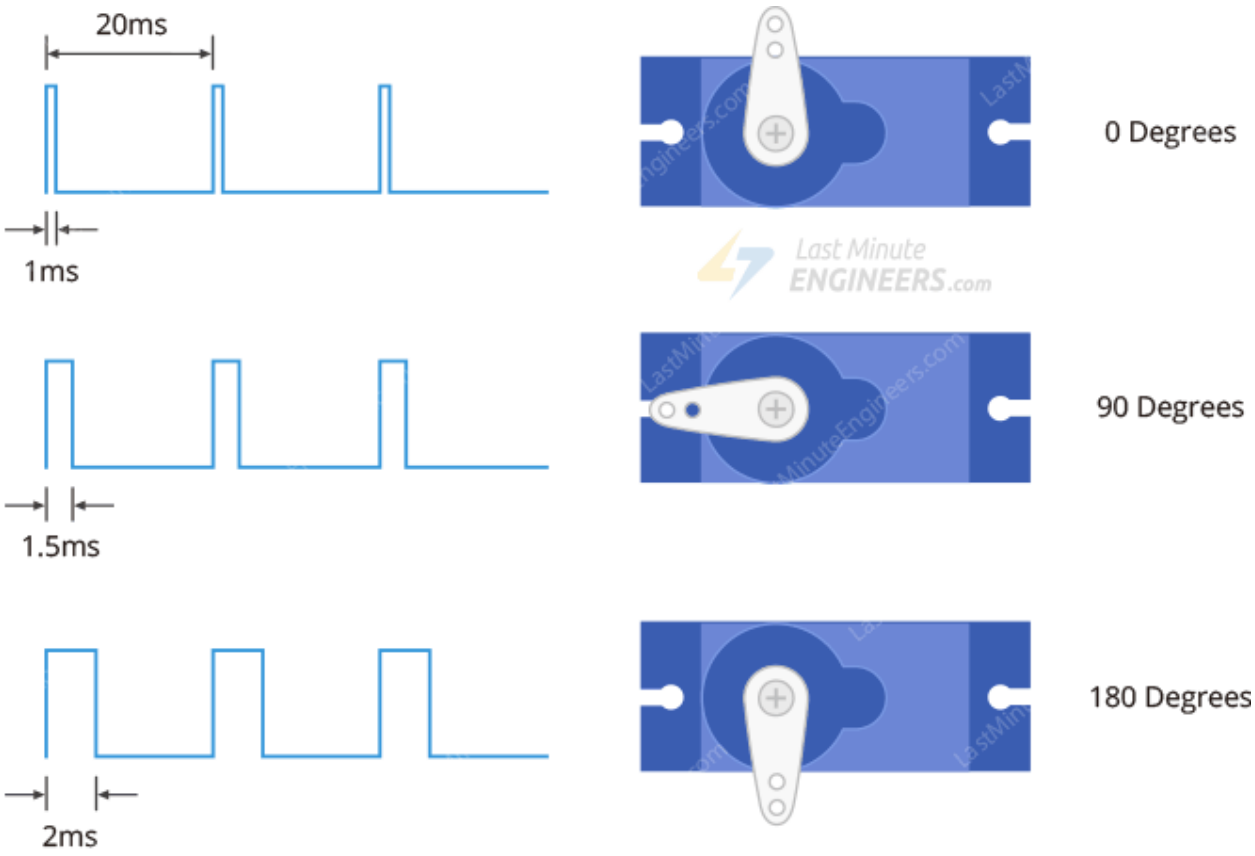


Fig 2.31

If the pulse is high for 1ms, then the servo angle will be zero.

If the pulse is high for 1.5ms, then the servo will be at its center position.

If the pulse is high for 2ms, then the servo will at 180 degrees.

Pulses ranging between 1ms and 2ms will move the servo shaft through the full 180 degrees of its travel.

Servo Motor Pinout

Servo motors typically have three connections and are as follows:

Fig 2.32

GND is a common ground for both the motor and logic.

5V is a positive voltage that powers the servo.

Control is input for the control system.

2.15. SIM900 GSM/GPRS shield

SIM900 GSM/GPRS shield is a GSM modem, which can be integrated into a great number of IoT projects. You can use this shield to accomplish almost anything a normal cell phone can; SMS text messages, Make or receive phone calls, connecting to internet through GPRS, TCP/IP, and more! To top it off, the shield supports quad-band GSM/GPRS network, meaning it works pretty much anywhere in the world.

Hardware Overview of SIM900 GSM/GPRS Shield

The SIM900 GSM/GPRS shield is designed to surround the SIM900 chip with everything necessary to interface with Arduino, plus a few extra goodies to take advantage of the chip's unique features.

Let's familiarize ourselves with these features and abilities of the shield. Here's a quick overview:

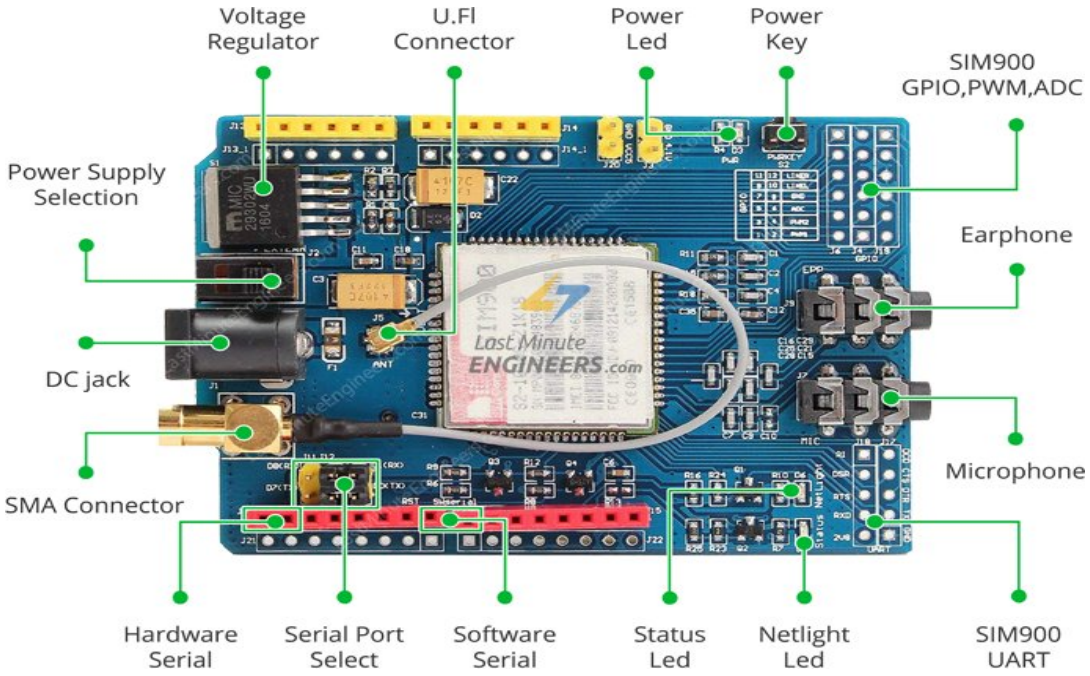


Fig 2.33

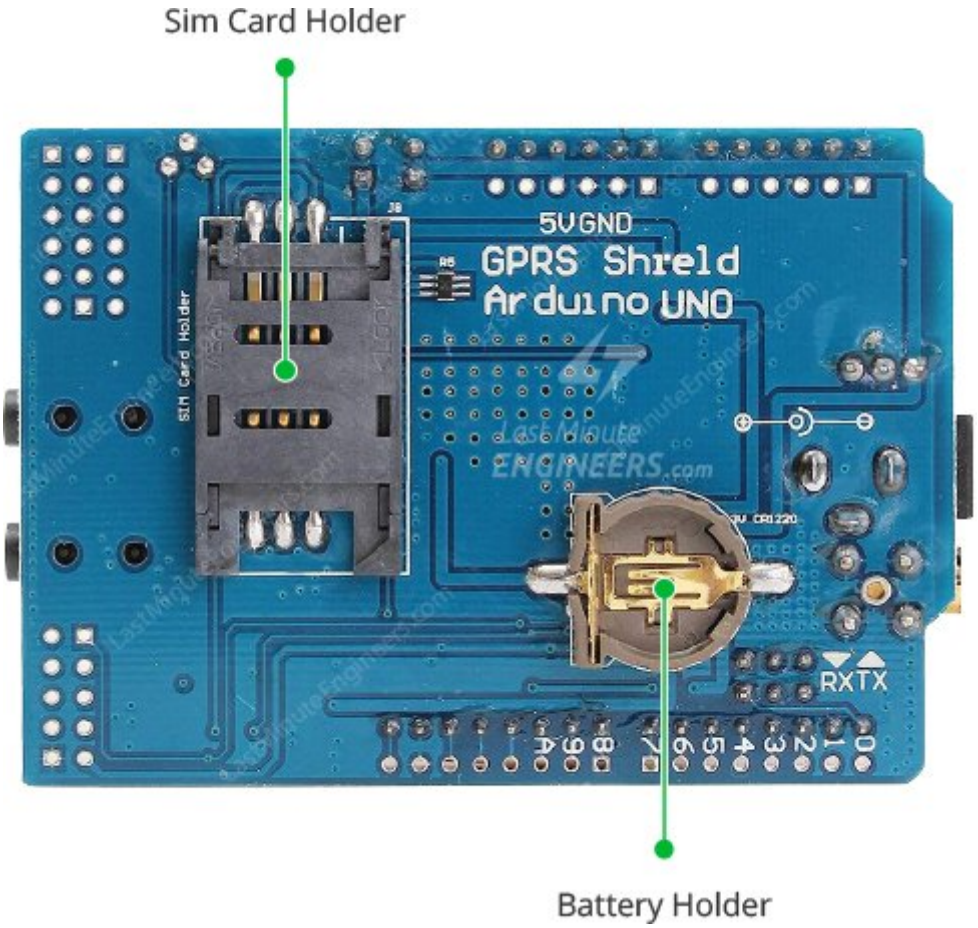


Fig 2.34

The SIM900 shield packs a surprising amount of features into its little frame. Some of them are listed below:

Supports Quad-band: GSM850, EGSM900, DCS1800 and PCS1900

Connect onto any global GSM network with any 2G SIM

Make and receive voice calls using an external earphone & electret microphone

Send and receive SMS messages

Send and receive GPRS data (TCP/IP, HTTP, etc.)

Scan and receive FM radio broadcasts

Transmit Power:

Class 4 (2W) for GSM850

Class 1 (1W) for DCS1800

Serial-based AT Command Set

U.FL and SMA connectors for cell antenna

Accepts Full-size SIM Card

LED Status Indicators

There are three LEDs on the SIM900 GSM/GPRS shield which indicates connectivity or power status. By observing these LEDs you can get visual feedback on what's going on with the shield.

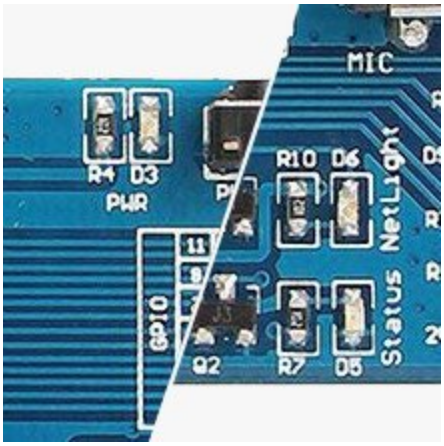


Fig 2.35

PWR: This LED is connected to the shield’s power supply line. If this LED is on, the shield is receiving power.

Status: This LED indicates SIM900’s working status. If this LED is on, the chip is in working mode.

Netlight: This LED indicates the status of your cellular network. It’ll blink at various rates to show what state it’s in.

off: The SIM900 chip is not running

64ms on, 800ms off: The SIM900 chip is running but not registered to the cellular network yet.

64ms on, 3 seconds off: The SIM900 chip is registered to the cellular network & can send/receive voice and SMS.

64ms on, 300ms off: The GPRS data connection you requested is active.

Supplying Power for SIM900 Shield

One of the most important parts of getting the SIM900 shield working is supplying it with enough power.

Depending on which state it's in, the SIM900 can be a relatively power-hungry device. The maximum current draw of the chip is around 2A during transmission burst. It usually won't pull that much, but may require around 216mA during phone calls or 80mA during network transmissions. This chart from the datasheet summarizes what you may expect:

The operating voltage of SIM900 chip is from 3.4V to 4.4V. To keep supply voltage safe at 4.1V, the shield comes with a high current, high accuracy, low-dropout voltage regulator MIC29302WU from Micrel – capable of handling load currents up to 3A.

Fig 2.36

You can add an external power supply to the shield with the 5.5mm DC jack, to which you can connect any 5V-9V DC wall adapter you have. Next to the DC jack, is a Slide Switch to select the power source labeled EXTERN. To use external power source, move the slider as shown above.

UART Communication

The SIM900 GSM/GPRS shield uses UART protocol to communicate with an Arduino. The chip supports baud rate from 1200bps to 115200bps with Auto-Baud detection.

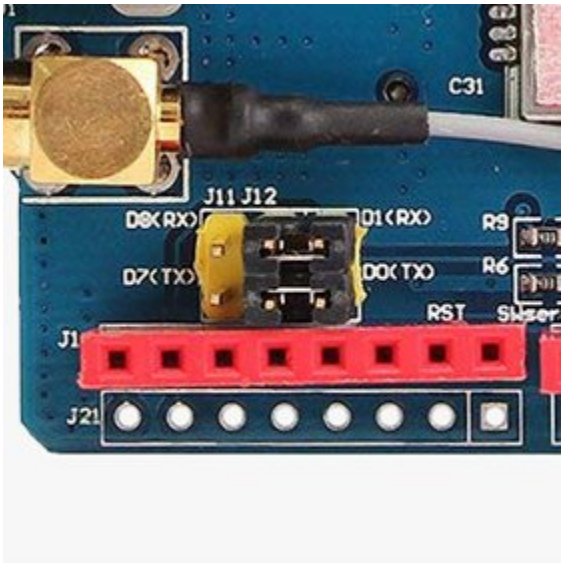


Fig 2.37

With the help of jumpers you can connect (RX,TX) of the shield to either Software Serial(D8,D7) or Hardware Serial(D1,D0) of the Arduino.

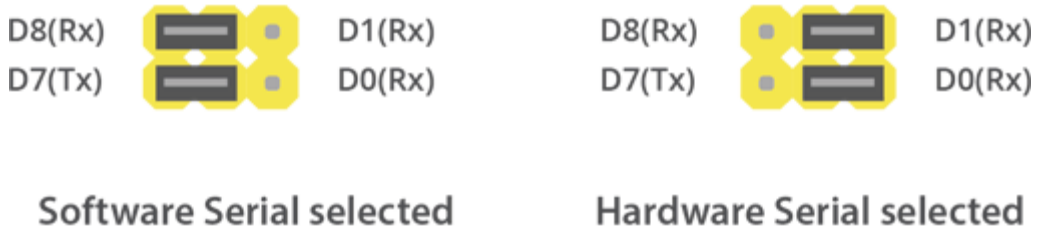
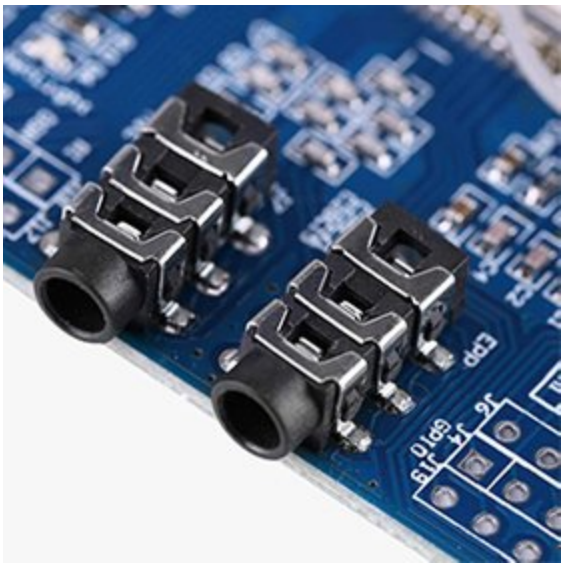


Fig 2.38

Speaker & Microphone

The shield comes with two standard 3.5mm jacks. One for stereo earphone and other for mono microphones. It allows you to use SIM900's audio interface to make and receive voice calls and listen FM radio.



Mic: You can connect an external electret microphone to this jack.

Earphone: You can connect earphones to this jack. Any 'iPhone' or 'Android' compatible earphones should work.

Antenna

An antenna is required to use the SIM900 for any kind of voice or data communications as well as some SIM commands.

Fig 2.39

The shield has two interfaces for connecting antenna viz. a U.FL connector and a SMA connector. They are connected through a patch cord.

The shield usually comes with a 3dBi GSM antenna and allows you to put the shield inside a metal case(as long the antenna is outside).

SIM Socket

There's a SIM socket on the back. Any activated, 2G full-size SIM card would work perfectly.

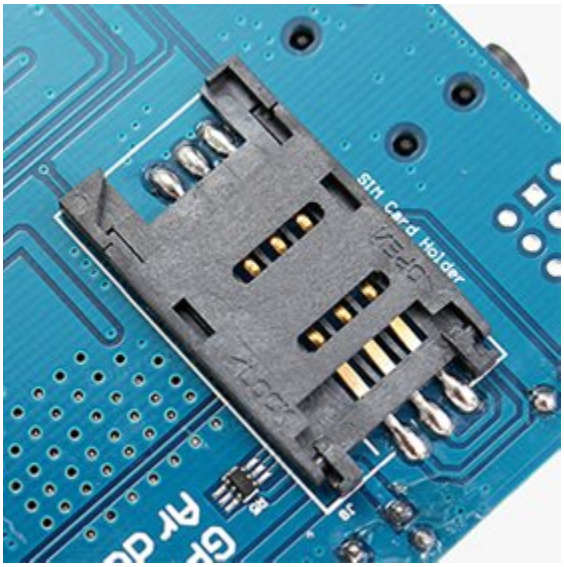
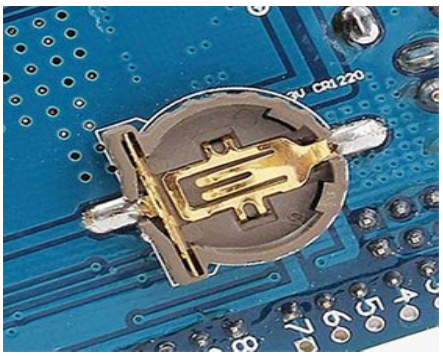


Fig 2.40

The workings of the SIM card socket can take some getting used to. To unlock the latch, push the top part of the assembly, and then lift it up. Place the SIM card into the bottom part of the socket. Then fold the arm back into the body of the socket, and gently push it forward towards the LOCK position.

]RTC(Real Time Clock)

The SIM900 shield can be configured to keep time. So there is no need for any separate RTC.



This will keep the time even when the power is OFF.

Fig 2.41

2.16. HC-05 - Bluetooth Module

HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for **transparent wireless serial connection setup**. Its communication is via serial communication which makes an easy way to interface with a controller or PC. Moreover, The **HC-05** is a popular bluetooth module which can add two-way (full-duplex) wireless functionality to your projects.

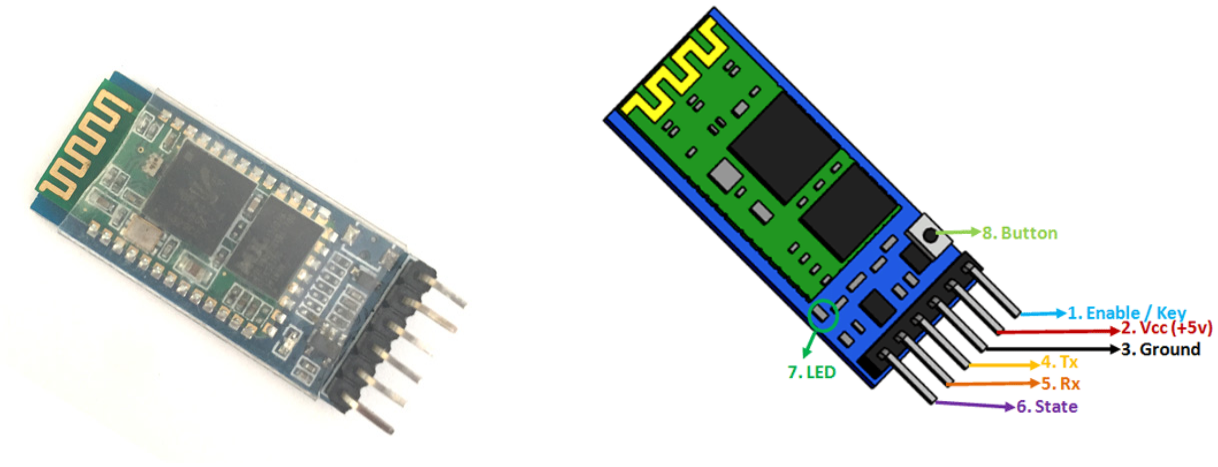


Fig 2.42

As show on the **Fig 2.42,** It has 6 pins,

1. Key/EN: It is used to bring Bluetooth modules in AT commands mode. If the Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is 38400bps and 9600 in data mode.

HC-05 module has two modes,

1. Data mode: Exchange of data between devices.

2. Command mode: It uses AT commands which are used to change settings of HC-05. To send these commands to the module serial (USART) port is used.

2. VCC: Connect 5 V or 3.3 V to this Pin.

3. GND: Ground Pin of module.

4. TXD: Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)

5. RXD: Receive data serially (received data will be transmitted wirelessly by Bluetooth module).

6. State: It tells whether a module is connected or not.

How to Use the HC-05 Bluetooth module ?

The HC-05 has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the AT Command mode where the default device settings can be changed. We can operate the device in either of these two modes by using the key pin as explained in the pin description.

It is very easy to pair the HC-05 module with microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU as shown in the figure below

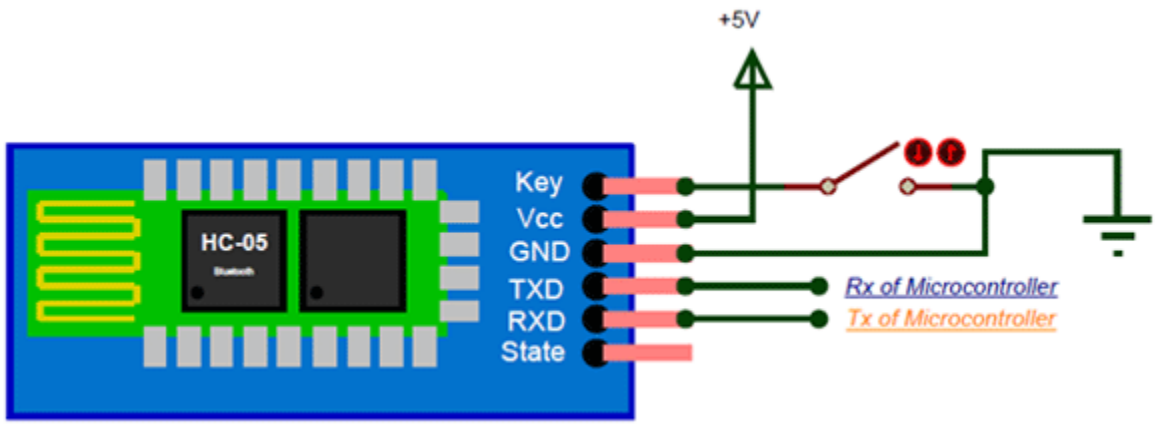


Fig 2.43

During power up the key pin can be grounded to enter into Command mode, if left free it will by default enter into the data mode. As soon as the module is powered you should be able to discover the Bluetooth device as “HC-05” then connect with it using the default password 1234 and start communicating with it.

3.Arduino Projects

3.1. Beginner

3.1.1. Simple Led Blink

In this project, we'll show how to make a simple LED blink that can serve as a springboard for our next project. We'll see an LED turning on and off.

We will be using the following equipments

- Arduino Uno
- 1 Led
- 2 Jumper wires
- Resister (220 ohms)

Circuit Design

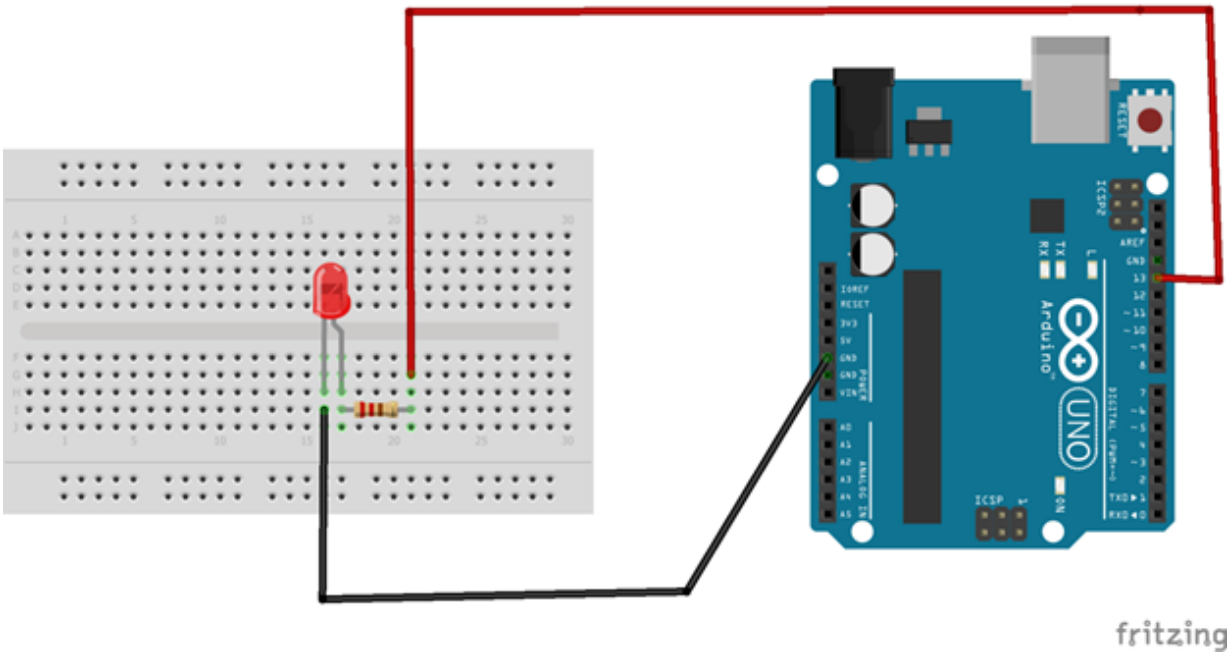


Fig 3.1

As shown above in the circuit design we used LEDs which have 2 pins: the short one is connected to the ground and the long one is connected to pin 13 with 220 ohms resistor.(description about all equipment used are mentioned in the above section).

One thing to remember in using wires is to always respect the color code: use black for ground and red for phase. This habit will help you in finding bugs as you work on more complex projects.

Code

```
#define LED 13 //The pin the LED is connected to

void setup() {

 pinMode(LED,OUTPUT);//make LED an output

 }

void loop() {

 digitalWrite(LED,HIGH);//Turn the LED on

 delay(1000);//1000 millisecond(1s) delay before going to the next line

 digitalWrite(LED,LOW);//Turn the LED off

 delay(1000);//1000 millisecond(1s) delay before going to the next line

}
```

As shown above in the code section we used **#define** it is used to set pins in this case we defined LED to be in pin 13 and in the **setup()** section the code involved are executed once at the start of the program in this program, it will configure the pin that connects to the LED as an output, because you will be telling the pin to do something, instead of querying the pin to determine its state so **pinMode()** is used to configure the direction one thing to notice here is if we don't explicitly tell Arduino what to do it will take all pins as inputs so we have to use **pinMode()** to tell the Arduino if we have output to show that is what happened here. The second required function in all Arduino programs is void loop(). The contents of the loop function repeat forever as long as the Arduino is on. If you want your Arduino to do something once

at boot only, you still need to include the loop function, but you can leave it empty. In this case we used **digitalWrite()** . It is a command that is used to set the state of an output pin. It can set the pin to either 5V or 0V. When an LED is connected to a pin, setting it to 5V will enable you to light up the LED.**Delay()** accepts a single integer or a number as an argument; this number represents the time(measured in milliseconds). The program should wait this amount of time before moving to the next line of code.

3.1.2. Analog Read Serial

In this project we will demonstrate how to read analog input from the physical using a potentiometer. A potentiometer is a simple mechanical device that provides a varying amount of resistance when its shaft is turned(you can find more of this from the equipment section mentioned above).

By passing voltage through a potentiometer and into an analog input on your board, it is possible to measure the amount of resistance produced by a potentiometer as an analog value.

By turning the shaft of the potentiometer, you change the amount of resistance on either side of the wiper, which is connected to the center pin of the potentiometer. This changes the voltage at the center pin. When the resistance between the center and the side connected to 5 volts is close to zero (and the resistance on the other side is close to 10k ohm), the voltage at the center pin nears 5 volts. When the resistances are reversed, the voltage at the center pin nears 0 volts, or ground. This voltage is the analog voltage that you're reading as an input. The Arduino boards have a circuit inside called an analog-to-digital converter or ADC that reads this changing voltage and converts it to a number between 0 and 1023.

For this project you first need to establish a serial communication between your computer and the arduino running the Arduino Software (IDE).

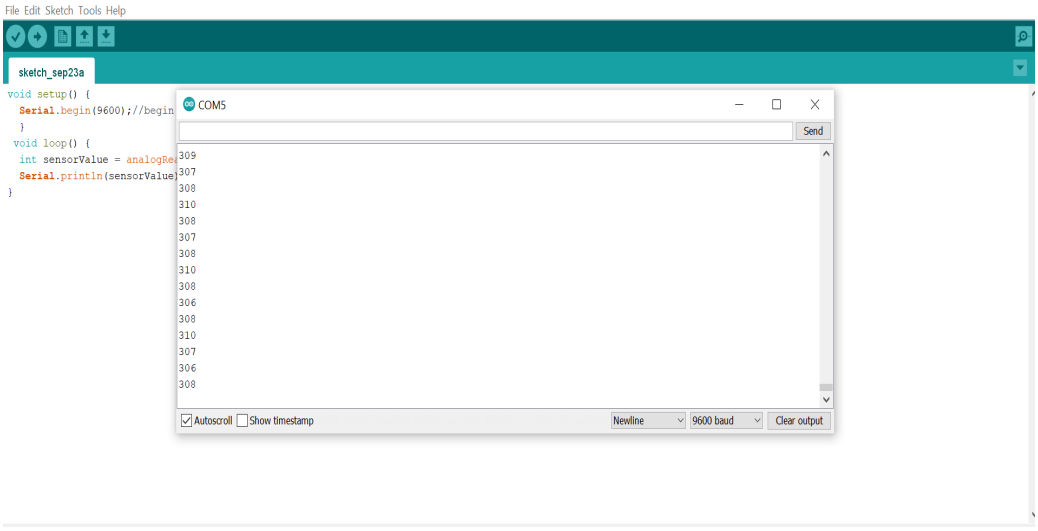
We will be using the following equipments

- Arduino Uno
- Bread Board
- 10k Potentiometer
- Jumper Wires

How to open a Serial Monitor?

In this project you are told to open a serial port but how to do it is a question that can be raised here so to open the monitor you first have to upload your code after you upload your code to the arduino board you have to click the circled button to open it.

So after this step the serial monitor will start based on the rate you gave it in the code and the info you specified in your code will appear.



Circuit Design

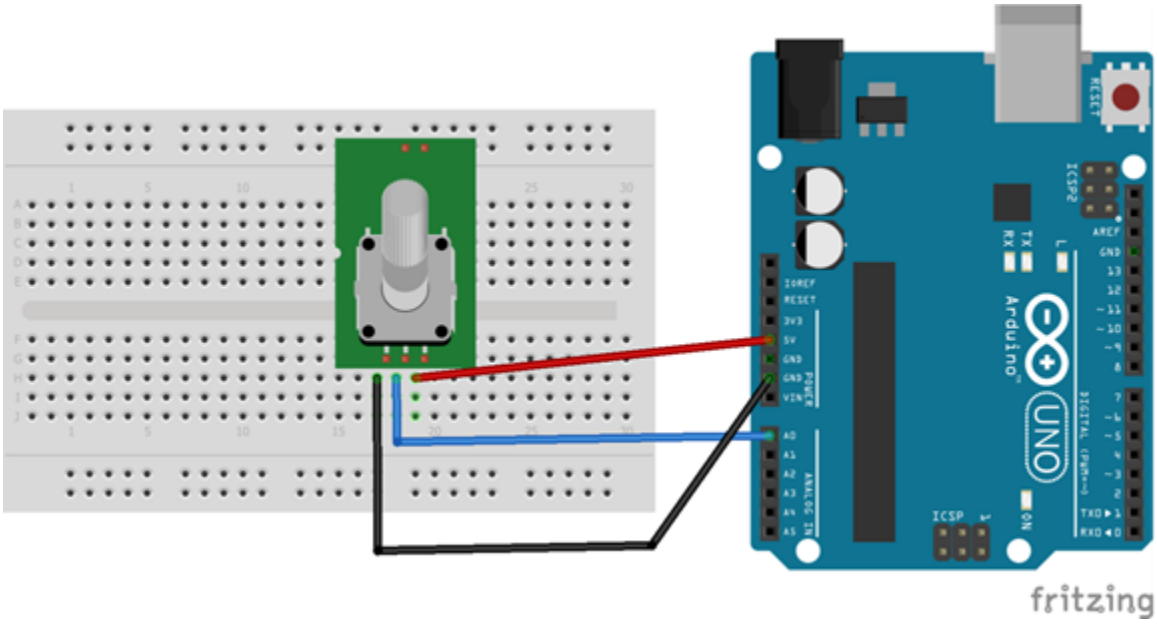


Fig 3.2

As shown above in the circuit design potentiometer has 3 pins Ground , Vcc & Output and the Vcc goes to 5v and the ground pin goes to GND and the middle one will go to the analog pins. In this case we used A0 as an analog pin.

Code

```
void setup() {

 Serial.begin(9600);//begin serial communications at 9600 bits of data

 }

 void loop() {

  int sensorValue = analogRead(A0);//Store the resistance value

  Serial.println(sensorValue);// print this to your serial monitor window
```

```
}
```

As shown above in the code section we used 2 functions which are **setup()** and **loop()** .The serial interface to the computer must be started in the **setup()** function. **Serial.begin()** takes one argument that specifies the communication speed, or baud rate. The baud rate specifies the number of bits being transferred per second. Faster baud rates enable you to transmit more data in less time, but can also introduce transmission errors in some communication systems. A common value is 9600 baud, which is what you will use throughout our document here. In each iteration through the loop, the senserValue variable is set to the present value that the ADC (analog to digital converter) reports from analog pin 0. The **analogRead()** command requires the number of the ADC pin to be passed to it. After the value has been read (a number between 0 and 1023), **Serial.println()** prints that value over serial to the computer's serial terminal.we will pass senserValue as an argument in the **Serial.println(senserValue)**.

3.1.3. Digital Read Serial

In this project we will monitor the state of a switch by establishing a serial communication between your Arduino and your Computer.

In this project we will be using a push button. Push Buttons or switches connect two points in a circuit when you press them(refer to the equipment section above for more). When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and reads as LOW, or 0. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that the pin reads as HIGH, or 1.

We will be using the following equipments

- Arduino Uno
- Push Button
- 10k ohm resistor
- Wires
- Breadboard

Circuit Design

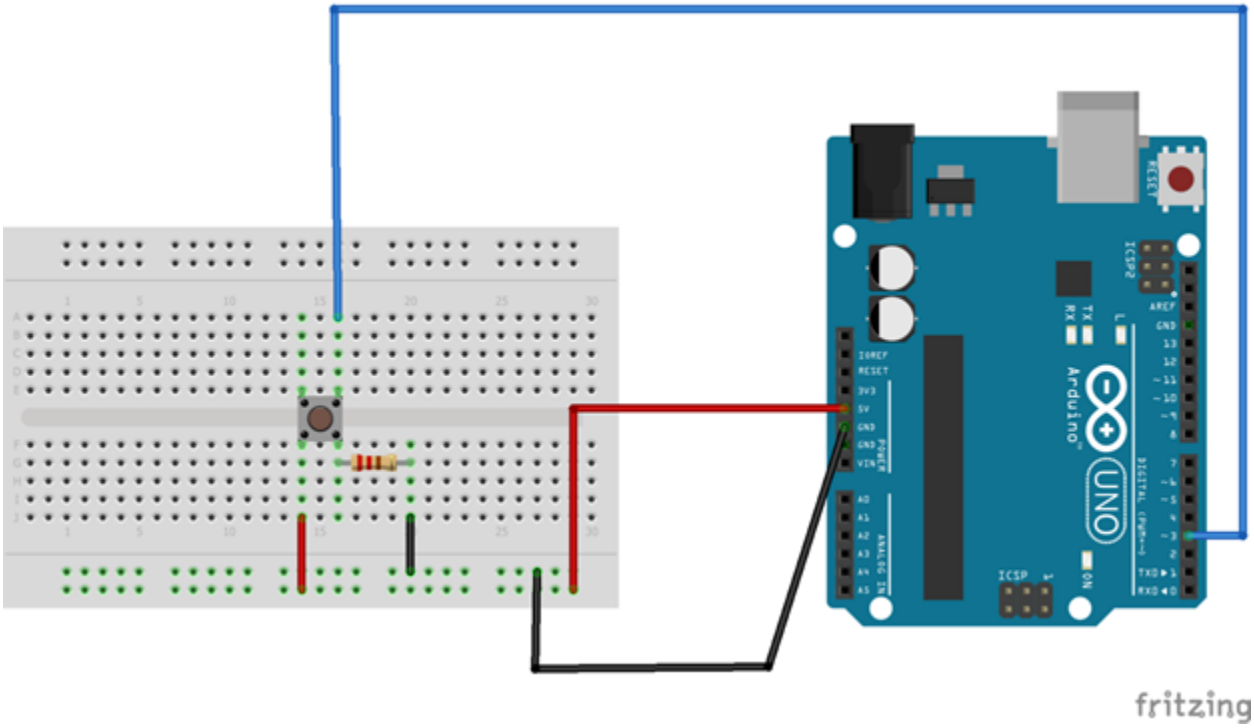


Fig 3.3

As shown above in the circuit design the push button has 2 pairs of pins: the 1st pair the Vcc and the ground is connected to it and the other one we connected a digital pin to it in our case we connected digital pin 2.

Code

```
int pushButton =2; // 2 is digital pin we name it pushButton

void setup() {

  Serial.begin(9600);// begin serial communications at 9600 bits of data

  pinMode(pushButton, INPUT);// make the pushButton as input

}

void loop() {
```

```
 int buttonState = digitalRead(pushButton); //read the input pin

 Serial.println(buttonState);// print this information to your serial monitor window

 delay(1);

}
```

As shown above in the code section we set variable pushButton to represent pin 2 .In the **setup()** section **Serial.begin()** takes one argument that specifies the communication speed, or baud rate. . The baud rate specifies the number of bits being transferred per second. pinMode() here is used to configure the direction of the pin we set pushButton as an output. In the loop section we used the **digitalRead()** function which is used to read the logic state at a pin. It is capable of telling whether the voltage at this pin is high (~ 5V) or low (~ 0V). **Serial.println()** prints that value over serial to the computer's serial terminal.we will pass buttonState as an argument in the **Serial.println(buttonState)**.

3.1.4. Led Fade

In this project we will demonstrate the fading of an LED. We will be using pulse width modulation turning a digital pin on and off very quickly with different ratios between on and off, to create a fading effect.For this project we will be using an LED that has 2 legs longer(anode) and shorter(cathode) legs connecting the anode leg to the 5v and cathode leg to the ground pin.

In order to fade your LED off and on, gradually increase your PWM(described in the above section) value from 0 (all the way off) to 255 (all the way on), and then back to 0 once again to complete the cycle. In the sketch below, the PWM value is set using a variable called brightness. Each time through the loop, it increases by the value of the variable fadeAmount.

We will be using the following equipments

- Arduino Uno
- LED
- 220 ohm resistor
- Wires
- Breadboard

Circuit Design

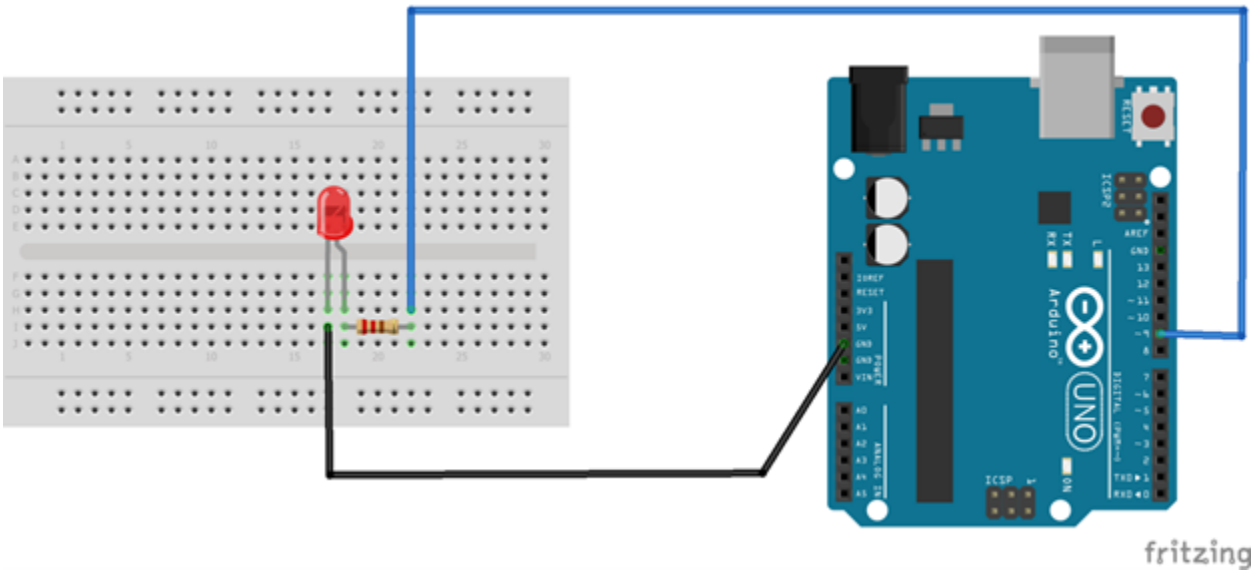


Fig 3.4

As shown above in the circuit design connect the cathode led to the ground pin and the anode leg to pin 9 which is an analog pin.

Code

```
int led = 9;           // the PWM pin the LED is attached to

int brightness = 0;            // how bright the LED is

int fadeAmount = 5;          // how many points to fade the LED by

void setup() {

 pinMode(led, OUTPUT);

}

void loop() {

  analogWrite(led, brightness);// set the brightness of pin 9:
```

```
 brightness = brightness + fadeAmount;// change the brightness for next time      through the loop:

 // reverse the direction of the fading at the ends of the fade:

if (brightness <= 0 || brightness >= 255) {

          fadeAmount = -fadeAmount;

 }

 // wait for 30 milliseconds to see the dimming effect

 delay(30);

}
```

First we defined some variables like led, brightness and fadeAmount and set the pin. In the **setup()** section we set the **pinMode()** of the led to be **OUTPUT** as we know by now **pinMode()** is used to configure the direction of the pin if it is either input or output.

In the **loop()** section The **analogWrite()** function that you will be using in the main loop of your code requires two arguments: One telling the function which pin to write to, and one indicating what PWM value to write. **Delay()** accepts a single integer or a number as an argument; this number represents the time(measured in milliseconds). The program should wait this amount of time before moving to the next line of code.

3.1.5. Buzzer

In this project we will demonstrate a buzzer(arduino buzzer is also called a piezo buzzer. It is basically a tiny speaker that you can connect directly to an Arduino) making sounds. In using buzzers there is one to notice there are two types of buzzers: active and passive buzzers The passive buzzer is an electromagnetic squeaker used to generate sound signals of different frequencies. The active buzzer is the simplest module to produce a sound of about 2 kHz, which can often be needed when working with Arduino and in other projects.

We will be using the following equipments

- Arduino Uno
- Jumper Wire
- Breadboard
- Buzzer

Circuit Design

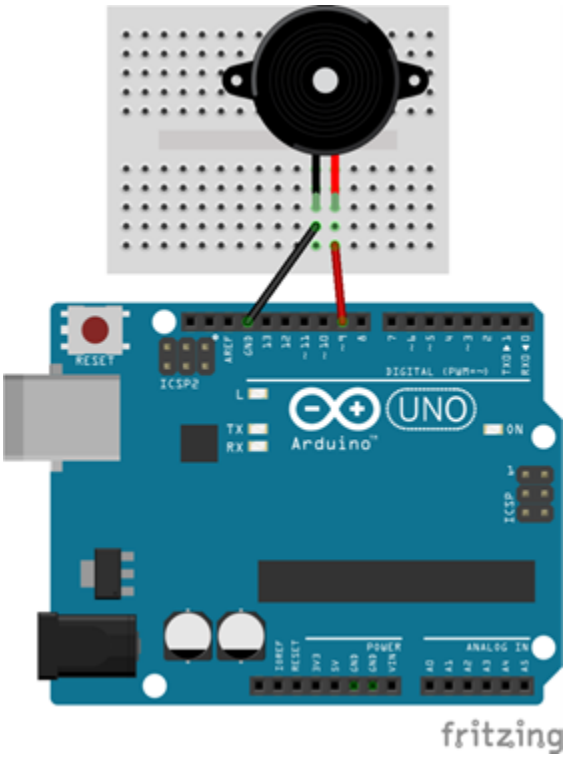


Fig 3.5

As shown above in the circuit design we need a Buzzer and connect its positive side to pin 9 and negative pin to the ground.

Code

```
const int buzzerpin = 9;//define the buzzer pin

void setup() {

 pinMode(buzzerpin,OUTPUT); //set the buzzerpin as an output pin

}

void loop() {

 for(int i=200;i<=800;i++)//loops from 200 to 800 (frequency)

 {

 tone(buzzerpin,i);//Generates a square wave of the specified frequency (and 50% duty cycle) on a pin

 delay(5);

 }

 delay(3000);//3 second delay on the highest frequency

 for(int i=800;i>=200;i--)

 {

  tone(buzzerpin,i);//loops from 200 to 800 (frequency)

 delay(5);

 }

}
```

First we defined our buzzerpin and set the pin number 9. In the **setup()** section we set the **pinMode()** of the buzzerpin to be **OUTPUT** as we know by now **pinMode()** is used to configure the direction of the pin if it is either input or output.

In the **loop()** section we need a for loop to range different frequency rates in this case from 200 to 800. **tone(pin,frequency)** Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to **noTone(). delay()** accepts a single integer or a number as an argument; this number represents the time(measured in milliseconds). The program should wait this amount of time before moving to the next line of code.

3.1.6. Simple position indicator

In this project we will demonstrate a simple position indicator project using an ultrasonic sensor.this sensor is a module which uses ultrasound It allows to detect a presence or to measure a distance via ultrasounds.It sends an ultrasound pulse via its trig terminals and receives them via his echo terminals simply 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.In this project we will use it to measure how close something is by approaching the ultrasonic sensor.

We will be using the following equipments

- Arduino Uno
- Jumper Wire
- Breadboard
- Ultrasonic Sensor

Circuit Design

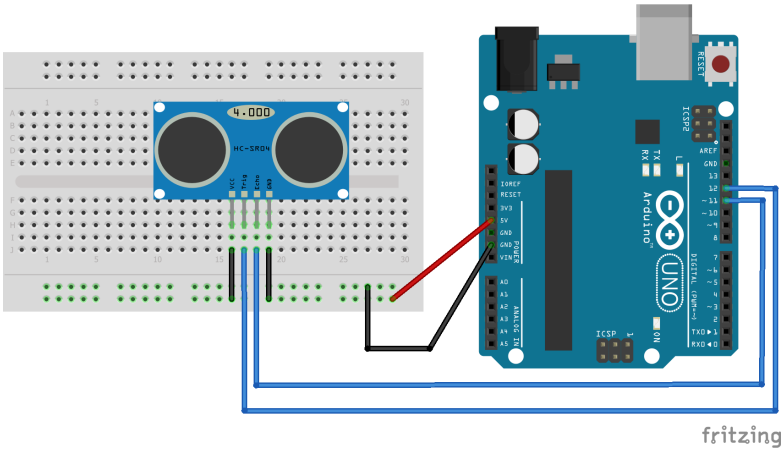


Fig 3.6

As shown above in the circuit design we used an Ultrasonic. This sensor has 4 pins : vcc ,ground ,echo & trig so we connect the vcc to 5v Gnd to ground and echo to pin 11 and trig to pin 12 these pins are defined in the code below.In using an Ultrasonic sensor use male to female jumper wire to connect to the board and arduino.

Code

```
int trigPin = 12;    // Trigger
int echoPin = 11;    // Echo
long duration, cm, inches;
 void setup() {
  //Serial Port begin
  Serial.begin (9600);
  //Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

```
void loop() {
 // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
 // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
 digitalWrite(trigPin, LOW);
 delayMicroseconds(5);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
 // Read the signal from the sensor: a HIGH pulse whose
 // duration is the time (in microseconds) from the sending
 // of the ping to the reception of its echo off of an object.
 duration = pulseIn(echoPin, HIGH);
// Convert the time into a distance
 cm = (duration/2) / 29.1;     // Divide by 29.1 or multiply by 0.0343
 inches = (duration/2) / 74;   // Divide by 74 or multiply by 0.0135
 Serial.print(inches);
 Serial.print("in, ");
 Serial.print(cm);
 Serial.print("cm");
 Serial.println();
 delay(250);
}
```

First we create variables for trigger and echo trigPin and echoPin, respectively. The trigger pin is connected to digital pin 11, and the echo pin is connected to pin 12 and sets the pinMode for trig and echo as an OUTPUT.

In the loop(), trigger the sensor by sending a HIGH pulse of 10 microseconds. But, before that, give a short LOW pulse to ensure you'll get a clean HIGH pulse.

pulseIn() function to get the sound wave travel time after that we calculated in both cm and inch and print in the serial.

3.1.7. Traffic Light System

In this project you'll build a traffic light system which we usually observe when we cross the road while walking or driving our car. This traffic light system controls both the pedestrians and the vehicle. The vehicle control LEDs are three while the pedestrian control LEDs are two.

Color code of the LED is

Vehicle color code			Pedestrian color code	
Green	GO		Green	GO
Yellow	Ready to GO or STOP		Red	STOP
Red	STOP			

We will be using the following equipments

- Resister
- Led
- Push Button
- Jumper wires

- Arduino Uno

Considering the above color code the ON/OFF pattern of the LEDs is as the following algorithm shown below.

Circuit Design

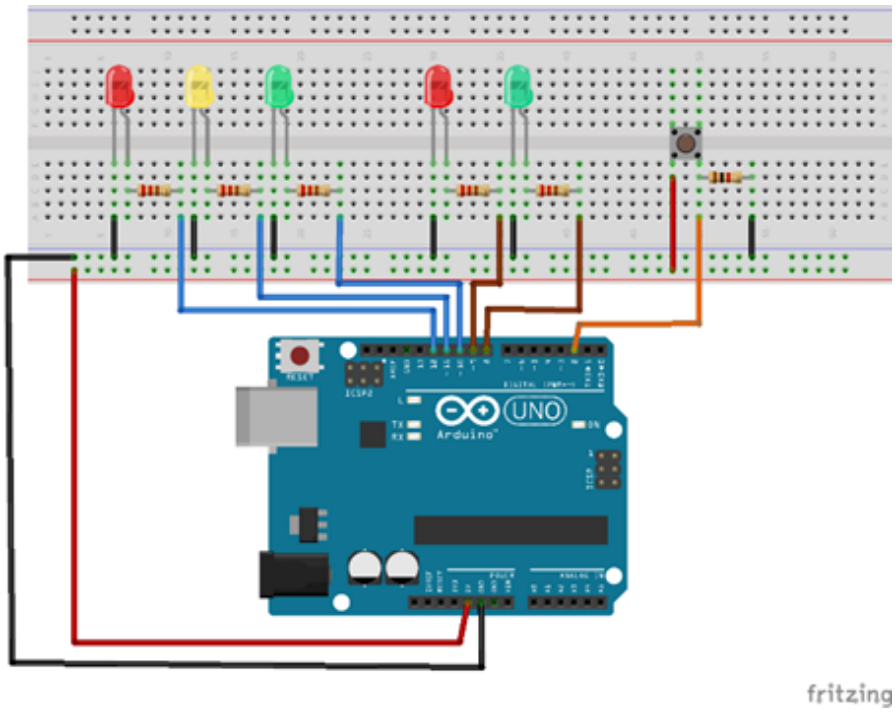


Fig 3.7

Code

```
int pg =  8; // pedestrian GREEN LED
```

```
int pr =  9; // pedestrian RED LED

int cg = 10; // car GREEN LED

int cy = 11; // car YELLOW LED

int cr = 12; // car RED LED

int bt =  2; // Button

void setup(){

        pinMode(pg, OUTPUT);

        pinMode(pr, OUTPUT);

        pinMode(cg, OUTPUT);

        pinMode(cy, OUTPUT);

        pinMode(cr, OUTPUT);

        pinMode(bt, INPUT);

}

void loop()

{

 if(digitalRead(bt) == HIGH){

        delay(2000);

        digitalWrite(pr, HIGH);

        digitalWrite(pg, LOW);

        digitalWrite(cr, LOW);

        digitalWrite(cy, HIGH);

        digitalWrite(cg, HIGH);
```

```
            delay(3000);
            digitalWrite(cr, HIGH);
            digitalWrite(cy, LOW);
            digitalWrite(cg, LOW);
            delay(2000);


            digitalWrite(pr, LOW);
            digitalWrite(pg, HIGH);
            delay(5000);
for(int i=0;i<5;i++){
            digitalWrite(pg, LOW);
             delay(500);
            digitalWrite(pg, HIGH);
            delay(500);
            }
            digitalWrite(pr, HIGH);
            digitalWrite(pg, LOW);
            digitalWrite(cr, LOW);
            digitalWrite(cy, LOW);
            digitalWrite(cg, HIGH);
            delay(2000);
 }
```

```
 else{
        digitalWrite(pr, HIGH);
        digitalWrite(pg, LOW);
        digitalWrite(cr, LOW);
        digitalWrite(cy, LOW);
        digitalWrite(cg, HIGH);}
}
```

3.1.8.Simple servo movement

In this project we will demonstrate a simple servo movement This is an easy project that will teach you how to control servos. Here we will make our servo rotate 90 and rest for 2 seconds and rotate back.

We will be using the following equipments

- Servo motor SG90
- bread board
- Jumper wires
- Arduino Uno

Circuit Design

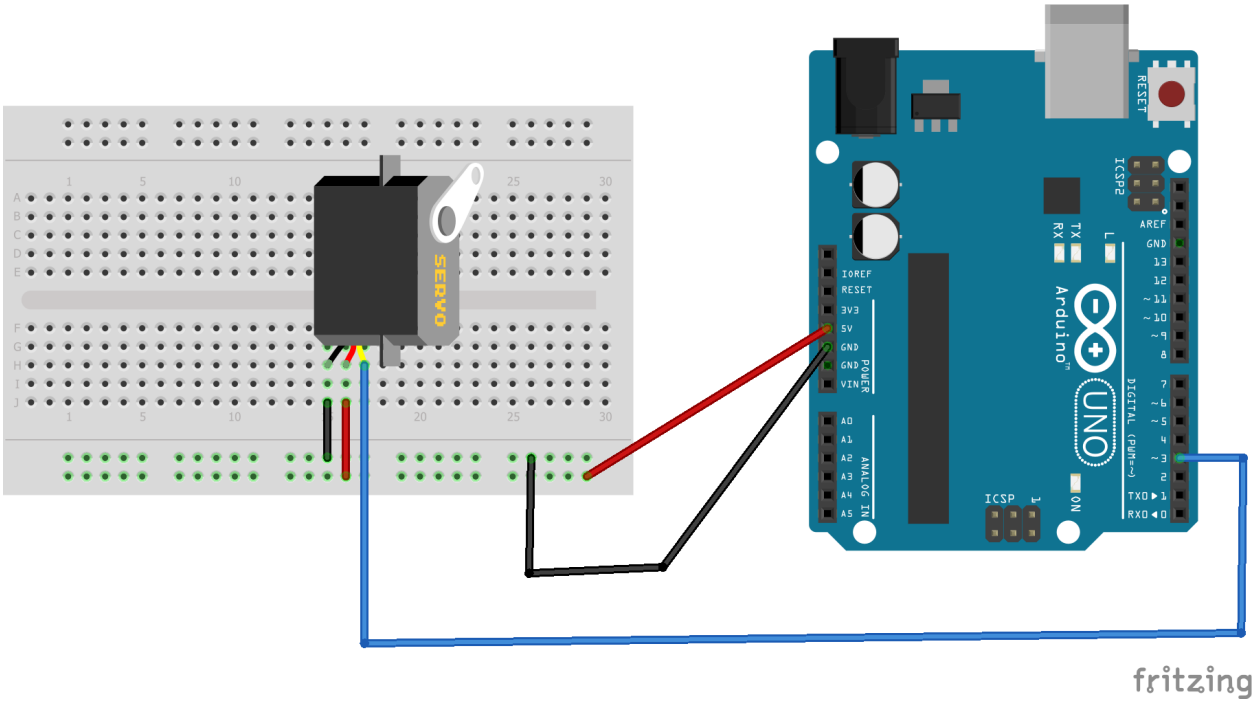


Fig 3.8

As shown above in the circuit design we used a servo motor and it has 3 pins vcc, ground and the other pin will be the pin we defined in our code 3 in our case.

Code

```
#include <Servo.h>//Include servo library

Servo servo; //give object reference

void setup() {

servo.attach(8); //attach pin 8 to servo

servo.write(0);//set servo direction to 0

delay(2000);// delay for 2 second

}

void loop() {
```

```
servo.write(90); //set servo direction to 90

delay(1000);//delay 1 second

servo.write(0);//set servo direction back to 0

delay(1000);//delay 1 second

}
```

3.2. Medium level projects

3.2.1. Measuring Proximity

In this project we will use two of the beginners projects, those are project number 6 which is simple position measurement and project number 1 which is led blink in this project we will be combine the two of them and make something valuable.So in this project you'll be building a circuit that is able measure distance wirelessly with the help of ultrasonic sensor and provides the proximity of the object with LEDs where all the LEDs turn ON when object is very close to ultrasonic sensor and turns OFF while object is far away.

The range of distance versus the amount of LED turned on as shown in the table below

Distance measured	Amt of LEDs turned ON
≤10 cm	4 LEDs
≤20 cm	3LEDs
≤30 cm	2LEDs
≤50 cm	1LED

>50 cm	NONE

We will be using the following equipments

- Ultrasonic sensor
- Led(4)
- Jumper wires
- Arduino Uno
- 220 ohms Resistor(4)

Circuit Design

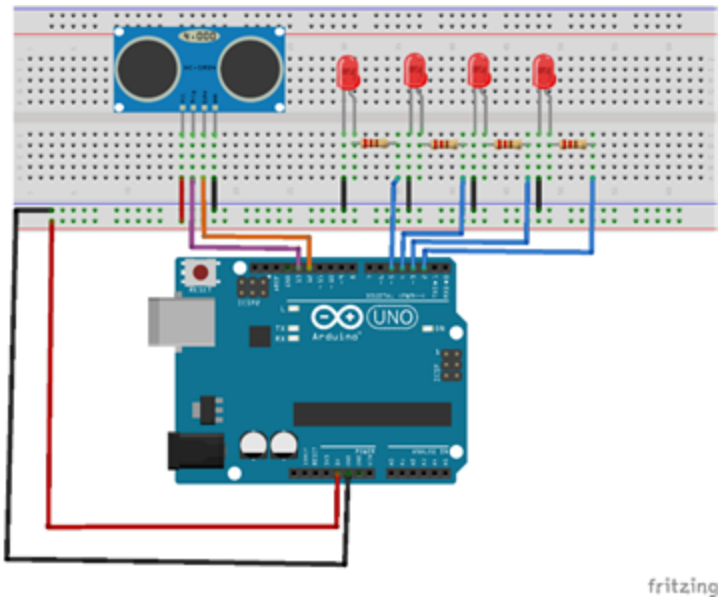


Fig 3.9

Code

```
int trig =13;

int echo = 12;

int LED_1 =2;

int LED_2 =3;

int LED_3 =4;

int LED_4 =5;

void setup() {

  pinMode(trig,OUTPUT);

  pinMode(LED_1,OUTPUT);

  pinMode(LED_2,OUTPUT);

  pinMode(LED_3,OUTPUT);
```

```
  pinMode(LED_4,OUTPUT);
  pinMode(echo,INPUT);
  Serial.begin(9600);
}
void loop() {
    digitalWrite(trig,LOW);
    digitalWrite(trig,HIGH);
    delayMicroseconds(20);
    digitalWrite(trig,LOW);
    int distance = pulseIn(echo,HIGH) * 340/2/10000;
    Serial.println(distance);
    delay(100);
    if (distance <= 10){
      digitalWrite(LED_1,HIGH);
      digitalWrite(LED_2,HIGH);
      digitalWrite(LED_3,HIGH);
      digitalWrite(LED_4,HIGH);
     }
    else if (distance <= 20){
      digitalWrite(LED_1,HIGH);
      digitalWrite(LED_2,HIGH);
      digitalWrite(LED_3,HIGH);
```

```
      digitalWrite(LED_4,LOW);
     }
   else if (distance <= 30){
      digitalWrite(LED_1,HIGH);
      digitalWrite(LED_2,HIGH);
      digitalWrite(LED_3,LOW);
      digitalWrite(LED_4,LOW)}
else if (distance <= 50) {
     digitalWrite(LED_1,HIGH);
     digitalWrite(LED_2,LOW);
     digitalWrite(LED_3,LOW);
     digitalWrite(LED_4,LOW);
   }
  else {
     digitalWrite(LED_1,LOW);
     digitalWrite(LED_2,LOW);
     digitalWrite(LED_3,LOW);
     digitalWrite(LED_4,LOW);
   }}
```

3.2.2. Servo Control

In this project we will use two of the beginners projects,as we know we have demonstrated a simple servo movement in the earlier beginner projects now we use potentiometer which is dealt in project number 3 and control the speed of the servo so here we will twist it a little bit we combine them two and control the servo using the potentiometer.

In this project we will be using a servo library. This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

We will be using the following equipments

- Servo motor
- Potentiometer
- Jumper wires
- Arduino Uno

Circuit Design

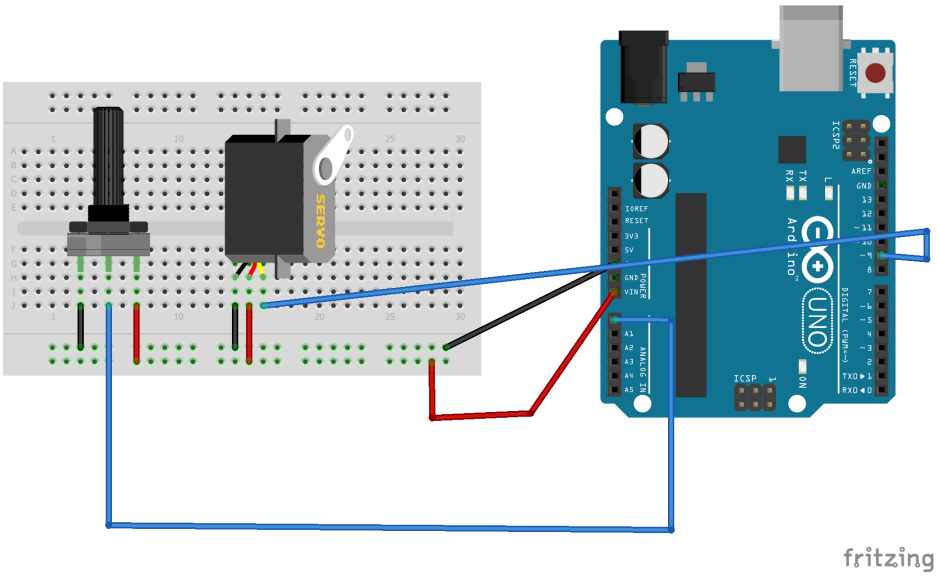


Fig 3.10

Servo motors have three wires: **power**, **ground**, and **signal**. The **power wire** is typically **red**, and should be connected to the 5V pin on the Arduino board. The **ground wire** is typically **black** or **brown** and should be connected to a ground pin on the Arduino board. The **signal** pin is typically **blue** and should be connected to a digital pin on the Arduino board. Note that servos draw considerable power, so if you need to drive more than one or two, you'll probably need to power them from a separate supply just the way we have done it in the previous (Motor Control) experiment. Be sure to connect the grounds of the Arduino and external power supply together.

Code

```
#include<Servo.h>

    Servo myservo;        // create servo object to control  a servo

    int potpin = 0;      //Analog pin connected to the potentiometer

    int val;

void setup(){

    myservo.attach(9);    //attaches the servo on pin9 to the servo object

 }

void loop() {

    val = analogRead(potpin)// reads the value of the potentiometer (value between 0 and 1023

    val = map(val, 0, 1023, 0, 180);   // scale's potentiometer value(0 - 1023) to be used with servo value(0 - 180)

    myservo.write(val);//sets the servo position according to the scaled value

     delay(15);  //waits for the servo to get there

 }
```

3.2.3. Controllable RGB LED Nightlight

In this project we will use RGB LED to make a controllable nightlight. We will use push buttons to change the lights. We have done a project with push buttons in the beginner section and you can find both the description of the RGB and push button on the component section.

In this project, you use a common anode LED. That means that the LED has four leads. One of them is an anode pin that is shared among all three diodes, while the other three pins connect to the cathodes of each diode color. Wire that LED up to three PWM pins through current-limiting resistors on the Arduino, as shown in the circuit diagram below. As with the single red LED, values of 220Ω will work well for current limiting. Because this LED is a common anode, that means the cathode of each diode is being controlled by the Arduino, instead of the anode as in the example with the red LED. When an Arduino's pin is set as an output, it is really doing one of two things.

Therefore, if an LED's anode is connected to 5V, and its cathode is connected to an Arduino pin configured as an output, its logic will be inverted. When you set the pin LOW, that will enable current to flow from 5V, through the resistor and LED, and into the Arduino's current sink. When you set the pin HIGH, it will be at the same (5V) potential as the anode of the LED, so no current will flow and the LED will turn off. You can configure a debounced button to cycle through a selection of colors each time you press it. To do this, it is useful to add an additional function to set the RGB LED to the next state in the color cycle. In the following code below , I have defined a total of seven color states, plus one off state for the LED. Using the analogWrite() function, you can choose your own color-mixing combinations. The only change to the loop() from the previous example is that instead of flipping a single LED state, an LED state counter is incremented each time the button is pressed, and it is reset back to zero when you cycle through all the options. Upload this code to your Arduino connected to the circuit you just built and enjoy your night light. Modify the color states by changing the values of analogWrite() to make your own color options.I challenge you to change the color as you do this project.

We will be using the following equipments

- Push button
- RGB
- Jumper wires
- Resistor

- Arduino Uno

Circuit Design

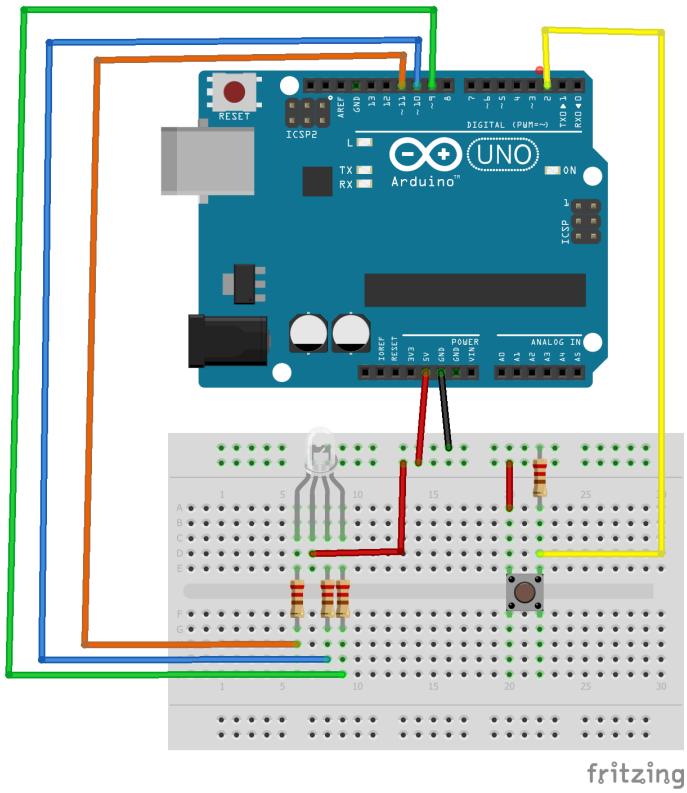


Fig 3.11

Code

```
const int BLED=9; // Blue LED Cathode on Pin 9

const int GLED=10; // Green LED Cathode on Pin 10

const int RLED=11; // Red LED Cathode on Pin 11

const int BUTTON=2; // The Button is connected to pin 2

boolean lastButton = LOW; // Last Button State
```

```
boolean currentButton = LOW; // Current Button State
int ledMode = 0; // Cycle between LED states
void setup()
{
 pinMode (BLED, OUTPUT); // Set Blue LED as Output
 pinMode (GLED, OUTPUT); // Set Green LED as Output
 pinMode (RLED, OUTPUT); // Set Red LED as Output
 pinMode (BUTTON, INPUT); // Set button as input (not required)
}
/*
* Debouncing Function
* Pass it the previous button state,
* and get back the current debounced button state.
*/
boolean debounce(boolean last)
{
 boolean current = digitalRead(BUTTON); // Read the button state
 if (last != current) // If it's different...
 {
 delay(5); // Wait 5ms
 current = digitalRead(BUTTON); // Read it again
 }
```

```
 return current; // Return the current value
}
/*
* LED Mode Selection
* Pass a number for the LED state and set it accordingly
* Note, since the RGB LED is COMMON ANODE, you must set the
* cathode pin for each color LOW for that color to turn ON.
*/
void setMode(int mode)
{
 //RED
 if (mode == 1)
 {
 digitalWrite(RLED, LOW);
 digitalWrite(GLED, HIGH);
 digitalWrite(BLED, HIGH);
 }
 //GREEN
 else if (mode == 2)
 {
 digitalWrite(RLED, HIGH);
 digitalWrite(GLED, LOW);
```

```
digitalWrite(BLED, HIGH);
}
//BLUE
else if (mode == 3)
{
digitalWrite(RLED, HIGH);
digitalWrite(GLED, HIGH);
digitalWrite(BLED, LOW);
}
//PURPLE (RED+BLUE)
else if (mode == 4)
{
analogWrite(RLED, 127);
analogWrite(GLED, 255);
analogWrite(BLED, 127);
}
//TEAL (BLUE+GREEN)
else if (mode == 5)
{
analogWrite(RLED, 255);
analogWrite(GLED, 127);
analogWrite(BLED, 127);
```

```
}
//ORANGE (GREEN+RED)
else if (mode == 6)
{
analogWrite(RLED, 127);
analogWrite(GLED, 127);
analogWrite(BLED, 255);
}
//WHITE (GREEN+RED+BLUE)
else if (mode == 7)
{
analogWrite(RLED, 170);
analogWrite(GLED, 170);
analogWrite(BLED, 170);
}
//OFF (mode = 0)
else
{
digitalWrite(RLED, LOW);
digitalWrite(GLED, LOW);
digitalWrite(BLED, LOW);
}
```

```
}

void loop()

{

 currentButton = debounce(lastButton); // Read debounced state

 if (lastButton == LOW && currentButton == HIGH) // If it was pressed...

 {

 ledMode++; // Increment the LED value

 }

 lastButton = currentButton; // Reset button value

 // If you've cycled through the different options,

 // reset the counter to 0

 if (ledMode == 8) ledMode = 0;

 setMode(ledMode); // Change the LED state

}
```

NB

Please beware the length of the code doesn't mean it's difficult, all the functions used here are covered in the beginner project just try it.it's very simple.

3.2.4. Light variation detecting

In this project we will use a photoresistor. This thing is an electronic component whose resistivity varies according to the amount of light received (the resistance decreases when exposed to the light). They are sensitive to light. Its resistance decreases when lighting increases.

A photoresistor is a semiconductor material which is used here to determine the amount of light received that can be used for different projects.

We will be using the following equipments

- Photoresistor
- Jumper wires
- Resistor
- Arduino Uno

Circuit Design

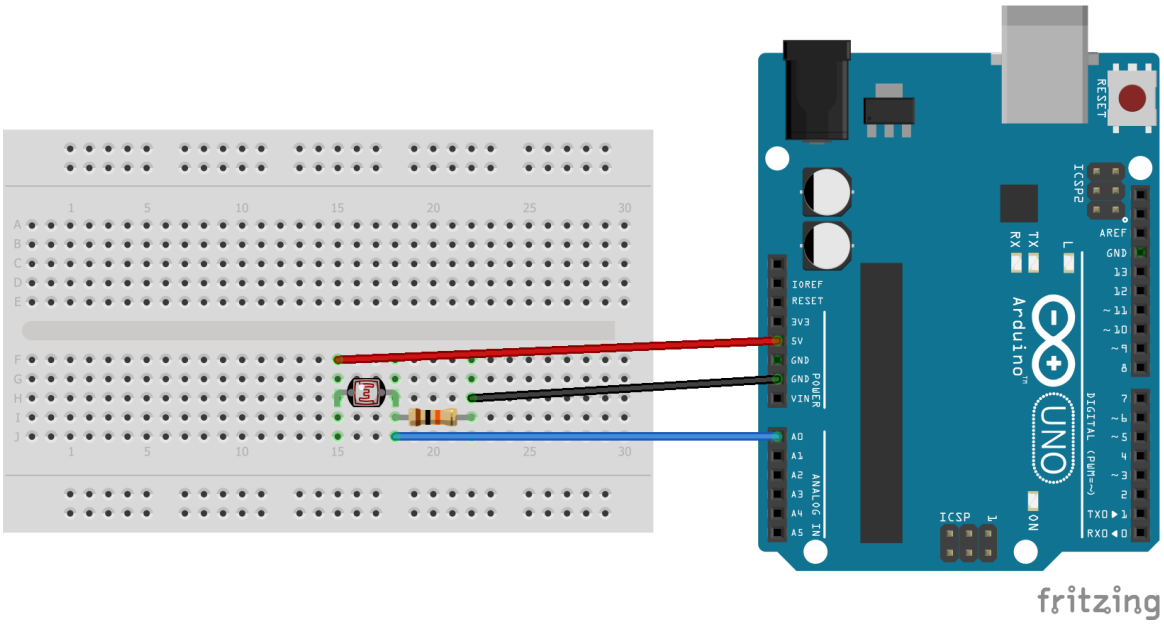


Fig 3.12

As shown above in the design we are expected to connect the one leg of the photoresistor to 5v and the other to the analog input A0 and 10kohm resistor end to the ground.

Code

```
 void setup() {

Serial.begin(9600);//strat the serial port with band rate of 9600

}

void loop() {

int value = analogRead(A0); //read the analog value and store it in the int variable 'value'

Serial.println("Analog Value :");//print Analog value:

Serial.println(value); // print the value read above

delay(250);//delay by 0.25 second before repeating the loop

}
```

This code is general and fundamental; all the descriptions of these functions are listed in previous projects.

3.2.5. Lcd display

In this project we will use Lcd 16x2 to display the message “4k Labs” . This LCD 16x2 is a 16-pin device that has 2 rows that can accommodate 16 characters each. LCD 16x2 can be used in 4-bit mode or 8-bit mode. It is also possible to create custom characters. It has 8 data lines and 3 control lines that can be used for control purposes.

The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.Here we will display the message and the second also so we will use 1 line for each as clearly described above 16x2 lcd display has 2 lines for display we will use the 1st to display our message the second to display the second since it first started.

We will be using the following equipments

- Potentiometer

- Lcd display 16x2
- Jumper wires
- Resistor
- Arduino Uno

Circuit Design

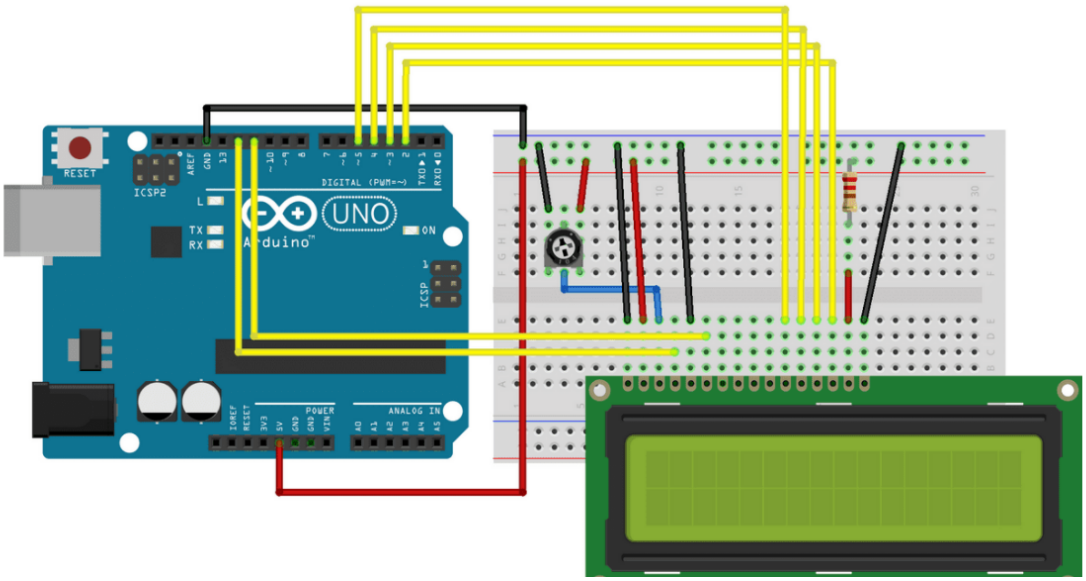


Fig 3.13

Pins of Lcd 16x2

A **register select (RS) pin** that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

A **Read/Write (R/W) pin** that selects reading mode or writing mode

An **Enable pin** that enables writing to the registers

8 **data pins (D0 -D7)**. The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a **display contrast pin (Vo)**, **power supply pins (+5V and Gnd)** and **LED Backlight (Bklt+ and BKlt-)** pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

Please be aware of the circuit connection here. The design is a bit confusing so use this description to do the connection.

Vss - Ground

Vdd - 5v

Rs - pin 12

Rw - Ground

E - pin 11

D4 - pin 5

D5 - pin 4

D6 - pin 3

D7 - pin 2

A - one end of the resistor

V0- middle pin of the potentiometer

one end of the potentiometer to Ground

The other end to 5v

k - Ground

Code

```
// include the library code:

#include <LiquidCrystal.h>
```

```
// with the arduino pin number it is connected to

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

  // set up the LCD's number of columns and rows:

  lcd.begin(16, 2);

  // Print a message to the LCD.

  lcd.print("4k Labs");

}

void loop() {

  // set the cursor to column 0, line 1

  // (note: line 1 is the second row, since counting begins with 0):

  lcd.setCursor(0, 1);

  // print the number of seconds since reset:

  lcd.print(millis() / 1000);

}
```

if you did the circuit connection , you will see this output.

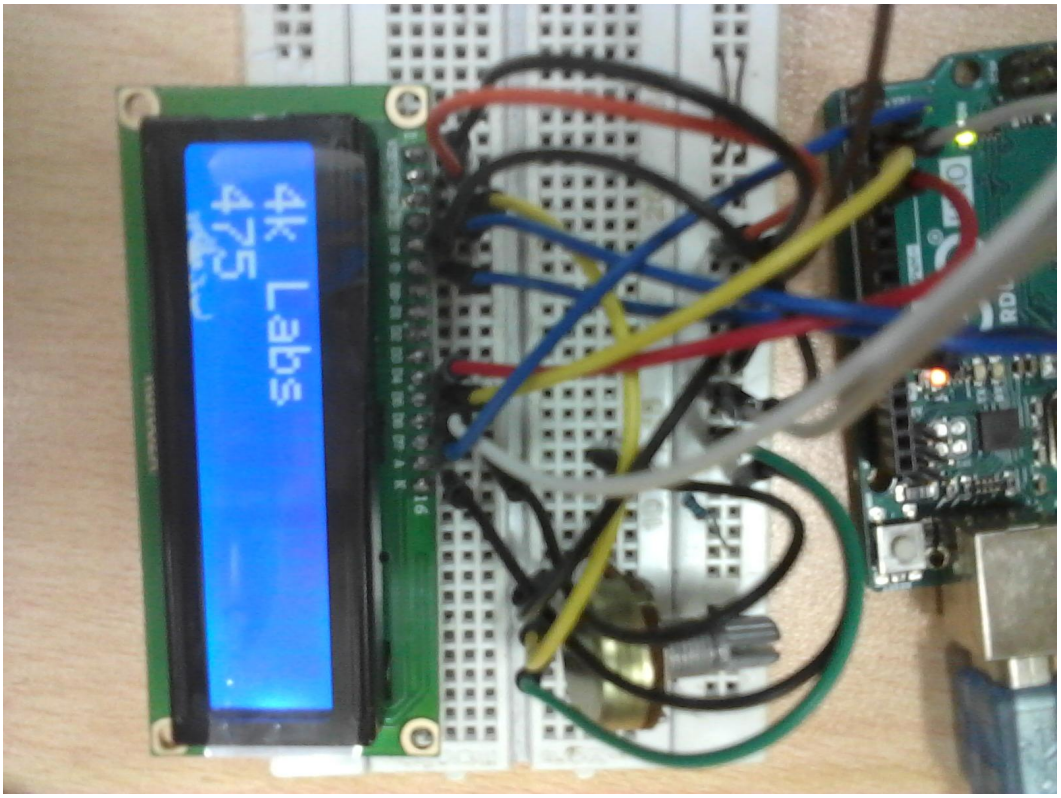


Fig 3.14

3.2.6. Humidity Sensor

In this project we will use DHT -11 .It is a digital-output, relative humidity, and temperature sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and sends a digital signal on the data pin.

In this project, you will learn how to use this sensor with Arduino UNO. The room temperature and humidity will be printed to the serial monitor.

We will be using the following equipments

- DHT-11
- Jumper wires
- Breadboard
- Arduino Uno

Circuit Design

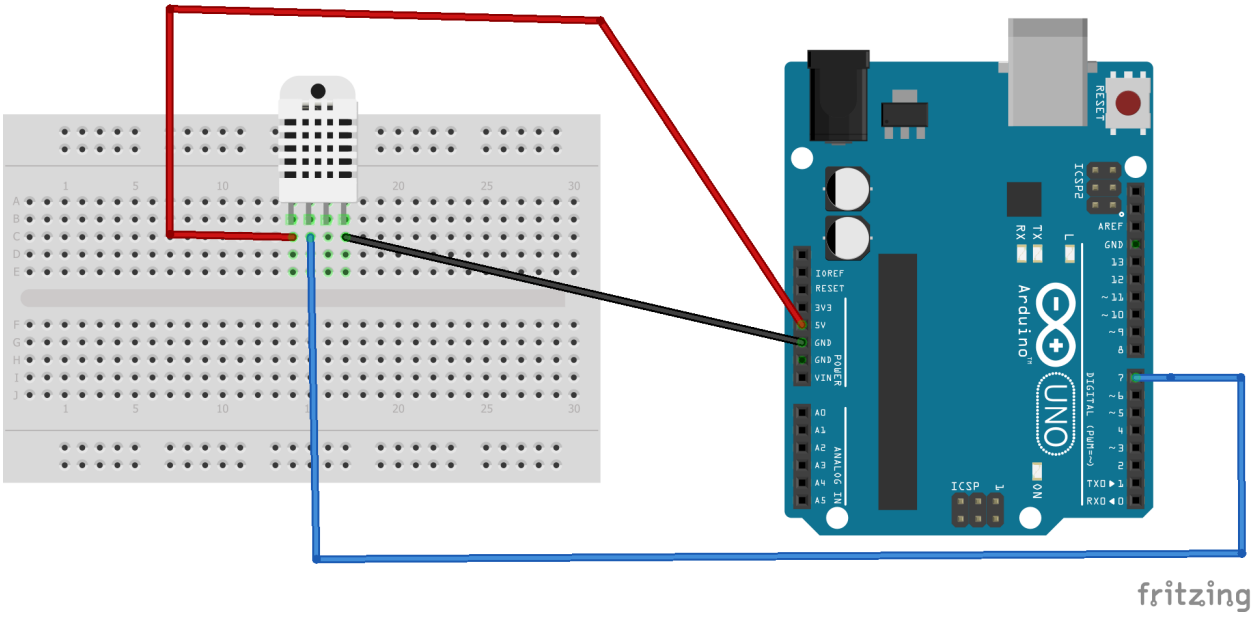


Fig 3.15

DHT-11 have 4 pins the 1st pin is for Vcc the 2nd pin goes to data the 3rd pin is ignored(not used) the 4th pin is for ground.so connect the circuit as shown above and upload the code below.

Code

```
// initialize the library by associating any needed LCD interface pin

#include <dht.h>

dht DHT;

#define DHT11_PIN 7

void setup(){

  Serial.begin(9600);//start the serial port at 9600 band
```

```
}

void loop(){

  int chk = DHT.read11(DHT11_PIN);//read temp value

  Serial.print("Temperature = ");//print Temperature = in the serial

  Serial.println(DHT.temperature);//print the value of temperature to the serial

  Serial.print("Humidity = ");//print Humidity =  in the serial

  Serial.println(DHT.humidity);//print the value of humidity to the serial

  delay(1000);//delay 1 second

}
```

please download the library from the drive link below in your ide go to Tools> Sketch library> Add .ZIP file and select the zip file you just downloaded and then just upload your code.

download here

3.2.6. Keypad

In this project we will use an arduino keypad 4x4. This keypad is a set of arranged buttons in rows and columns called matrix where each button is called key.There are various types of keypad but here we will be using keypad 4x4 which have 16 keys.

We will be using the following equipments

- Keypad 4x4
- Jumper wires
- Arduino Uno

Circuit Design

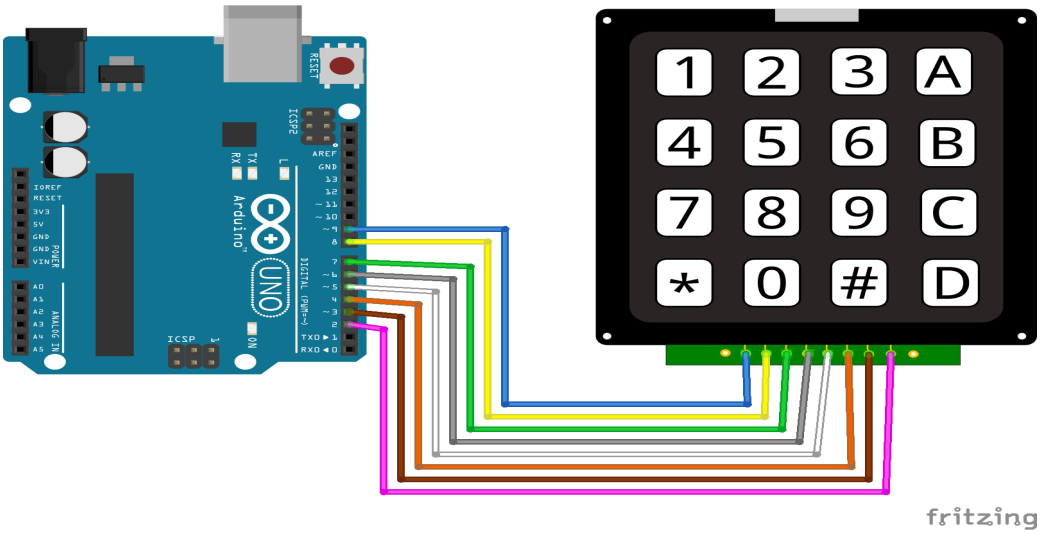


Fig 3.16

keypad pins are classified into two groups those are row and column in our case keypad 4x4 have 8 pins 4 row pins (R1,R2,R3,R4) and 4 column pins (C1,C2,C3,C4) so here pin 9,8,7,6 are the row pins and pin 5,4,3,2 are the column pins.

Code

```
#include <Keypad.h>

const int ROW_NUM = 4; //four rows

const int COLUMN_NUM = 4; //four columns

char keys[ROW_NUM][COLUMN_NUM] = {

  {'1','2','3', 'A'},

  {'4','5','6', 'B'},

  {'7','8','9', 'C'},
```

```
  {'*','0','#', 'D'}

};

byte pin_rows[ROW_NUM] = {9, 8, 7, 6}; //connect to the row pinouts of the keypad

byte pin_column[COLUMN_NUM] = {5, 4, 3, 2}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), pin_rows, pin_column, ROW_NUM, COLUMN_NUM );

void setup(){

  Serial.begin(9600);

}

void loop(){

  char key = keypad.getKey();

if (key){

    Serial.println(key);}}
```

please download the keypad library from the drive link below in your ide go to Tools> Sketch library> Add .ZIP file and select the zip file you just downloaded and then just upload your code.

download here

3.3. Advanced

3.3.1. Making the Arduino uno in breadboard

In this project we will be making an arduino in board so till now we have done 15 basic arduino project but if we think on making a prototype of our ideas we need to use this method it is not necessary to always use the arduino uno we can build that in the breadboard by just using the following equipments.

We will be using the following equipments

- AtMega 328
- 16MHZ Crystal

- 220PF,10MF&100NF capacitors
- 10k ohm resistor
- Bread board
- Arduino Uno(for burning the bootloader)
- Jumper cables

Circuit Design

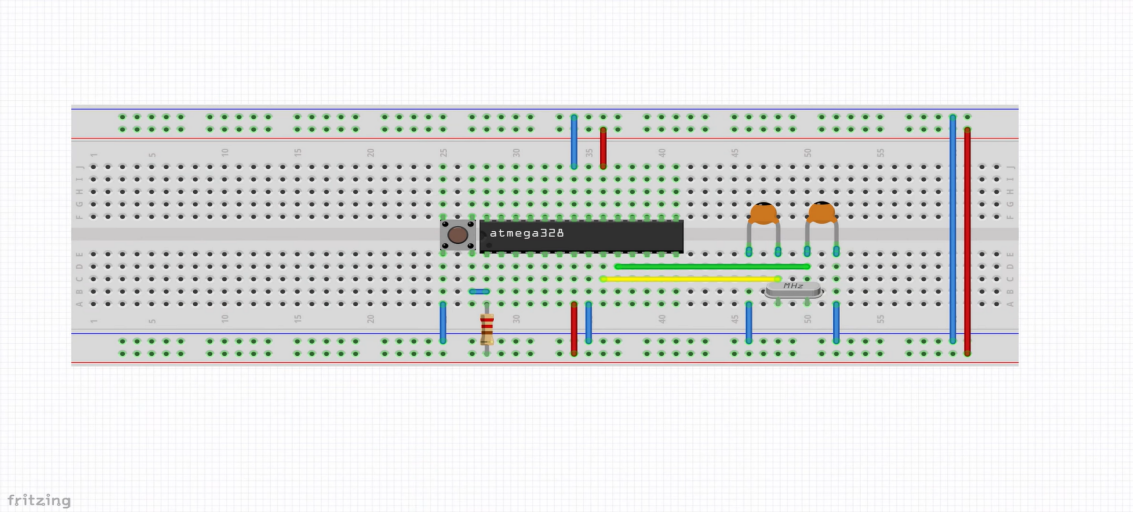


Fig 3.17

so this circuit can replace arduino uno how it is going to replace it is described below.

The tag switch is used as the reset button when we want to upload new code to the microcontroller and the 10k ohm resistor acts as a pullup to prevent atmega from resetting.

crystal acts as clock for our microcontroller

If errors happen by voltage filtration please add 2 capacitors 10MF and 100NF that are connected to VCC and GND to resolve this issue.

Bootloader

As we start working with Atmega 328p we have to know about bootloader because the microcontrollers need to have it in order to work so what is a bootloader A bootloader is a vendor-proprietary image responsible for bringing up the kernel on a device. It guards the device state and is responsible for initializing the Trusted Execution Environment and binding its root of trust. So the bootloader needs to be burned in every Atmega 328p before working on it. This led us to the question of how to burn the bootloader to the microcontroller.

How to burn bootloader

So to burn the bootloader first connect the Atmega 328 as shown above and you need an arduino uno wire it up with our previously done circuit like the image below.

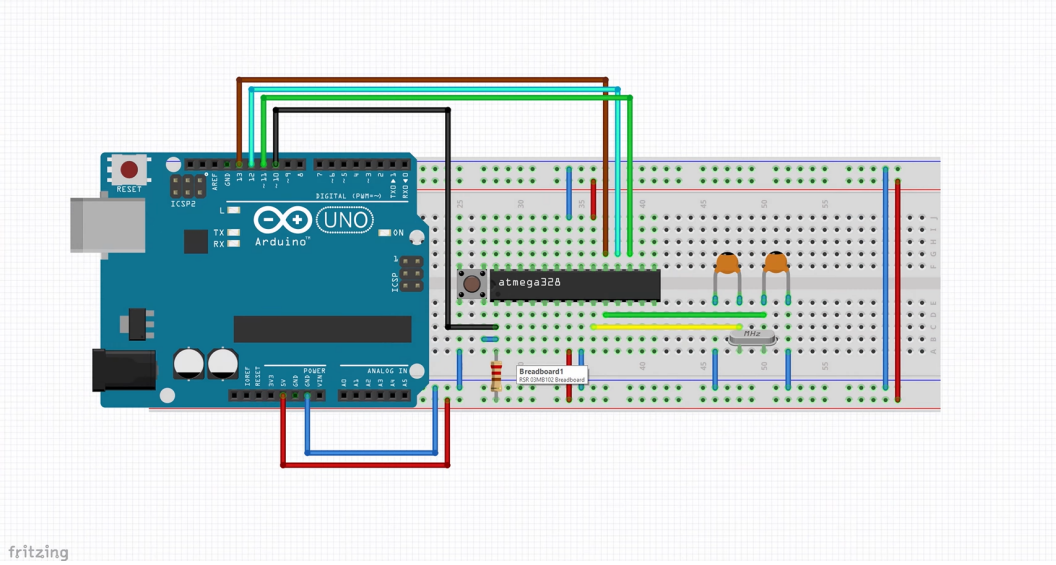


Fig 3.18

After you make this connection what you have to do is go to the arduino ide and go to File > Example > ArduinoISP open it and touch the upload button.

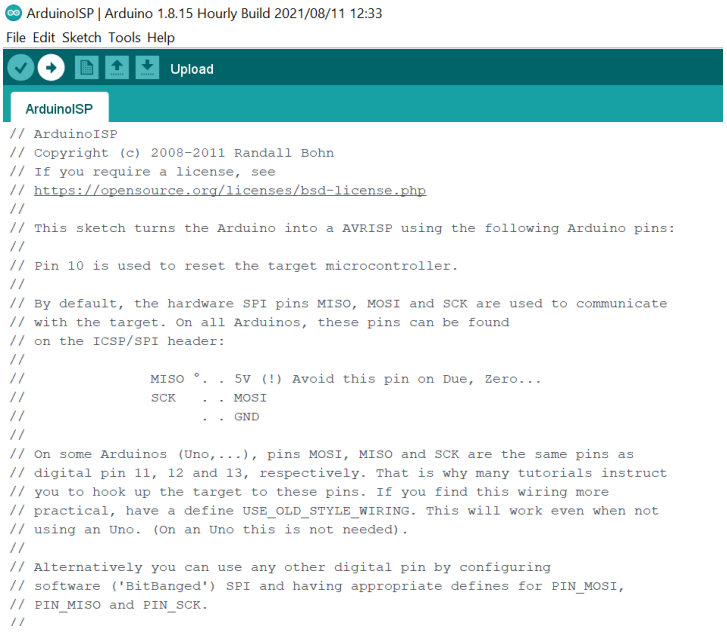


Fig 3.19

After you upload this code to the arduino uno then go to Tools > select Board(port) select arduino uno and again in Tools > select programmer and pick Arduino as Isp after you do this you are almost done.

Final step is go to Tools > select Burn bootloader

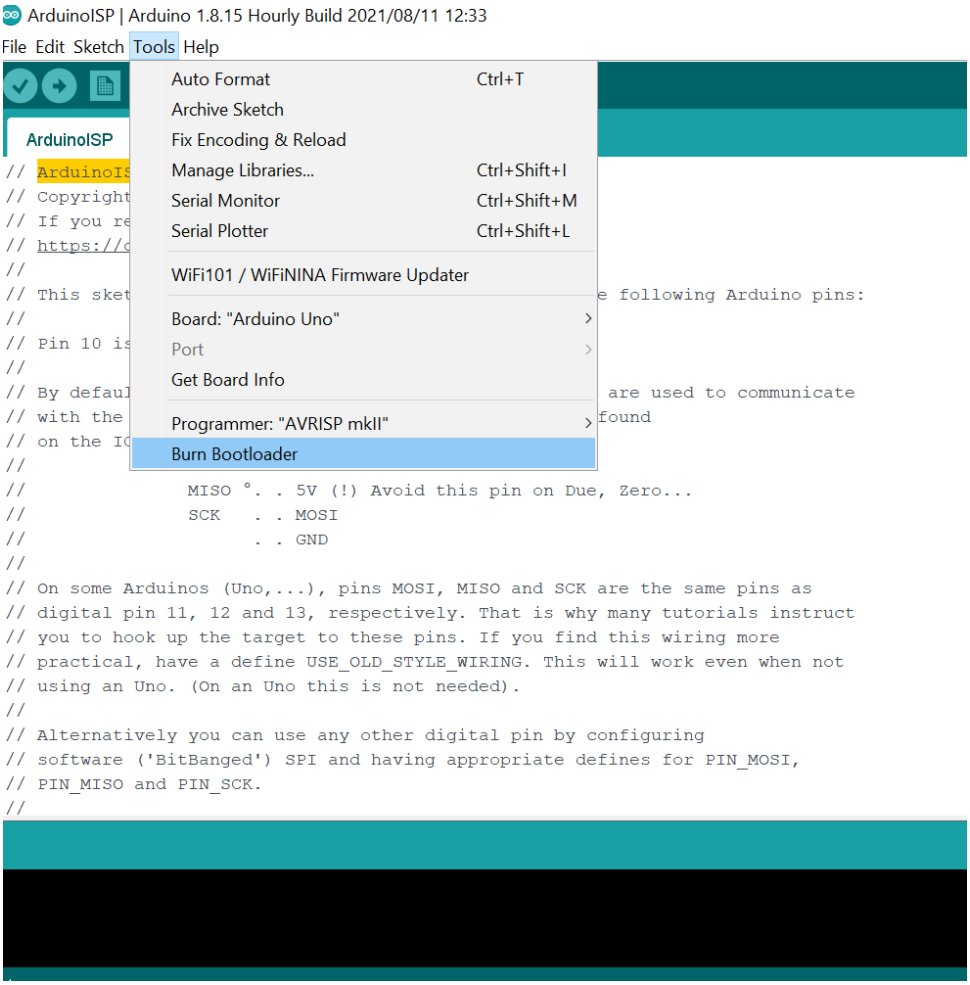


Fig 3.20

It will take a minute to burn then it will finish after it is done you have an arduino uno in the breadboard.

3.3.2. Home security system

In this project we will be working on a GSM based home security system here using PIR Sensor, Arduino Uno, and GSM module “SIM900A”. The PIR sensor is used for Intruder detection. Whenever an unauthorized person is detected a message is sent on the desired number. A relay module is also connected with the Arduino Uno which can be used to turn on an Alarm or it can be used to turn on any light as an indication that an intruder is detected.

SIM900 GSM/GPRS shield is a GSM modem, which can be integrated into a great number of IoT projects. You can use this shield to accomplish almost anything a normal cell phone can; SMS text messages, Make or receive phone calls, connecting to the internet through GPRS, TCP/IP, and more! To top it off, the shield supports quad-band GSM/GPRS network, meaning it works pretty much anywhere in the world.

The PIR Motion sensor module is an automatic control module based on infrared technology. It adopts the LHI788 probe, which has high sensitivity, high reliability, low voltage working mode and low power consumption. It can be widely used in various types of automatic induction electrical equipment.

We will be using the following equipments

- PIR Motion Sensor
- 16MHZ Crystal
- Arduino GSM shield
- Buzzer
- Arduino Uno
- Jumper cables

Circuit Design

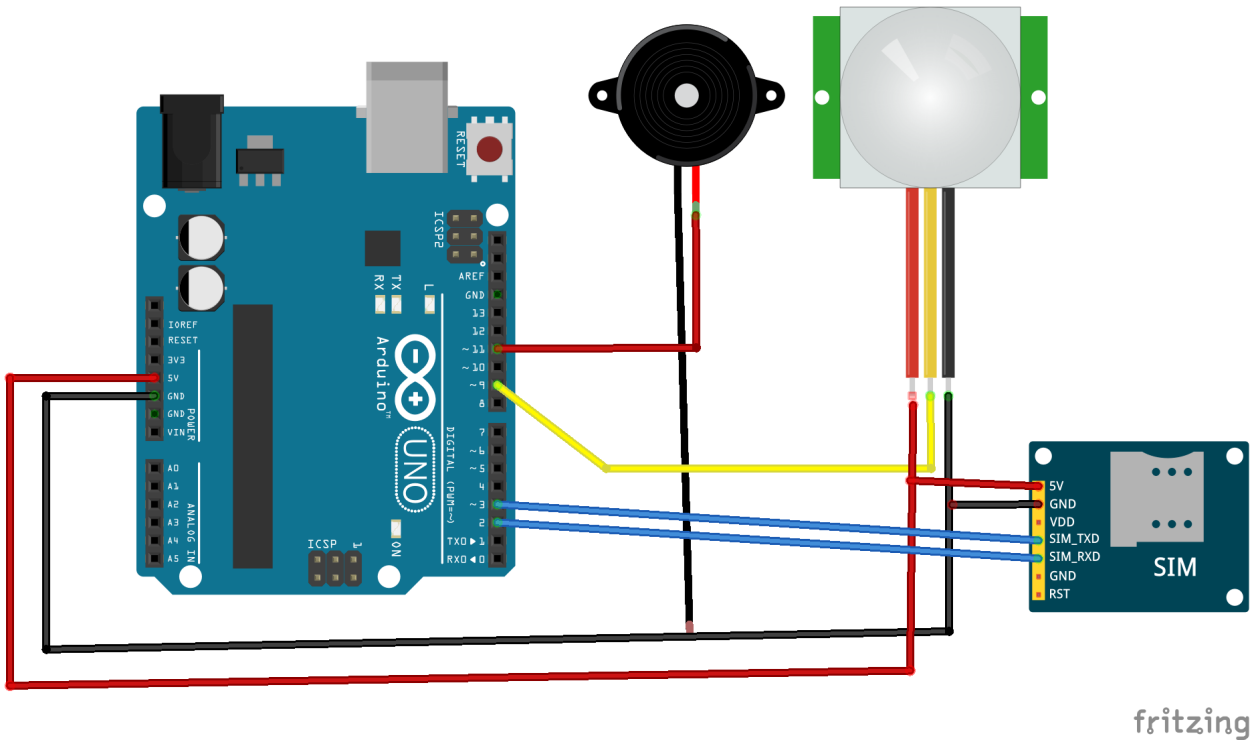


Fig 3.21

Please connect this circuit design based on the connection given connect the buzzer to pin 11 and ground pin and the pir sensor to 5v to pin 9 and ground pins and finally the GSM sim 900A connect 5v , ground and pin 2 to Tx pin 3 to Rx.

Code

```
#include <SoftwareSerial.h>

SoftwareSerial SIM900(2, 3);

String textForSMS;


int PirSensor = 10;

int buzzer = 9;
```

```
int red = 7;
int green = 8;
void setup() {
  randomSeed(analogRead(0));
  Serial.begin(9600);
  SIM900.begin(9600); // original 19200. while enter 9600 for sim900A
  Serial.println(" logging time completed!");
  pinMode(PirSensor, INPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  digitalWrite(buzzer, LOW);
  digitalWrite(red, LOW);
  digitalWrite(green, LOW);
  delay(100);
}
void loop() {
 if ( digitalRead(PirSensor) == HIGH) //
  {
    textForSMS =  "\Any Person in your Room  Plz Check It ";
    digitalWrite(buzzer, HIGH);
    digitalWrite(red, HIGH);
    digitalWrite(green, LOW);
    sendSMS(textForSMS);
    Serial.println(textForSMS);
    Serial.println("message sent.");
    delay(8000);
  }
```

```
  if ( digitalRead(PirSensor) == LOW) //
  {
    digitalWrite(buzzer, LOW);
    digitalWrite(red, LOW);
    digitalWrite(green, HIGH);
    delay(1000);
  }

}
void sendSMS(String message)
{
  SIM900.print("AT+CMGF=1\r");   // AT command to send SMS message
  delay(1000);
  SIM900.println("AT + CMGS = \"+251967259678\"");// recipient's mobile number, in international format
  delay(1000);
  SIM900.println(message);       // message to send

  SIM900.println((char)26);      // End AT command with a ^Z, ASCII code 26
  delay(1000);
  SIM900.println();
  // give module time to send SMS
}
```

3.3.3. Bluetooth controlled car

In this project we will be building a mobile application controlled car here we will not be building the application we will be using a builtin application to control the car so for this project we will be using a bluetooth module which is explained well in the above component section.

In this project we will be using the Hc-05 bluetooth module which is an easy to use bluetooth serial port protocol module, designed for transparent wireless serial connection setup. We need this because we want to send commands from the phone to our car so that we can control the car from our phone.

The other main equipment we will be using here are the parts of the car which are the wheels, the gear motors and the frame of the car. This can be taken from a previously made toy car or you can build your own.

Motor Driver Shield is also used because it allows us to use Arduino to control the working speed and direction of the motor

We will be using the following equipments

- Motor Driver L298n
- Hc-05 Bluetooth Module
- Car wheels
- 2 Gear motors
- Arduino Uno
- Jumper cables
- Battery

Circuit Design

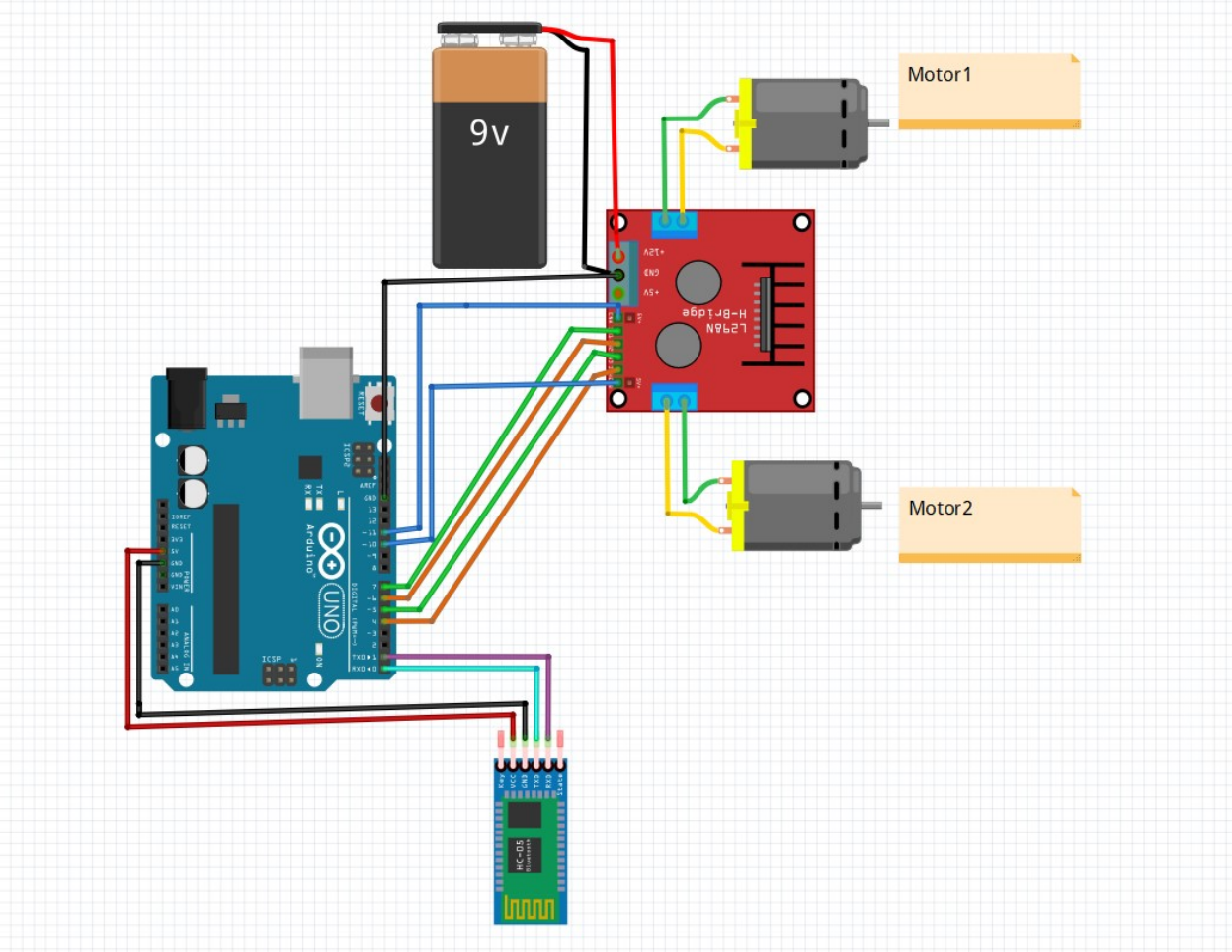


Fig 3.21

The following circuit connection is used when we want to power the Arduino and the driver separately for better performance and speed .

As shown in the figure connect each gear motors to the motor driver based on the numbering given there m1 and m2 after this connect the HC-05 bluetooth module pins with ground , 5v ,Tx and Rx after connecting this the last thing is connecting the battery.Please beware in connecting Tx and Rx you have to upload the code before connecting them and you have to connect the Tx pin of the HC-05 to the Rx pin of the arduino uno and the Rx pin of the HC-05 to the Tx pin of the arduino uno.

Code

```
// Motor 1 connections, which controls the right and left movement

int enA = 11;

int in1 = 7;

int in2 = 6;


// Motor 2 connections , which controls the backward and forward movement


int enB = 10;

int in3 = 4;

int in4 = 3;

char command;

void setup() {

  // Set all the motor control pins to outputs

  pinMode(enA, OUTPUT);

  pinMode(enB, OUTPUT);

  pinMode(in1, OUTPUT);

  pinMode(in2, OUTPUT);

  pinMode(in3, OUTPUT);

  pinMode(in4, OUTPUT);


  // Turn off motors - Initial state
```

```
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
void loop() {
   if(Serial.available() > 0){
   command = Serial.read();
   stop(); //initialize with motors stopped
  //Change pin mode only if the new command is different from the previous.
  //Serial.println(command);
   switch(command){
   case 'F':
     forward();
     break;
   case 'B':
      backward();
     break;
   case 'L':
     left();
     break;
```

```
    case 'R':
      right();
      break;
    } }
}

// This function lets you control spinning direction of motors

void forward() {
  // Set motors to maximum speed
  // For PWM maximum possible values are 0 to 255
    analogWrite(enB, 255);
  // Turn on motor 1
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
 void backward()
{
  // Set motors to maximum speed
  // For PWM maximum possible values are 0 to 255
  analogWrite(enB, 255);

// Turn on motor 1
```

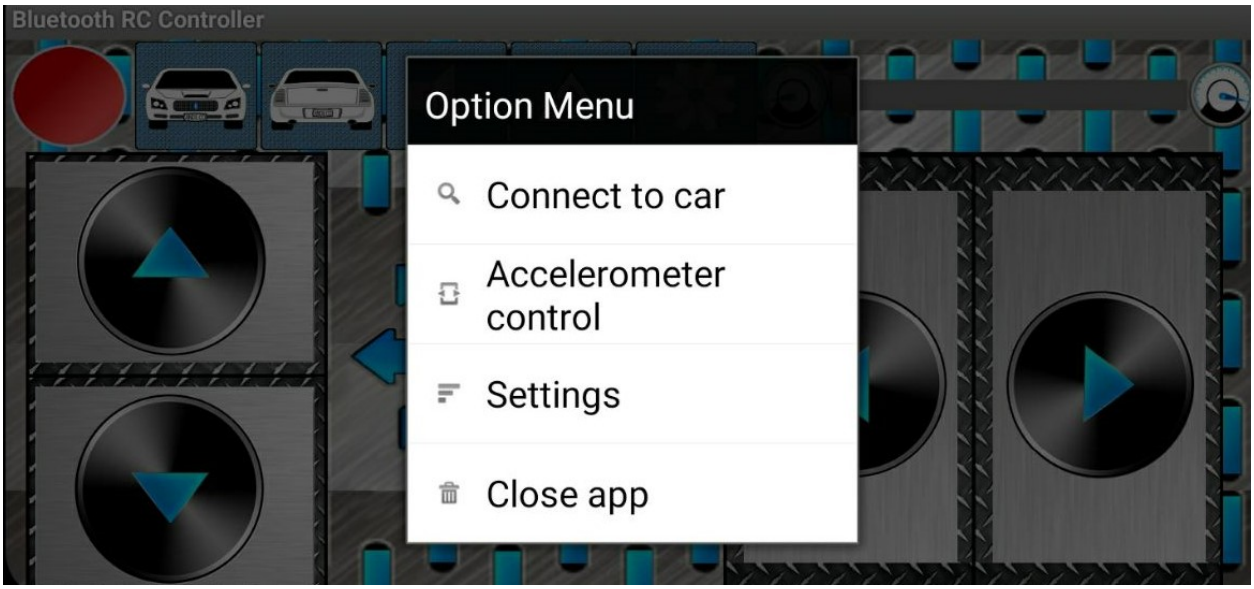
```
  digitalWrite(in1,LOW);
  digitalWrite(in2, HIGH);
}
void right()
{
  analogWrite(enA, 255);
  // Turn on motor 2
  digitalWrite(in3,LOW);
  digitalWrite(in4, HIGH);
}
void left()
{
  analogWrite(enA, 255);
  // Turn on motor 2
  digitalWrite(in3,HIGH);
  digitalWrite(in4, LOW);
}
void stop()
{
   // Turn off motors
  digitalWrite(in1, LOW);
```

```
  digitalWrite(in2, LOW);

  digitalWrite(in3, LOW);

  digitalWrite(in4, LOW);

}}
```

After uploading the code you will need an application to control the direction of your car. Lucky for you we have an application already built for this matter. What you have to do is go to the link below and download the application from Google Play Store and install it.

Arduino Bluetooth RC Controller Download Here (or just search for Arduino Bluetooth Rc Controller)

After installing the application what you have to do is just connect to the car by pairing with the bluetooth device on your car and just enjoy your remote controlled car.



References

1.lastminuteengineers

https://lastminuteengineers.com

2.SparkFun

https://learn.sparkfun.com/tutorials/what-is-an-arduino/all

3.Makerspaces

https://www.makerspaces.com/basic-electronics/

4.Electricaltechnology

https://www.electricaltechnology.org

5.Arduino book by Jermy Blum

https://www.jeremyblum.com/2019/11/22/exploring-arduino-2/

6.Diy builder youtube channel

https://www.youtube.com/watch?v=Q36NbjPMV5k

7.Arduino projects

https://create.arduino.cc/projecthub/