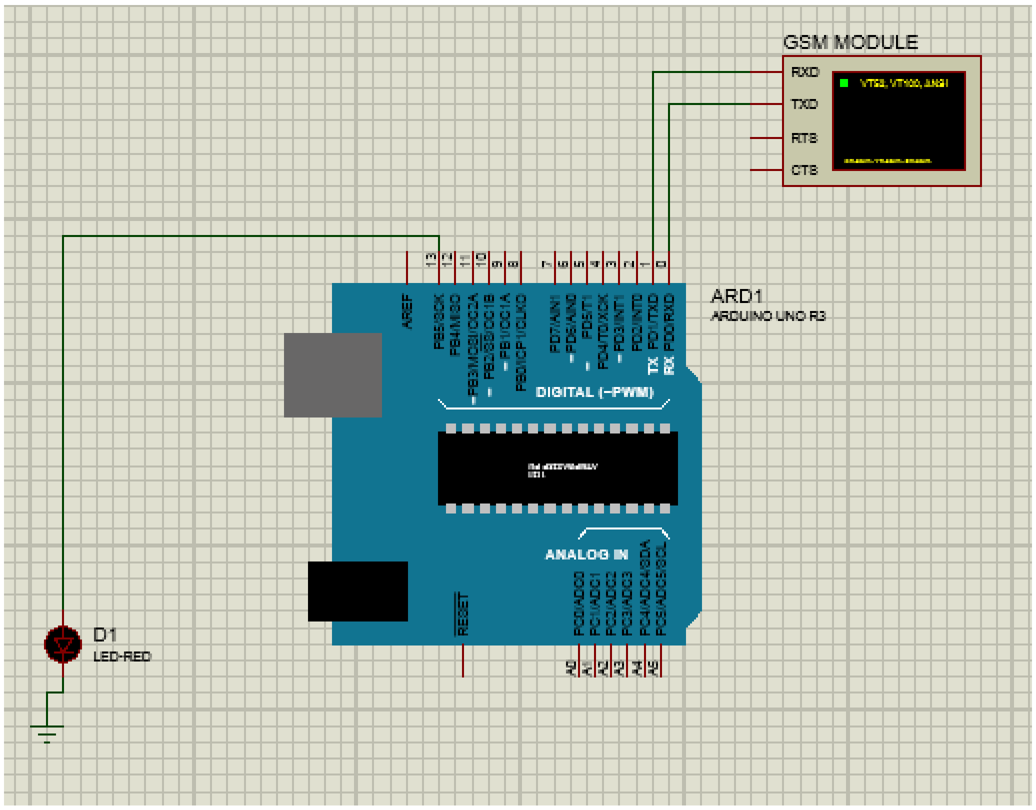
GSM RECEPTION

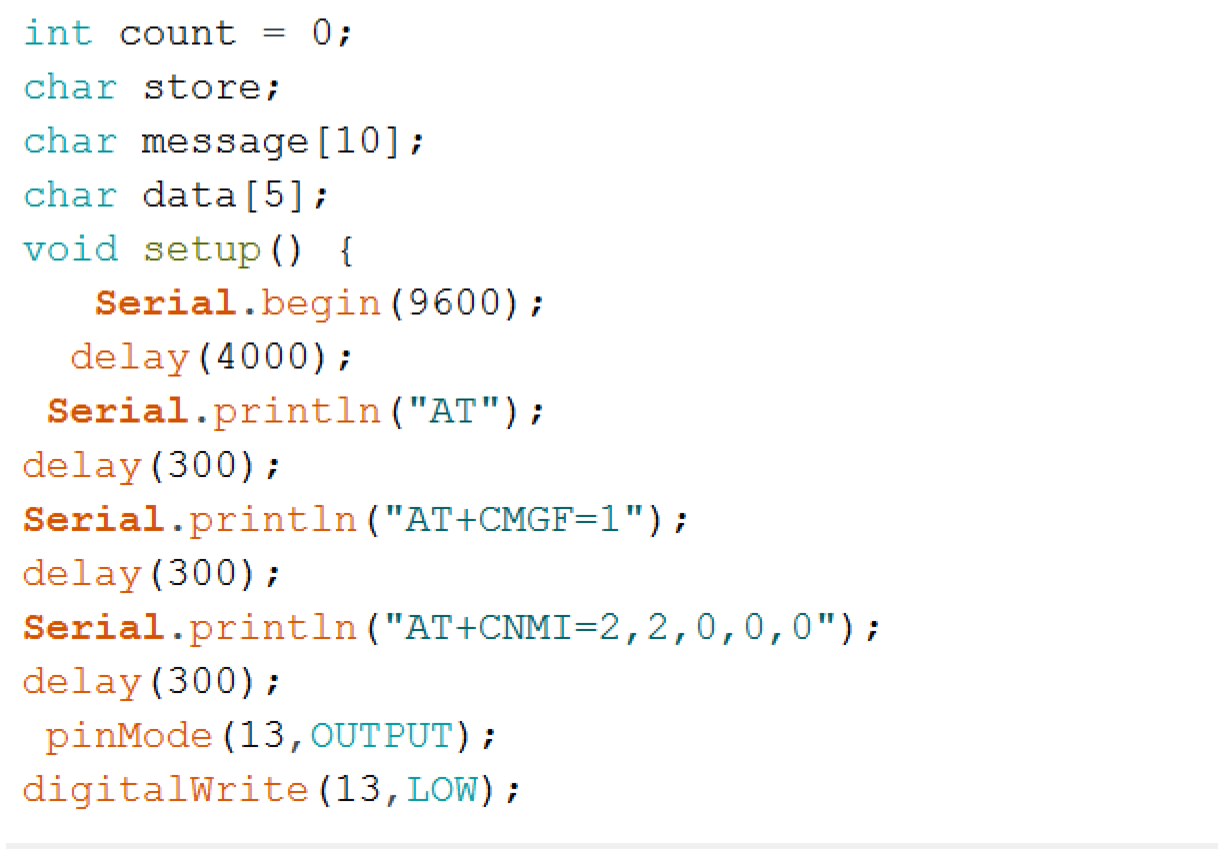
Gsm can be used to receive a message from any sim and forward that message to arduino and arduino uses that message to perform any particular task.

Example**- Below code is used to turn ON and OFF an led connected to pin number 13 of arduinouno. If we send a message *ON# from our sim to gsm the led is turned on and if message is *OFF# the led is turned off.**

Hardware connections of arduino ,gsm and led



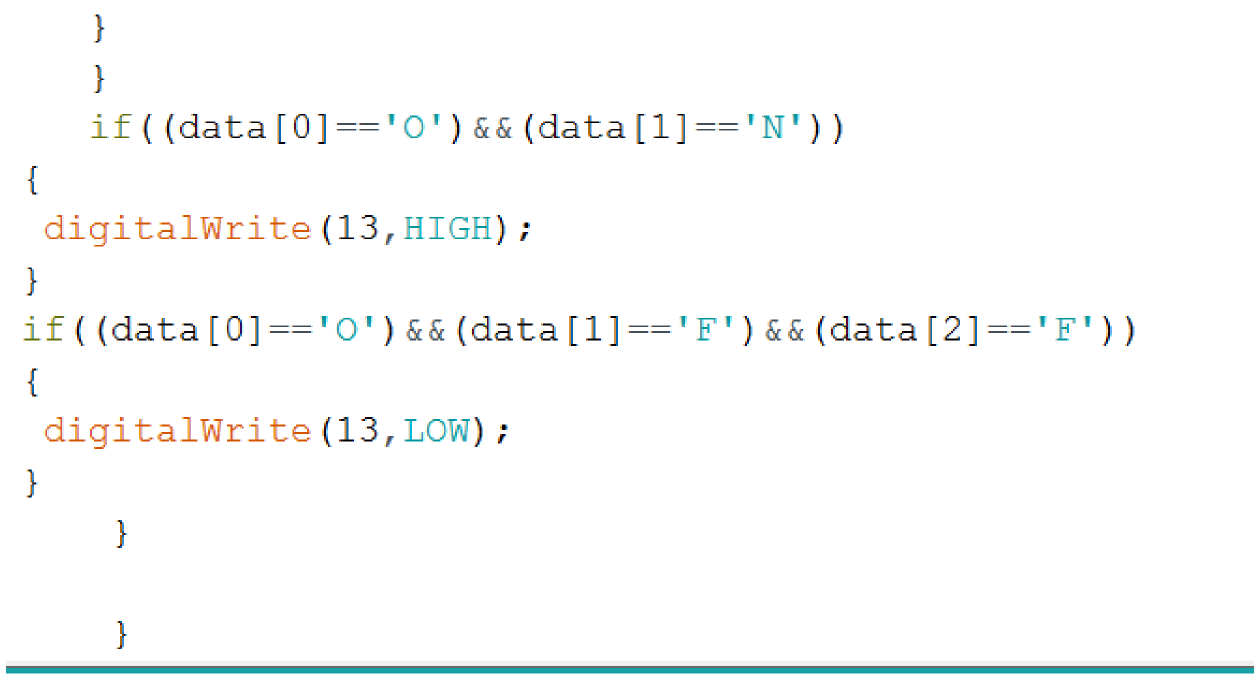
CODE FOR ABOVE PROBLEM



```
int count = 0;
char store;
char message[10];
char data[5];
void setup() {
   Serial.begin(9600);
  delay(4000);
 Serial.println("AT");
delay(300);
Serial.println("AT+CMGF=1");
delay(300);
Serial.println("AT+CNMI=2,2,0,0,0");
delay(300);
 pinMode(13,OUTPUT);
digitalWrite(13,LOW);
```

```
}

void loop() {
  delay(500);
  if(Serial.available()>0)
  {
    sei();
    store=0;
    int length = Serial.available();
    for(int i = 0; i < length; i++){
      message[i] = Serial.read();

   }
   for(int i=0;i<length;i++)
   {
```

```
   char b=message[i];
   if(b=='*')
   {
   store=1;
   count=0;
   }
   if((store==1)&&(b!='#'))
   {
     if(b=='#')
   {
   store=0;
   }
     data[count]=message[i+1];
   count++;
```



```
    }
    }
    if((data[0]=='O')&&(data[1]=='N'))
{
 digitalWrite(13,HIGH);
}
if((data[0]=='O')&&(data[1]=='F')&&(data[2]=='F'))
{
 digitalWrite(13,LOW);
}
     }

     }
```

<u>RELAY and its application</u>

Microcontrollers, like the Arduino , use relatively small voltages (3.3V/5V) and produce only tiny electrical currents (~50mA) on their I/O pins. This is perfect for powering, reading, and switching devices like buttons, LED's and small sensors.

So, what happens when you need to drive or switch a device that needs a much larger voltage or current? Motors, lamps, sirens, pumps, or any device that utilizes household mains (a big factor in home automation) can't be run directly from a microcontroller, as the voltage and current supplied won't be enough to drive these devices. Microcontrollers can't directly switch these circuits either, as the large voltages and currents would burn that microcontroller out in an instant!

The solution? Enter the relay!

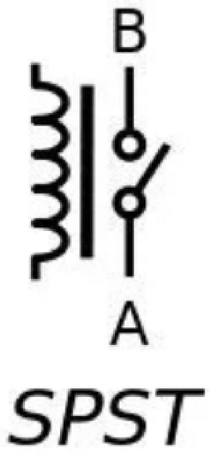
Relays are electromagnetic switches

They consist of an electromagnet coil and a lever switch. When the coil is powered on or off, it switches the lever on or off. The key, is that the coil itself utilizes a low driving voltage/current, which can be readily turned on or off by a microcontroller or sensor. The switched lever and connecting circuit within the relay on the other hand, are designed to work with much larger voltages/currents (the relays on our PiOT relay board for example are rated at 10A/250VAC or

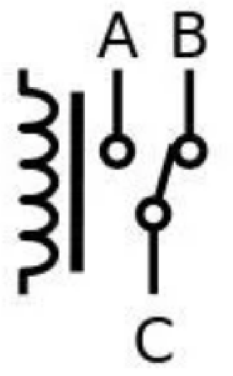
10A/24VDC each!) Thus, our microcontroller drives the relay's coil at a low power, which switches the relay's lever, which is connected to much higher power external circuit. This primarily enables the activation of large power circuits from significantly smaller ones.

Types of relays

SPST – Single Pole Single Throw. These have two terminals which can be connected or disconnected. Including two for the coil, such a relay has four terminals in total.

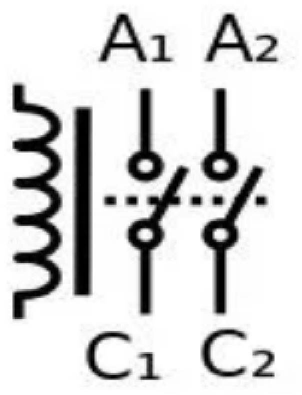


SPDT – Single Pole Double Throw. A common terminal connects to either of two others. Including two for the coil, such a relay has five terminals in total.



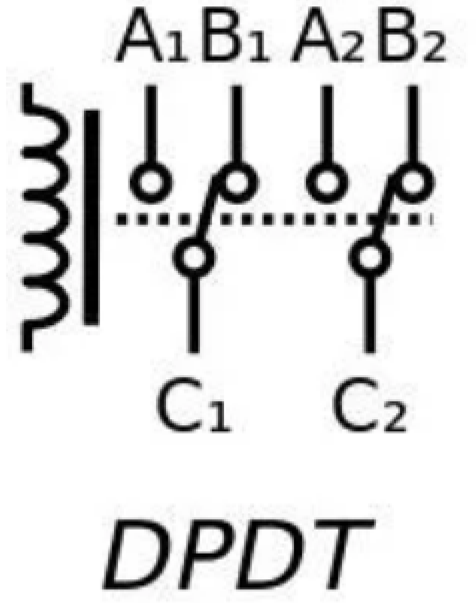
SPDT

DPST – Double Pole Single Throw. These have two pairs of terminals. Equivalent to two SPST switches or relays actuated by a single coil. Including two for the coil, such a relay has six terminals in total. The poles may be Form A or Form B (or one of each).



DPST

DPDT – Double Pole Double Throw. These have two rows of change-over terminals. Equivalent to two SPDT switches or relays actuated by a single coil. Such a relay has eight terminals, including the coil.



CODES MADE RELATED TO RELAY

1.) Simply turning on and off a motor connected to relay after every 3 seconds

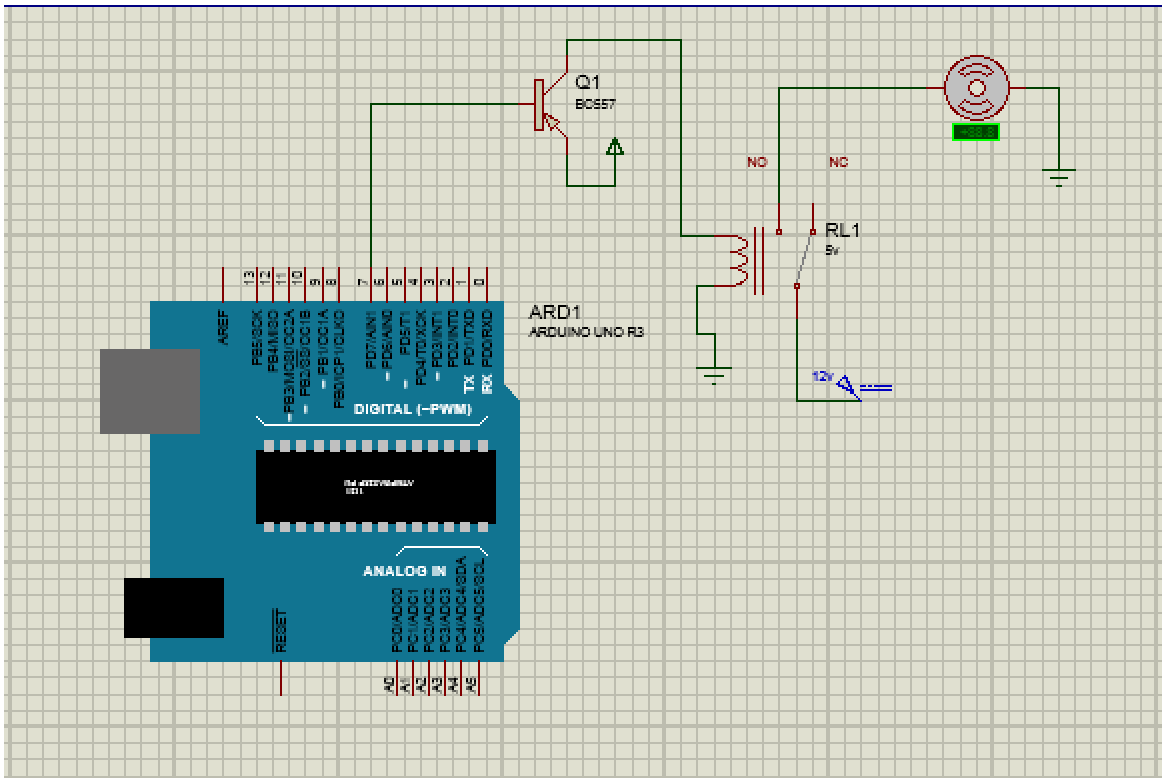
2.) Turning on and off a dc motor through Bluetooth

3.) Switching on and off and AC bulb through voice using Bluetooth

Hardware connections of relay

Relay that we used for performing practical id SPDT 5 volts relay which has five contact points. Two points are of coil, one is common where we provide the voltage required by our device to operate may it be dc voltage or AC voltage. And other are NO and NC.

Connections of arduino and relay to control DC motor

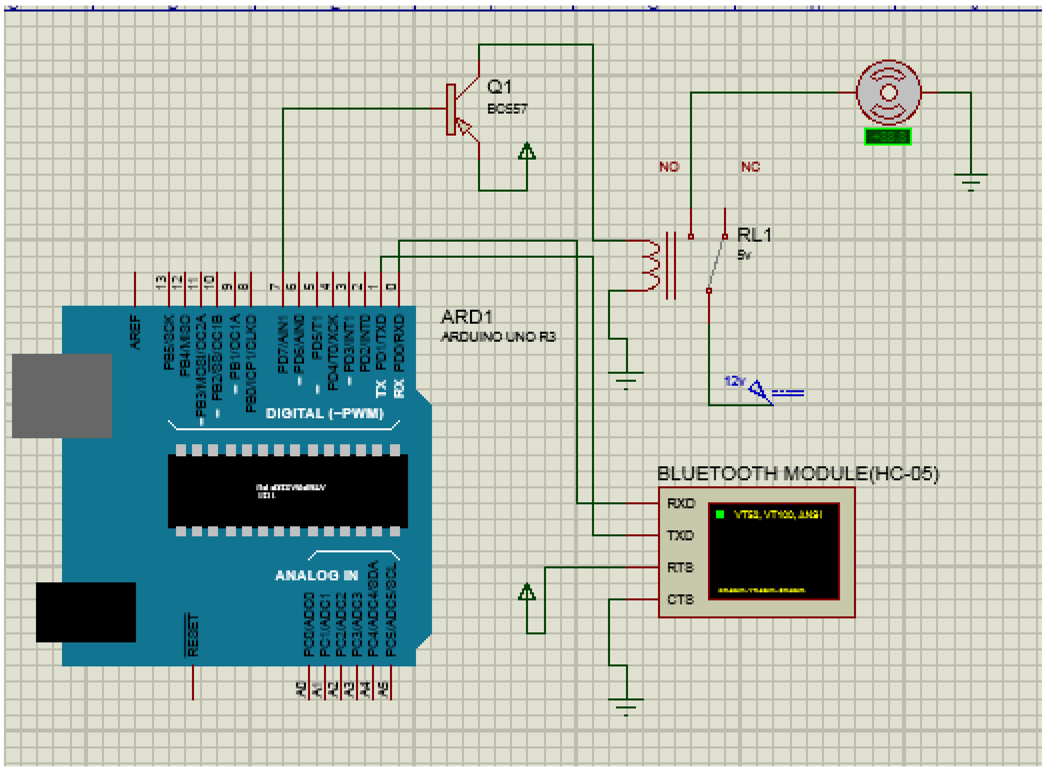


Code for turning on and off a motor connected to relay after every 3 seconds

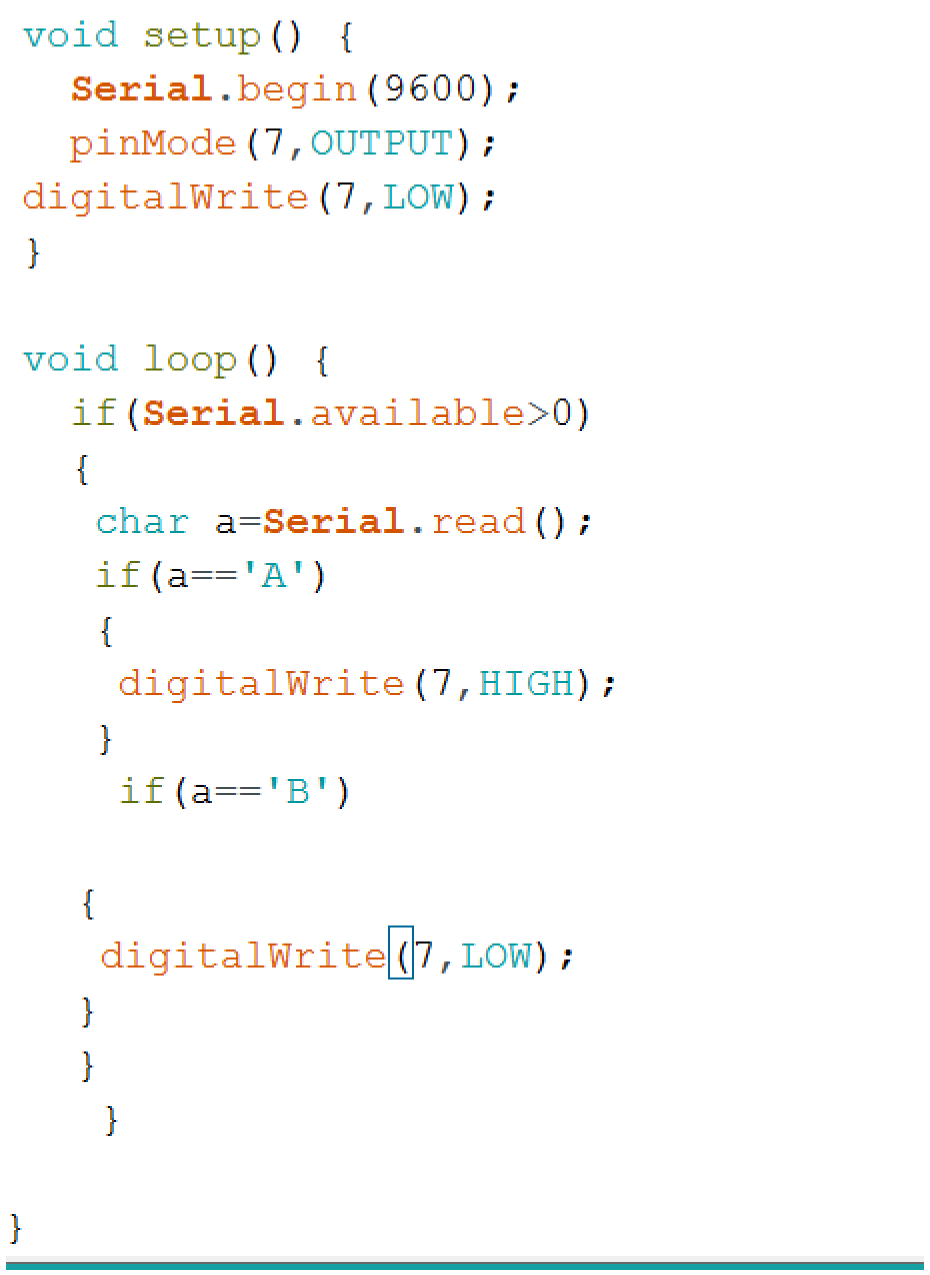
```
void setup() {
  pinMode(7,OUTPUT);
digitalWrite(7,LOW);
}

void loop() {
 digitalWrite(7,HIGH);
 delay(3000);
 digitalWrite(7,LOW);
 delay(3000);
}
```

2.) Code for turning on and off a 12volt dc motor through Bluetooth

Hardware

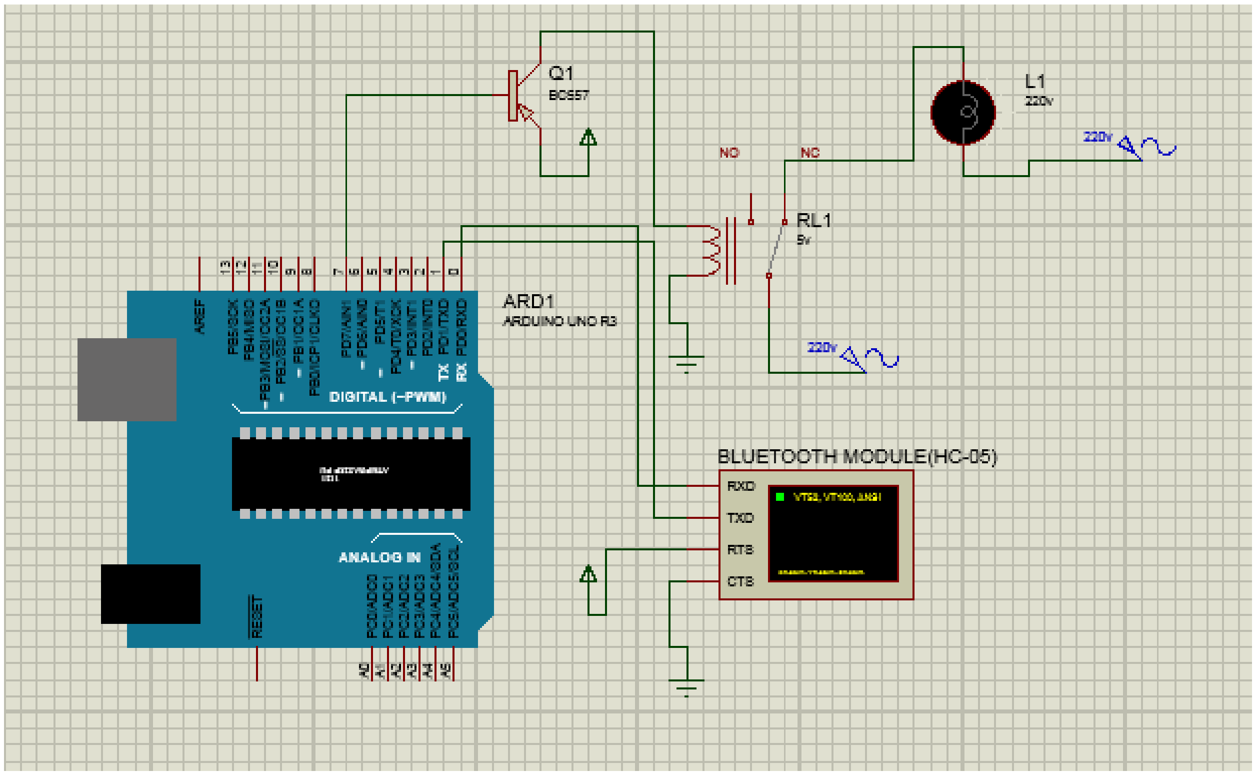


CODE:

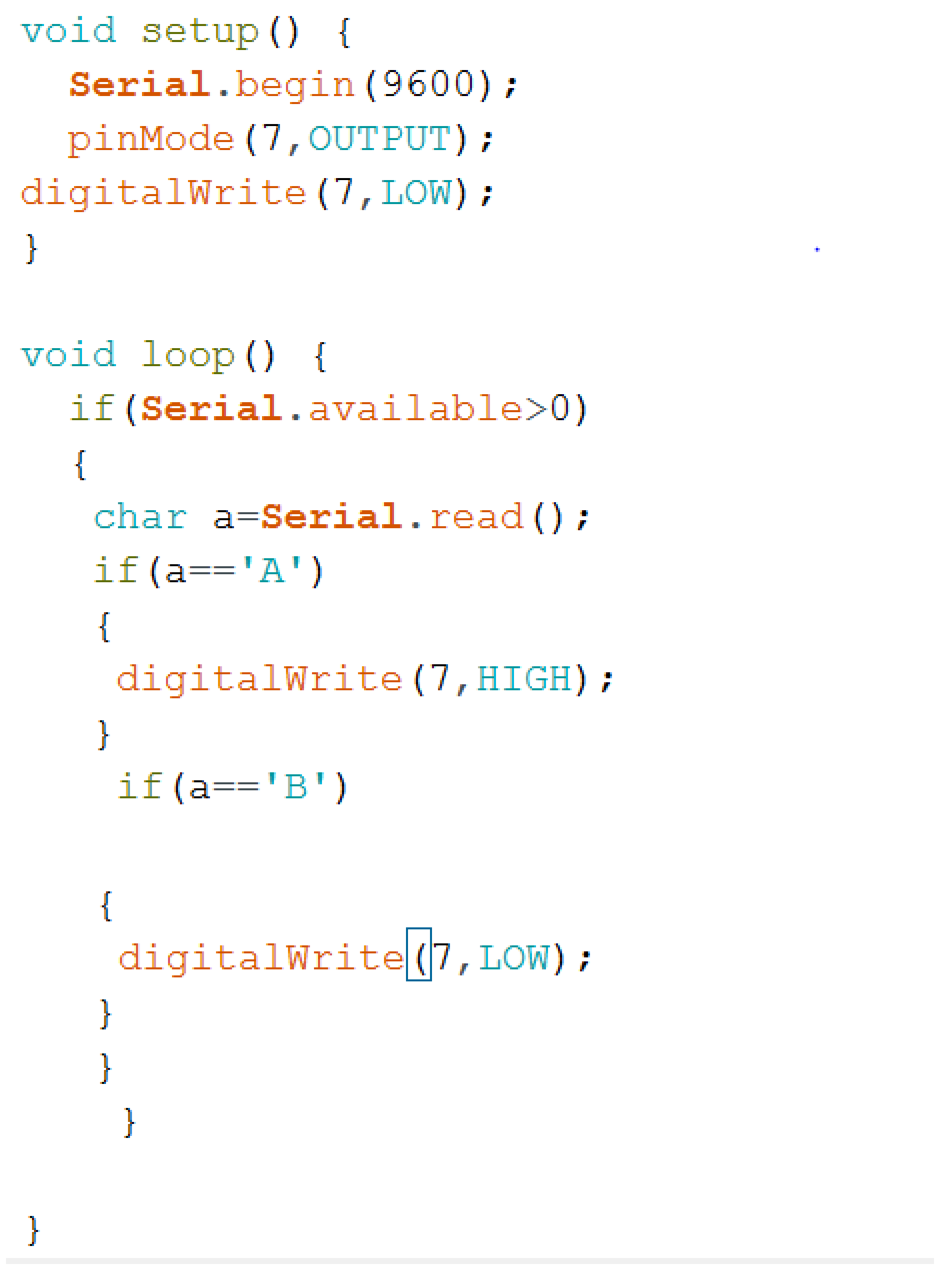


```
void setup() {
  Serial.begin(9600);
  pinMode(7,OUTPUT);
digitalWrite(7,LOW);
}

void loop() {
  if(Serial.available>0)
  {
   char a=Serial.read();
   if(a=='A')
   {
    digitalWrite(7,HIGH);
   }
    if(a=='B')

   {
   digitalWrite(7,LOW);
  }
  }
   }

}
```

3. **Code for switching on and off and AC bulb through voice using Bluetooth**


Q1
BC557
L1
220v
220v
NO
NC
RL1
5v
220v
ARD1
ARDUINO UNO R3
AREF
DIGITAL (~PWM)
ANALOG IN
RESET
TX
RX
BLUETOOTH MODULE(HC-05)
RXD
TXD
RTS
CTS
VT52, VT100, ANSI

CODE:



```
void setup() {
  Serial.begin(9600);
  pinMode(7,OUTPUT);
digitalWrite(7,LOW);
}

void loop() {
  if(Serial.available>0)
  {
   char a=Serial.read();
   if(a=='A')
   {
    digitalWrite(7,HIGH);
   }
    if(a=='B')

   {
    digitalWrite(7,LOW);
   }
   }
    }

}
```

MOTORS

we will be studying two motors

1.)DC geared motor
2.)Servo motor

DC geared motor operated on 12volts supply and it is used for clockwise and anticlockwise rotation.
Servo motors are used to achieve rotations at particular angles.\

As supply requirement of DC motors is 12 volts so arduino directly cannot be used to operate dc motor we use motor driver IC l293d to operate DC geared motors.

ABOUT L293D MOTOR DRIVER IC

L293d motor driver ic servers two purpose:

1. Takes 5volts signal from arduino and in turn provides 12volts signals for operating dc motor.
2. It it used for fulfilling proper current requirement of dc motor.

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC.

There are two Enable pins on l293d. Pin 1 and pin 9, for being able to drive the motor, the pin 1 and 9 need to be high. For driving the motor with left side you need to enable pin 1 to high. And for right side you need to make the pin 9 to high.

Working of L293D

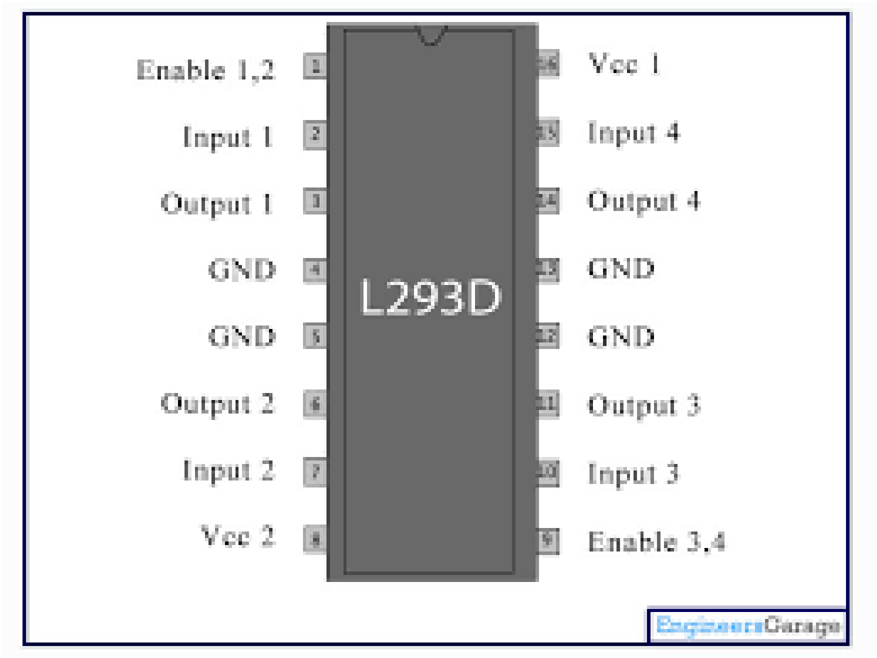
There are 4 input pins for l293d, pin 2,7 on the left and pin 15 ,10 on the right as shown on the pin diagram. Left input pins will regulate the rotation of motor connected across left side and right input for motor on the right hand side. The motors are rotated on the basis of the inputs provided across the input pins as LOGIC 0 or LOGIC 1.

In simple you need to provide Logic 0 or 1 across the input pins for rotating the motor.

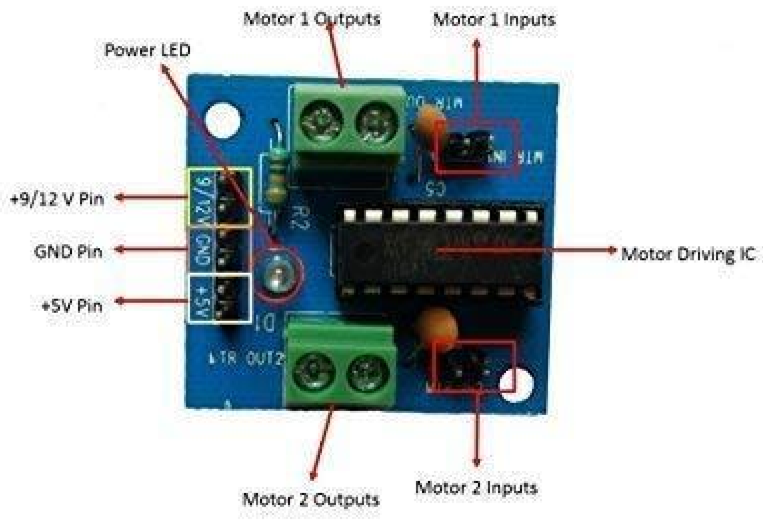
L293D Logic Table.

Lets consider a Motor connected on left side output pins (pin 3,6). For rotating the motor in clockwise direction the input pins has to be provided with Logic 1 and Logic 0.

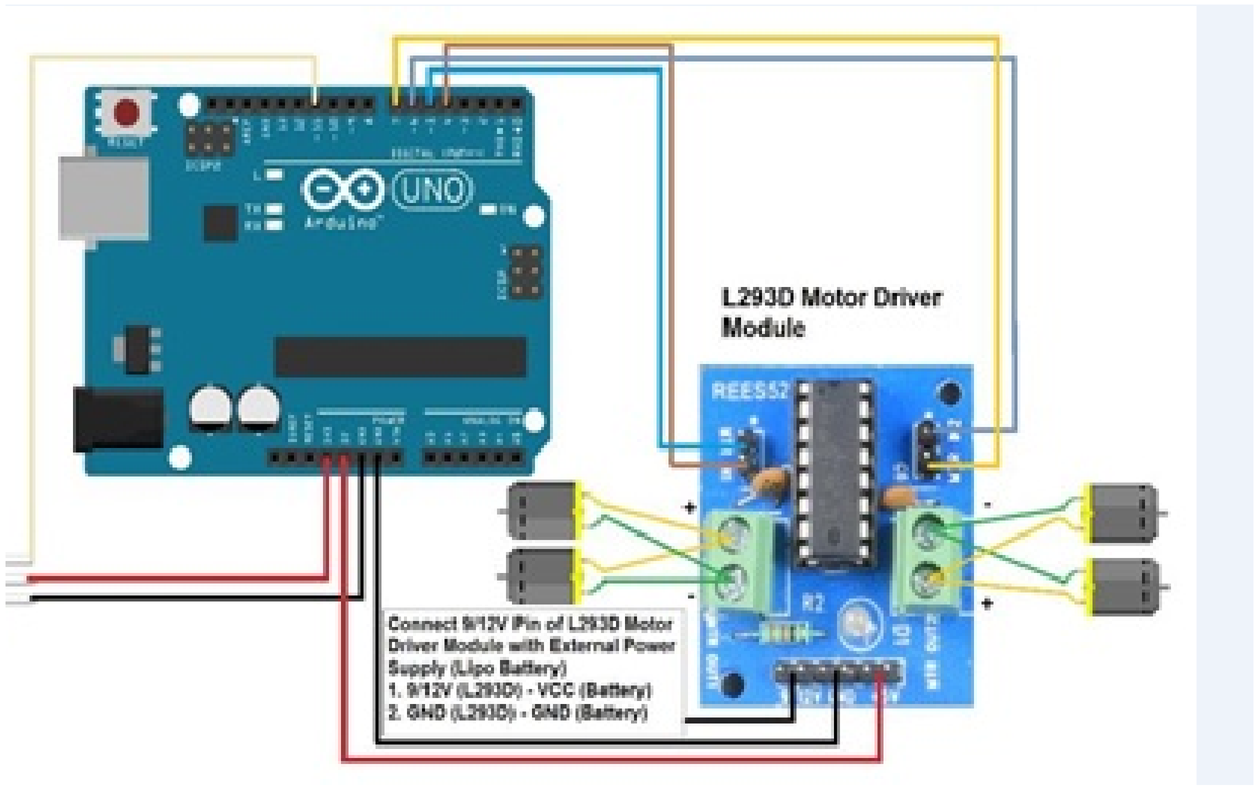
- **Pin 2** = **Logic 1** and **Pin 7** = **Logic 0** | Clockwise Direction
- **Pin 2** = **Logic 0** and **Pin 7** = **Logic 1** | Anticlockwise Direction
- **Pin 2** = **Logic 0** and **Pin 7** = **Logic 0** | Idle [No rotation] [Hi-Impedance state]
- **Pin 2** = **Logic 1** and **Pin 7** = **Logic 1** | Idle [No rotation]



MOTOR DRIVER MODULE



CONNECTION DIAGRAM


L293D Motor Driver Module
REES52
Connect 9/12V Pin of L293D Motor Driver Module with External Power Supply (Lipo Battery)
1. 9/12V (L293D) - VCC (Battery)
2. GND (L293D) - GND (Battery)
UNO
Arduino

CODES FOR DC MOTOR

1. Code for rotating dc motor in clockwise and anti-clockwise direction after every 3 seconds.
2. Rotating the motor clockwise and anti-clockwise through Bluetooth.
3. Bluetooth based robotic car

Code for rotating dc motor in clockwise and anti-clockwise direction after every 3 seconds

Suppose INPUT1 and INPUT2 are connected to pin 13 and 12 o arduino . And motor is connected across out1 and out2

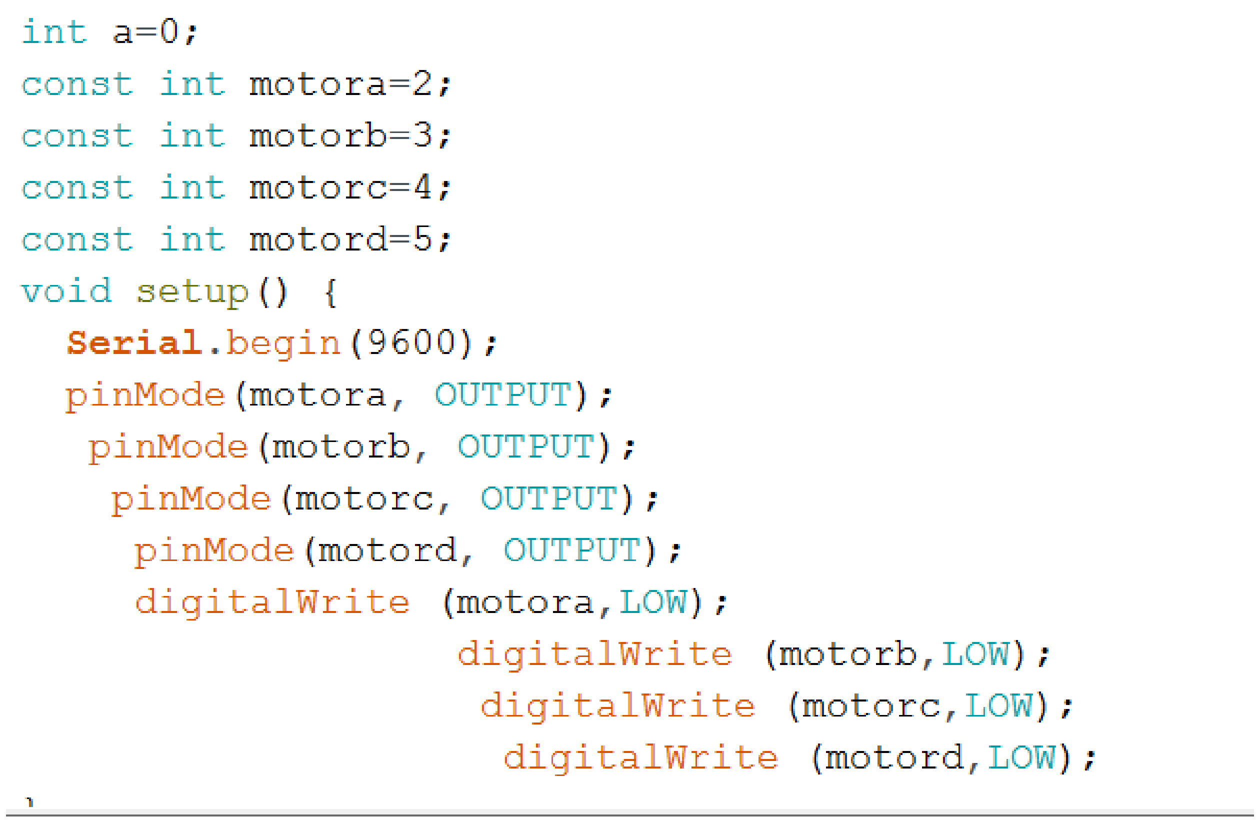
Code:

```
void setup() {
 pinMode(13,OUTPUT);
  pinMode(12,OUTPUT);
  digitalWrite(13,LOW);
    digitalWrite(12,LOW);
  }
void loop() {
   digitalWrite(13,HIGH);
    digitalWrite(12,LOW);
    delay(3000);
     digitalWrite(13,LOW);
    digitalWrite(12,HIGH);
delay(3000);
}
```

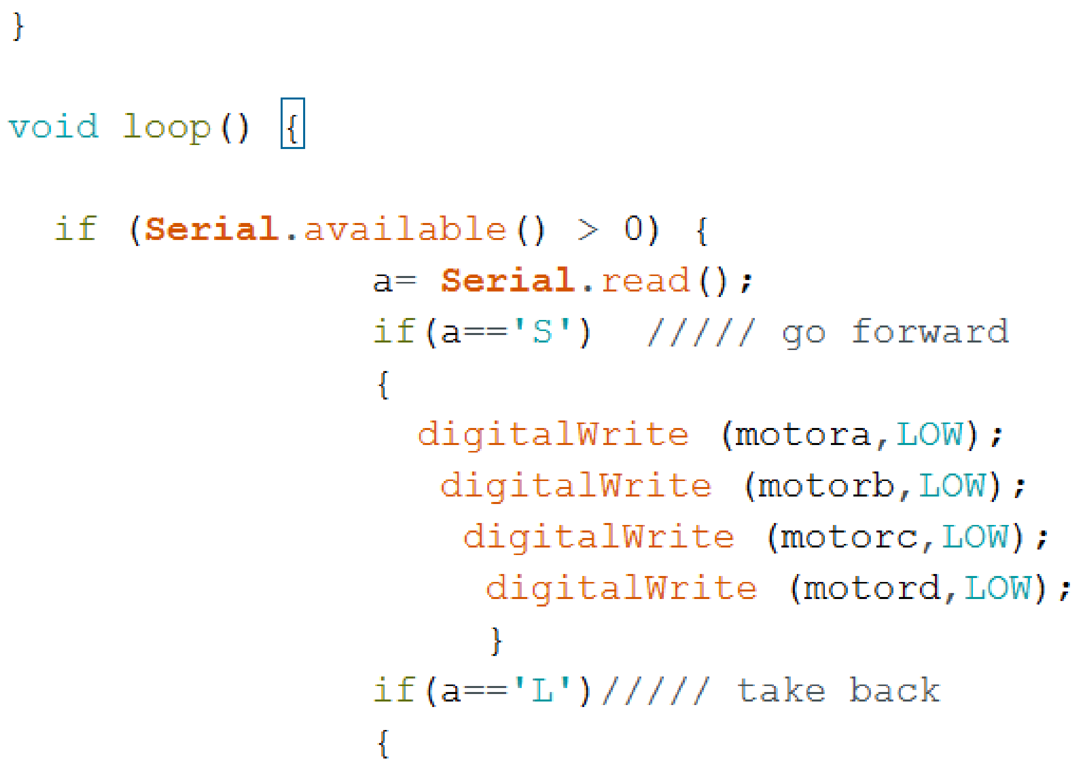
3.) Rotating motor clockwise and anticlockwise using Bluetooth

```
void setup() {
  Serial.begin(9600);
 pinMode(13,OUTPUT);
  pinMode(12,OUTPUT);
  digitalWrite(13,LOW);
    digitalWrite(12,LOW);
  }
void loop() {
  if(Serial.available()>0)
  □
    char a= Serial.read();
    if(a=='A')
    {
      digitalWrite(13,HIGH);
    digitalWrite(12,LOW);
      -
      }
      if(a=='B')
    {
      digitalWrite(13,LOW);
    digitalWrite(12,HIGH);
      }
  }
}
```

4.) Code for Bluetooth based robotic car



```
int a=0;
const int motora=2;
const int motorb=3;
const int motorc=4;
const int motord=5;
void setup() {
  Serial.begin(9600);
  pinMode(motora, OUTPUT);
   pinMode(motorb, OUTPUT);
    pinMode(motorc, OUTPUT);
     pinMode(motord, OUTPUT);
     digitalWrite (motora,LOW);
                  digitalWrite (motorb,LOW);
                   digitalWrite (motorc,LOW);
                    digitalWrite (motord,LOW);
```



```
}

void loop() {

  if (Serial.available() > 0) {
                a= Serial.read();
                if(a=='S')  ///// go forward
                {
                  digitalWrite (motora,LOW);
                   digitalWrite (motorb,LOW);
                    digitalWrite (motorc,LOW);
                     digitalWrite (motord,LOW);
                     }
                if(a=='L')///// take back
                {
```

```
 digitalWrite (motora,HIGH);
   digitalWrite (motorb,LOW);
    digitalWrite (motorc,LOW);
     digitalWrite (motord,LOW);

}
if(a=='R')  ///// go forward
{
digitalWrite (motora,LOW);
   digitalWrite (motorb,LOW);
    digitalWrite (motorc,HIGH);
     digitalWrite (motord,LOW);

                                  }
if(a=='F')///// take back

{
digitalWrite (motora,HIGH);
   digitalWrite (motorb,LOW);
    digitalWrite (motorc,HIGH);
     digitalWrite (motord,LOW);

}
if(a=='B')  ///// go forward
{
  digitalWrite (motora,LOW);
   digitalWrite (motorb,HIGH);
    digitalWrite (motorc,LOW);
     digitalWrite (motord,HIGH);
}
     }
```

```
}
```

SERVO MOTOR

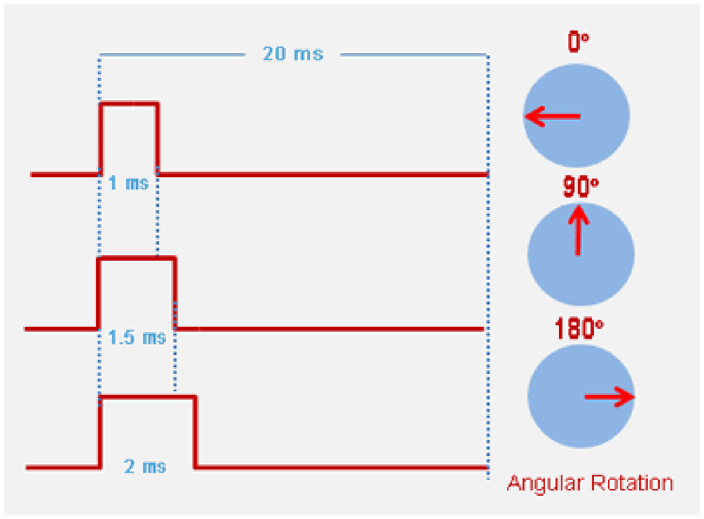
A **servo motor** is an electrical device which can push or rotate an object with great precision. If you want to rotate and object at some specific angles or distance, then you use servo motor. It is just made up of simple motor which run through **servo mechanism**. If motor is used is DC powered then it is called DC servo motor, and if it is AC powered motor then it is called AC servo motor. We can get a very high torque servo motor in a small and light weight packages. Doe to these features they are being used in many applications like toy car, RC helicopters and planes, Robotics, Machine etc.

Servo motors are rated in kg/cm (kilogram per centimeter) most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm tells you how much weight your servo motor can lift at a particular distance. For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motors shaft, the greater the distance the lesser the weight carrying capacity.

The position of a servo motor is decided by electrical pulse

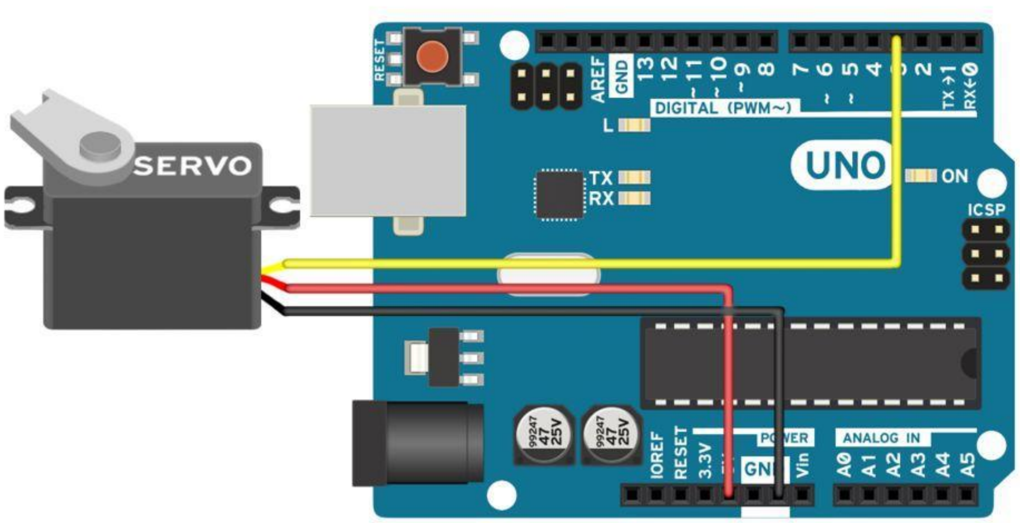
All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU.

Servo motor is controlled by PWM (Pulse with Modulation) which is provided by the control wires.



TIMING TABLE FOR SERVO ANGLES

HARDWARE CONNECTIONS OF SERVO WITH ARDUINO



CODE :

Code to run a servo motor from 0 degree to 180 degree and then again from 180 degree back to 0 degree again and again in loop.

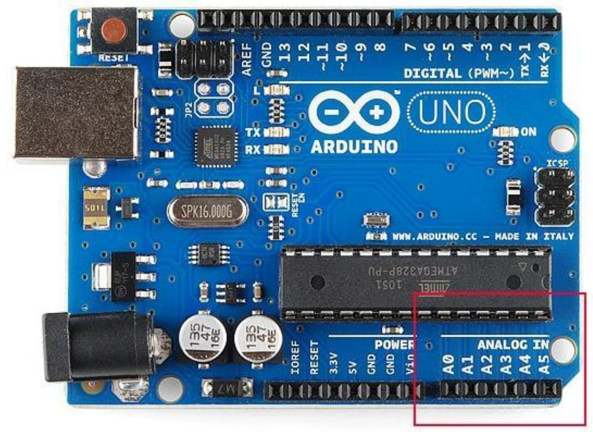
```
#include <Servo.h>
Servo myservo;
int pos = 0;
void setup() {
  myservo.attach(3);  // attaches the servo on pin 3 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'

    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

ADC(ANALOG TO DIGITAL CONVERTION)

Microcontrollers are capable of detecting binary signals: is the button pressed or not? These are digital signals. When a microcontroller is powered from five volts, it understands zero volts (0V) as a binary 0 and a five volts (5V) as a binary 1. The world however is not so simple and likes to use shades of gray. What if the signal is 2.72V? Is that a zero or a one? We often need to measure signals that vary; these are called analog signals. A 5V analog sensor may output 0.01V or 4.99V or anything inbetween. Luckily, nearly all microcontrollers have a device built into them that allows us to convert these voltages into values that we can use in a program to make a decision.

What is the ADC?

An Analog to Digital Converter (ADC) is a very useful feature that converts an analog voltage on a pin to a digital number. By converting from the analog world to the digital world, we can begin to use electronics to interface to the analog world around us.



Not every pin on a microcontroller has the ability to do analog to digital conversions. On the Arduino board, these pins have an 'A' in front of their label (A0 through A5) to indicate these pins can read analog voltages.

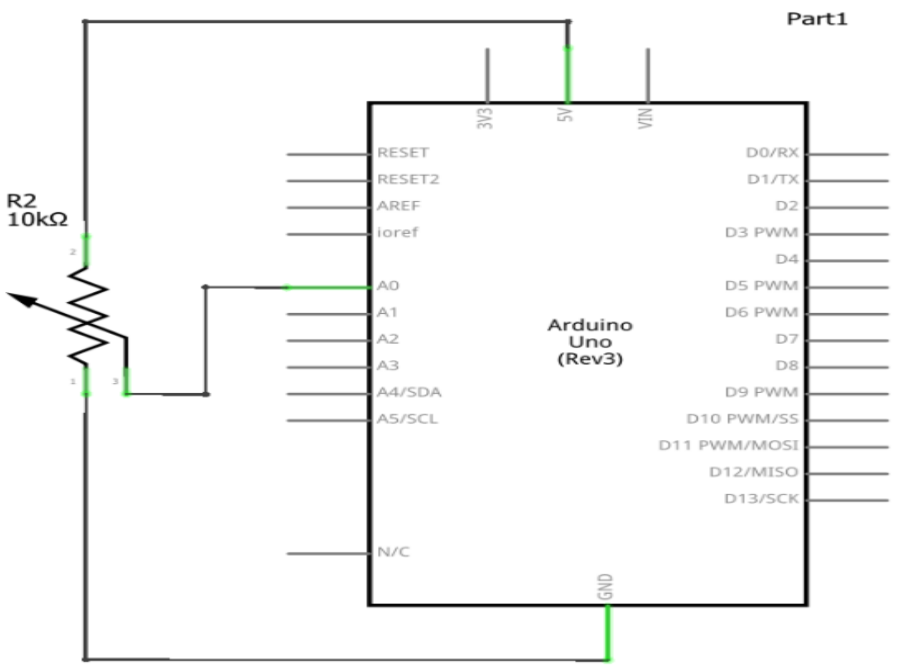
Arduino board has six ADC channels, as show in figure below. Among those any one or all of them can be used as inputs for analog voltage. The **Arduino Uno ADC** is of 10 bit resolution (so the integer values from (0-(2^10) 1023)). This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. So for every (5/1024= 4.9mV) per unit.

we can read value from ADC of channel '0' by directly calling function "analogRead(pin);", here "pin" represents pin where we connected analog signal, in this case it would be "A0". The value from ADC can be taken into an integer as "int ADCVALUE = analogRead(A0); ", by this instruction the value after ADC gets stored in the integer "ADCVALUE".

CODE AND HARDWARE CONNECTIONS FOR ADC

To show the concept of ADC we have interfaced potentiometer(Variable resistor) with arduino and we have connecte potentiometer to A0(Analog 0) pin of Arduino.

Hardware conections:

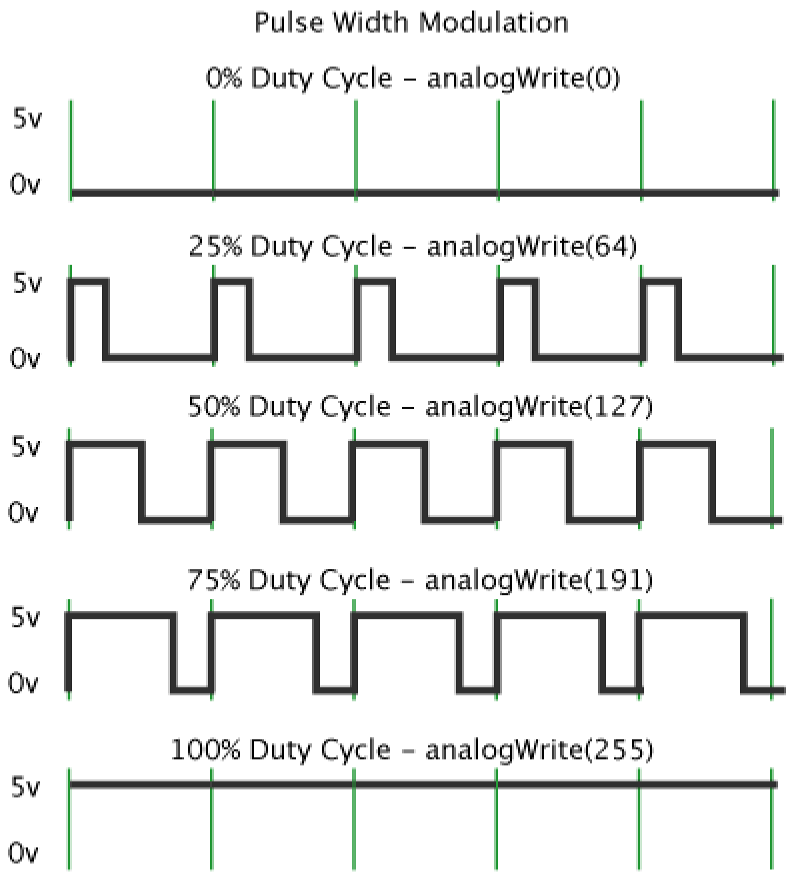


CODE:

```
void setup() {
pinMode(A0,INPUT);
}

void loop() {
  float a=analogRead(A0);
  Serial.print(a);   ///////digital value
  float b=(a*5)/1024;
  Serial.print("     ");
  Serial.println(b);   ///////Voltage value of potentiometer
  }
```

PWM(PULSE WIDTH MODULATION):

Pulse Width Modulation or PWM is a common technique used to vary the width of the pulses in a pulse-train. Pulse width modulation is basically, a square wave with a varying high and low time.



Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
25% Duty Cycle – analogWrite(64)
50% Duty Cycle – analogWrite(127)
75% Duty Cycle – analogWrite(191)
100% Duty Cycle – analogWrite(255)
5v
0v

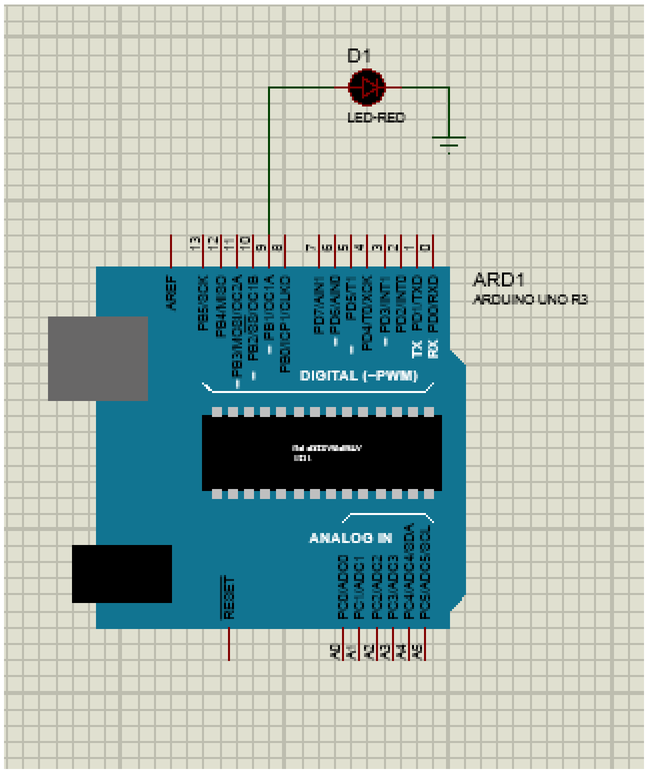
- **On-Time** – Duration of time signal is high.
- **Off-Time** – Duration of time signal is low.
- **Period** – It is represented as the sum of on-time and off-time of PWM signal.
- **Duty Cycle** – It is represented as the percentage of time signal that remains on during the period of the PWM signal.

Arduinouno supports 8-bit PWM so a call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time).

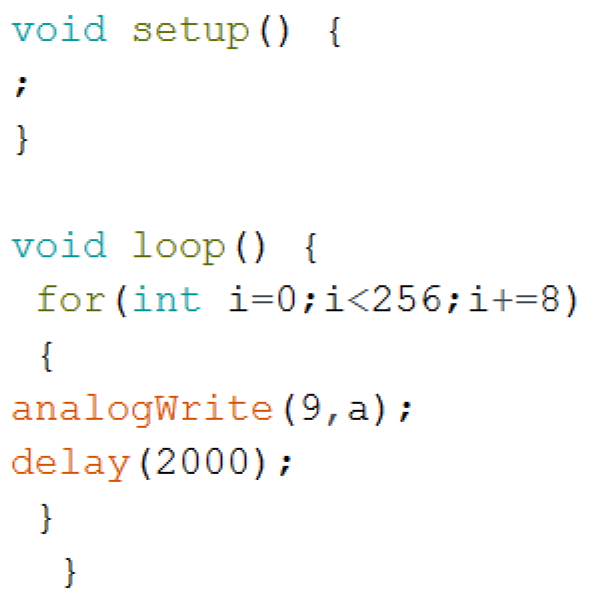
CODES FOR PWM

1.) Controlling the brightness of led through PWM signal
2.) Controlling the brightness of led with the change in the value of ADC value of potentiometer
3.) Controlling the led brightness with the signal received from Bluetooth.

CODE 1HARDWARE CONNECTIONS AND CODE

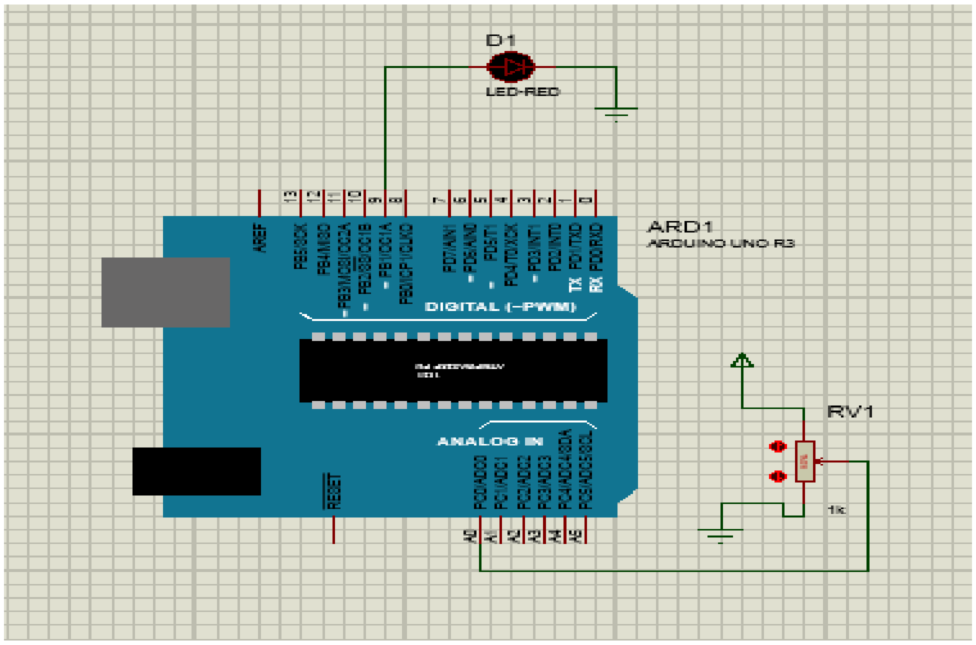


CODE:



```
void setup() {
;
}

void loop() {
 for(int i=0;i<256;i+=8)
 {
analogWrite(9,a);
delay(2000);
 }
  }
```

CODE 2HARDWARE CONNECTIONS AND CODE

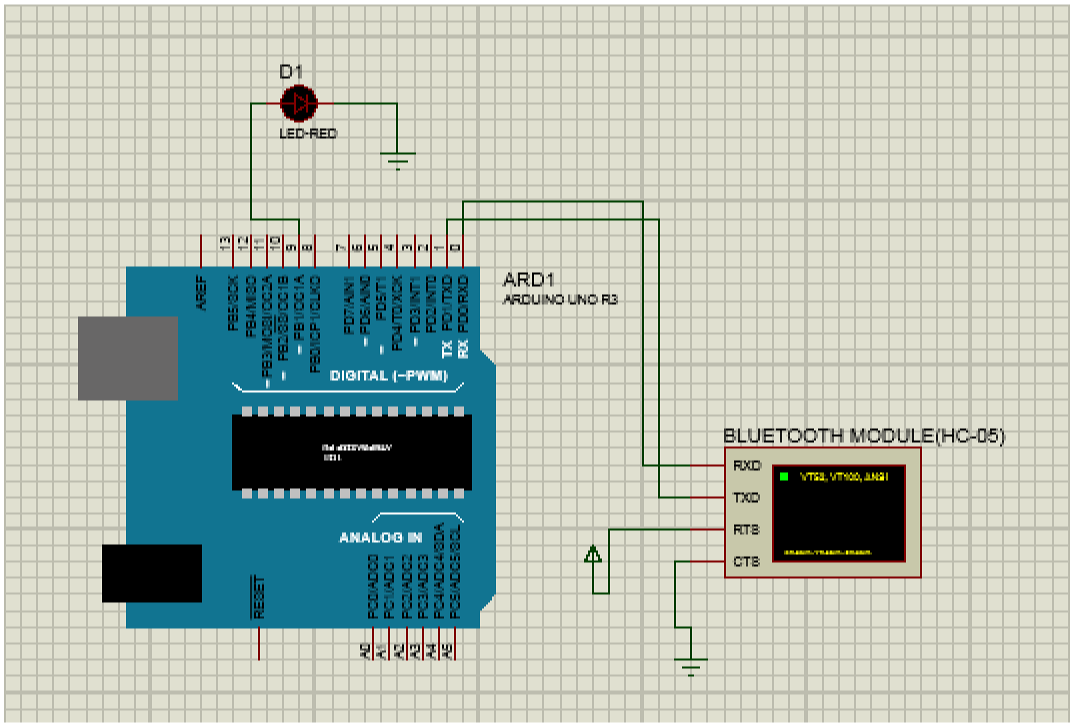


CODE:

```
void setup() {
pinMode(A0,INPUT);
}

void loop() {
  int a=analogRead(A0);
  a=a/4;
analogWrite(9,a);
  }
```

CODE 3 HARDWARE CONNECTIONS AND CODE

HARDWARE:



CODE:

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

void loop() {
  if(Serial.available()>0)
  {

      int a=Serial.read();
      Serial.println(a);
      a=a-48;
      a=a*25;
      Serial.println(a);
      analogWrite(9,a);
```

SENSORS

Sensors are used today in a wide range of applications throughout the electronics industry. As electrical components become smaller and more intricate, the need to ensure specific measurements are adhered to grows in importance, with quality control processes becoming fundamental and complex.

Sensors covered underthis session are:

1.) Ultrasonic sensor(SR-04)

2.) DHT11 sensor(Digital Humidity Temperature Sensor)

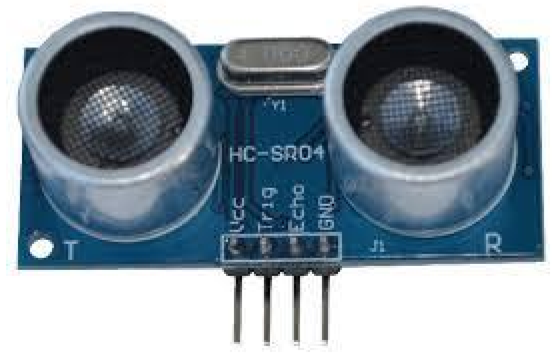
3.) Alcohol Sensor

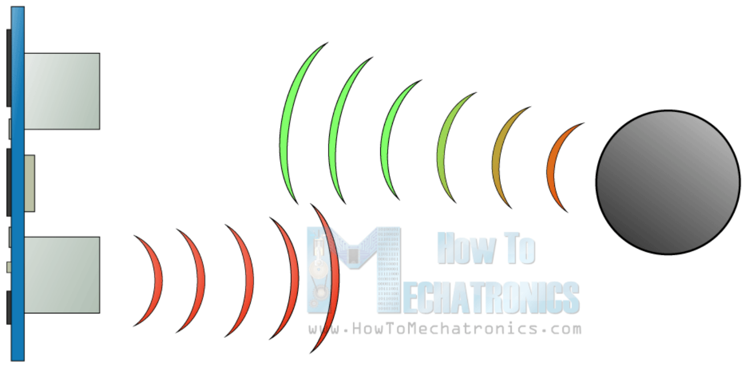
ULTRASONIC SENSOR(SR-04)

Ultasonic sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse.

Ultrasonic sensors are a reliable, cost-effective solution for distance sensing, level, and obstacle detection.

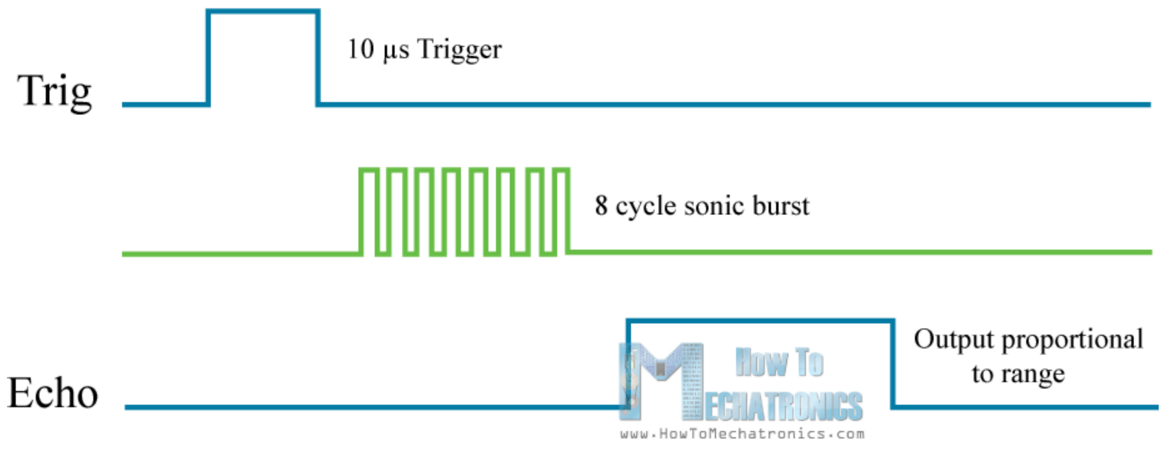
These devices work on a principle similar to that of transducers used in radar and sonar systems, which evaluate attributes of a target by interpreting the echoes from radio or sound waves, respectively.





The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board.

In order to generate the ultrasound you need to set the Trig on a High State for 10 µs. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave traveled.



For example, if the object is 10 cm away from the sensor, and the speed of the sound is 340 m/s or 0.034 cm/µs the sound wave will need to travel about 29 u seconds to cover 1 cm. But what you will get from the Echo pin will be double that number because the sound wave needs to travel forward and bounce backward. So in order to get the distance in cm we need to divide the received travel time value from the echo pin by 29 and divide it by 2.

CODE1 : To measure the distance of obstacle from ultrasonic sensor and display the result on Serial Monitor

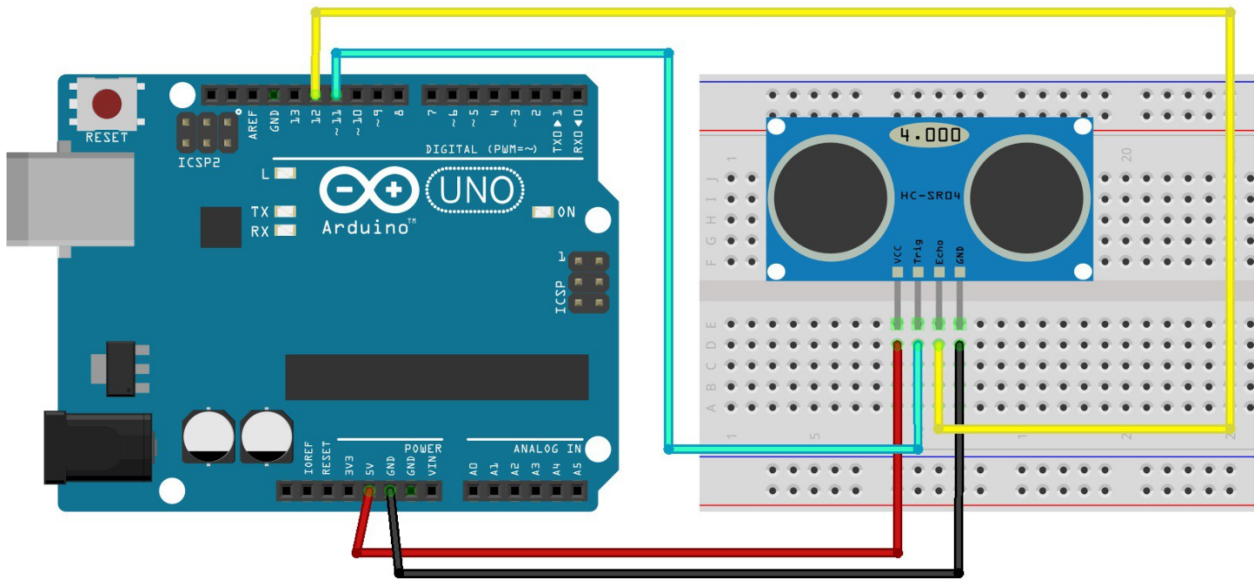
CODE EXPLANATION

First you have to define the Trig and Echo pins. In this case they are the pins number 11 and 12 on the Arduino Board and they are named trigPin and echoPin. Then you need a Long variable, named "duration" for the travel time that you will get from the sensor and an integer variable for the distance.

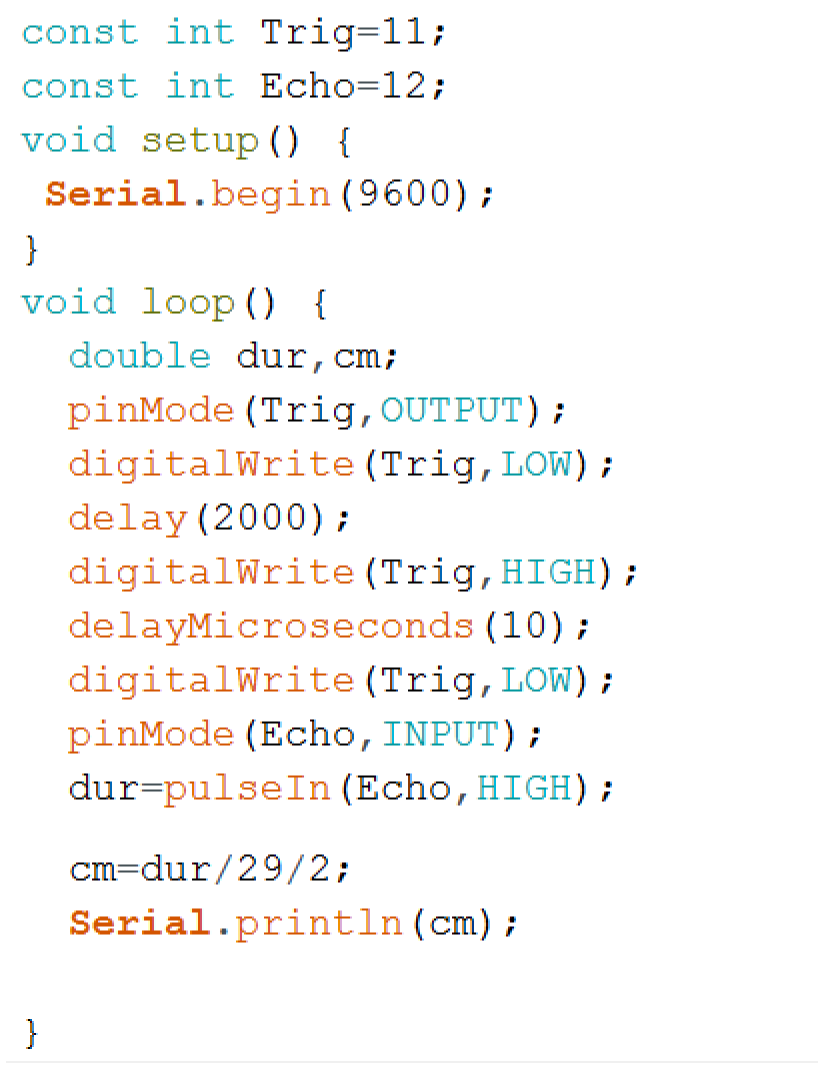
In the setup you have to define the trigPin as an output and the echoPin as an Input and also start the serial communication for showing the results on the serial monitor.

In the loop first you have to make sure that the trigPin is clear so you have to set that pin on a LOW State for just 2 µs. Now for generating the Ultra sound wave we have to set the trigPin on HIGH State for 10 µs. Using the ***pulseIn()*** function you have to read the travel time and put that value into the variable "duration". This function has 2 parameters, the first one is the name of the echo pin and for the second one you can write either HIGH or LOW. In this case, HIGH means that the ***pulsIn()***function will wait for the pin to go HIGH caused by the bounced sound wave and it will start timing, then it will wait for the pin to go LOW when the sound wave will end which will stop the timing. At the end the function will return the length of the pulse in microseconds. For getting the distance we will divide the duration by 29 and divide it by 2 as we explained this equation previously. At the end we will print the value of the distance on the Serial Monitor.

HARDWARE CONNECTIONS AND CODE:



CODE:



```
const int Trig=11;
const int Echo=12;
void setup() {
 Serial.begin(9600);
}
void loop() {
  double dur,cm;
  pinMode(Trig,OUTPUT);
  digitalWrite(Trig,LOW);
  delay(2000);
  digitalWrite(Trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig,LOW);
  pinMode(Echo,INPUT);
  dur=pulseIn(Echo,HIGH);

  cm=dur/29/2;
  Serial.println(cm);

}
```

CODE2 : To measure the distance of obstacle from ultrasonic sensor and to turn the buzzer on if the distance is less than 20cm and turn it off if the distance is greater than 20 cm

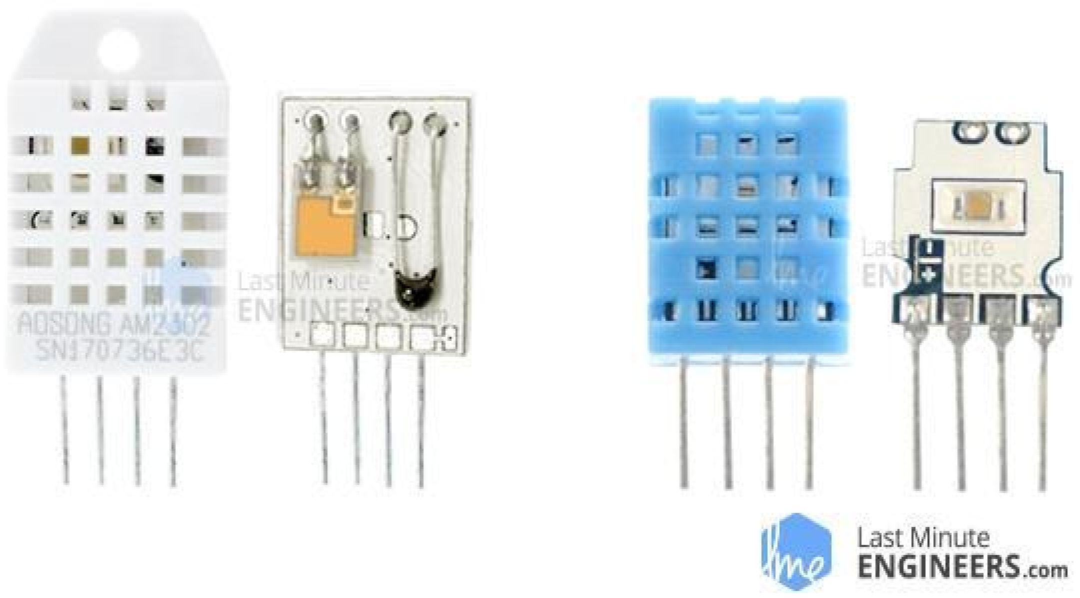
```
const int Trig=11;
const int Echo=12;
const int buzzer=11;

void setup() {
 Serial.begin(9600);
 pinMode(buzzer,OUTPUT);
 digitalWrite(buzzer,LOW);
}

void loop() {
  double dur,cm;
  pinMode(Trig,OUTPUT);
  digitalWrite(Trig,LOW);
  delay(2000);

  digitalWrite(Trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig,LOW);
  pinMode(Echo,INPUT);
  dur=pulseIn(Echo,HIGH);
  cm=dur/29/2;
  Serial.println(cm);
  if(cm<20)
  {
    digitalWrite(buzzer,HIGH);
  }
  else
  ▯
    digitalWrite(buzzer,LOW);
  }}
```

DHT11 (DigitalHumidity Temperature Sensor)

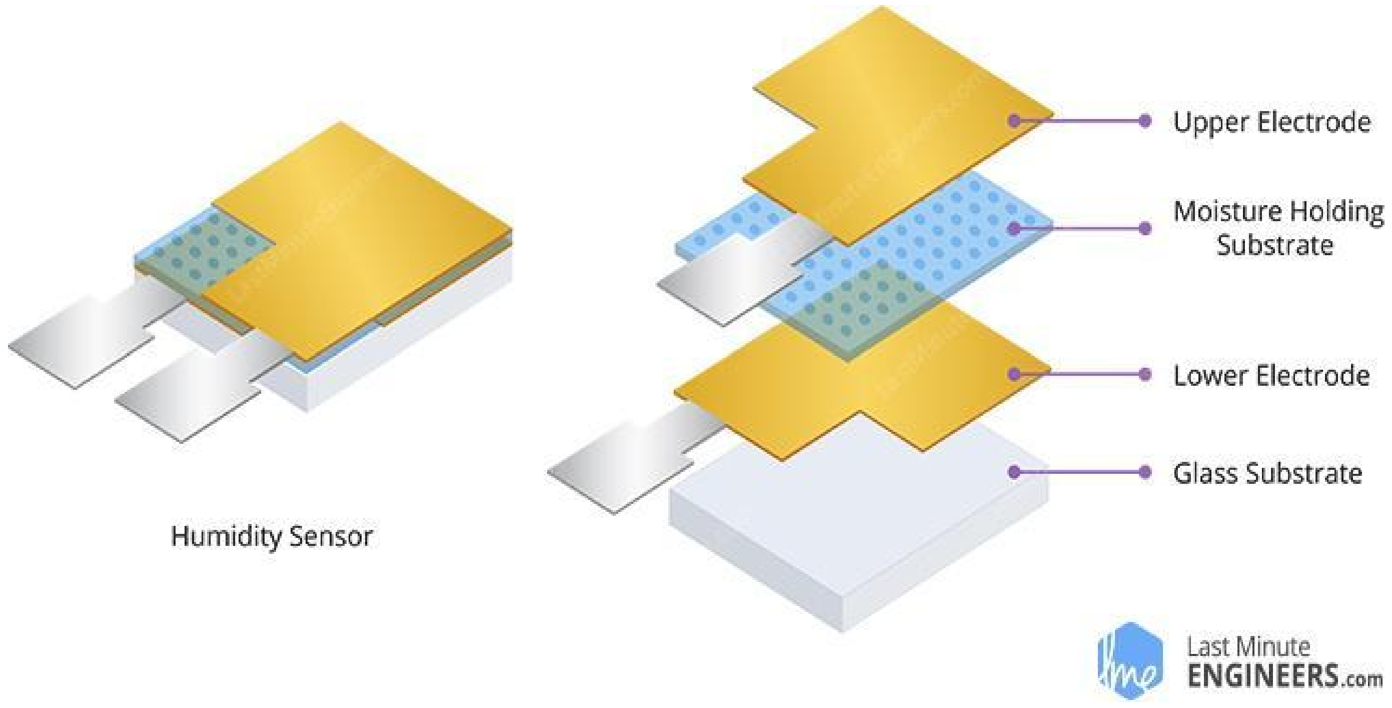
DHT11 is used to measure the temperature and humidity.

How does DHT11 works?

Inside the case, on the sensing side, there is a humidity sensing component along with a NTC temperature sensor (or thermistor)



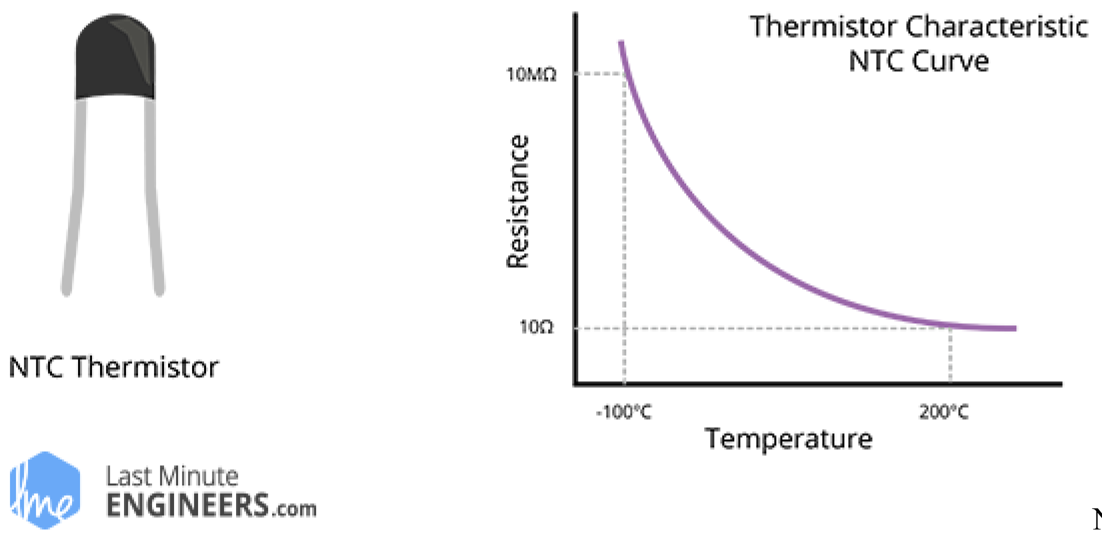
Humidity sensing component is used, of course to measure humidity, which has two electrodes with moisture holding substrate (usually a salt or conductive plastic polymer) sandwiched between them. The ions are released by the substrate as water vapor is absorbed by it, which in turn increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.



Internal Structure of Humidity Sensor

Besides, they consist of a NTC temperature sensor/Thermistor to measure temperature. A thermistor is a thermal resistor – a resistor that changes its resistance with temperature. Technically, all resistors are thermistors – their resistance changes slightly with temperature – but the change is usually very very small and difficult to measure.

Thermistors are made so that the resistance changes drastically with temperature so that it can be 100 ohms or more of change per degree! The term “NTC” means “Negative Temperature Coefficient”, which means that the resistance decreases with increase of the temperature.

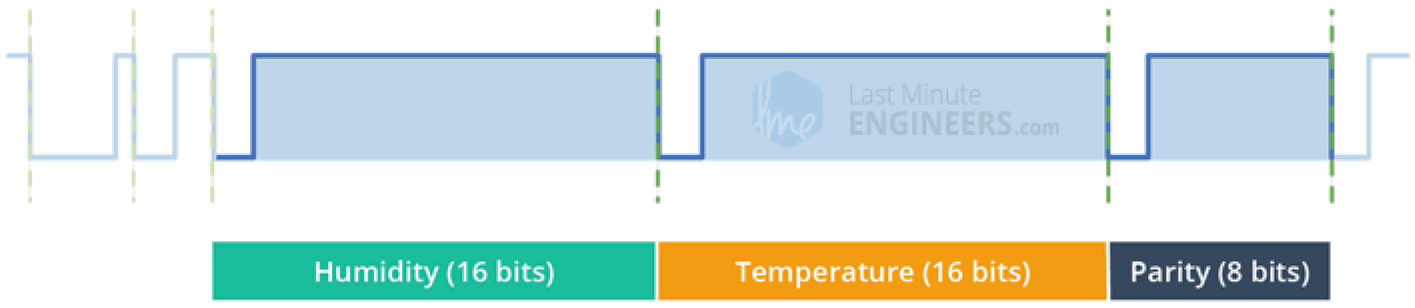


NTC

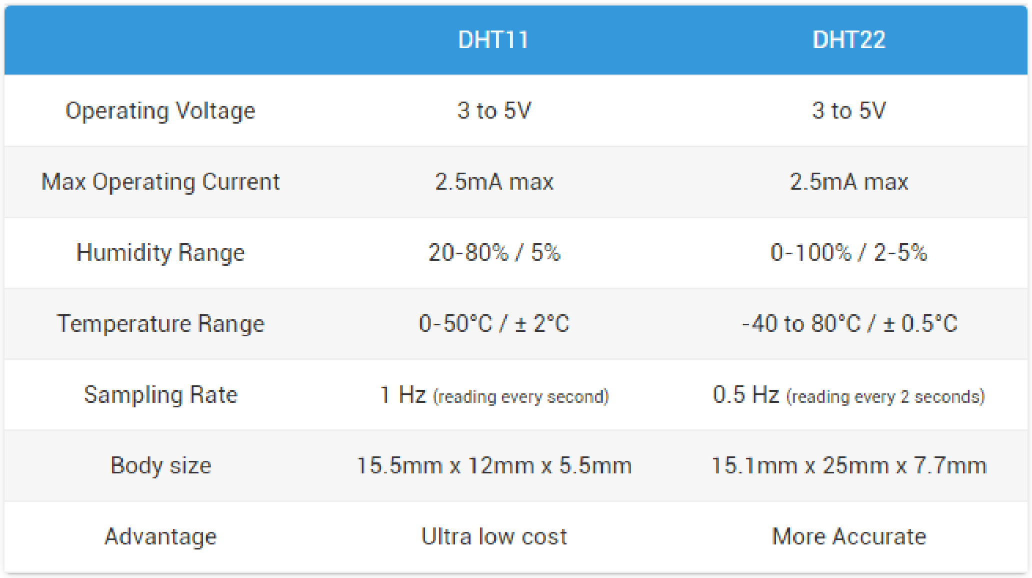
Thermistor with Characteristic Curve

On the other side, there is a small PCB with an 8-bit SOIC-14 packaged IC. This IC measures and processes the analog signal with stored calibration coefficients, does analog to digital conversion and spits out a digital signal with the temperature and humidity.

DHT11 outputs a string of 40 bits data consisting of relative temperature and humidity values. The 40 bit data contains 16 bits of humidity, 16 bits of temperature and last 8 bits represent parity bits (a simplest form of error detecting code) with Most Significant Bit (MSB) first. The corresponding timing diagram is shown as below.



DHT11 V/S DHT22

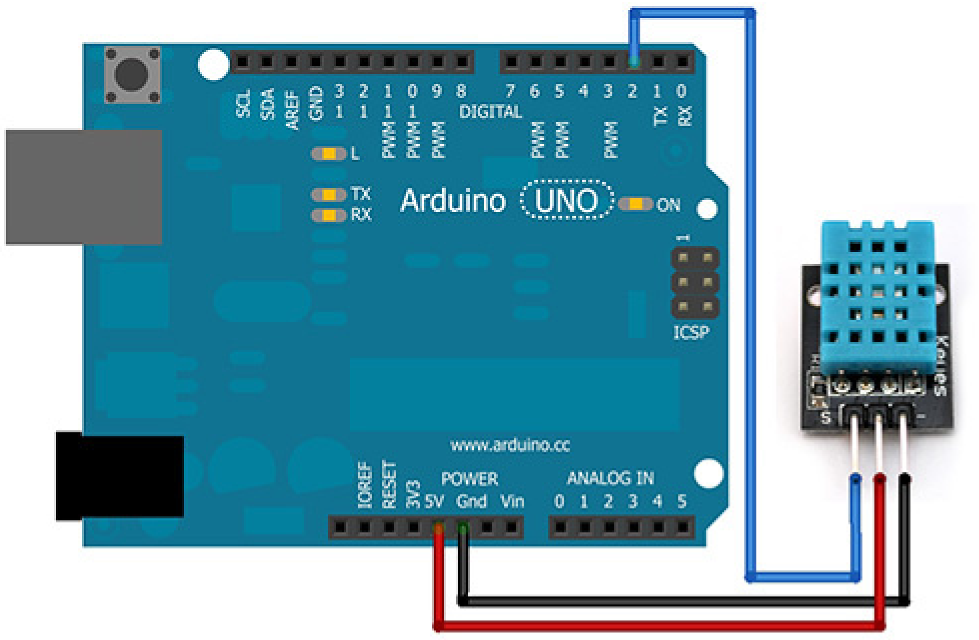


	DHT11	DHT22
Operating Voltage	3 to 5V	3 to 5V
Max Operating Current	2.5mA max	2.5mA max
Humidity Range	20-80% / 5%	0-100% / 2-5%
Temperature Range	0-50°C / ± 2°C	-40 to 80°C / ± 0.5°C
Sampling Rate	1 Hz (reading every second)	0.5 Hz (reading every 2 seconds)
Body size	15.5mm x 12mm x 5.5mm	15.1mm x 25mm x 7.7mm
Advantage	Ultra low cost	More Accurate

CODE AND HARDWARE CONNECTIONS FOR DHT11

CODE 1: Measure temperature and humidity using DHT11 and display it on Serial Monitor

HARDWARE:



CODE:

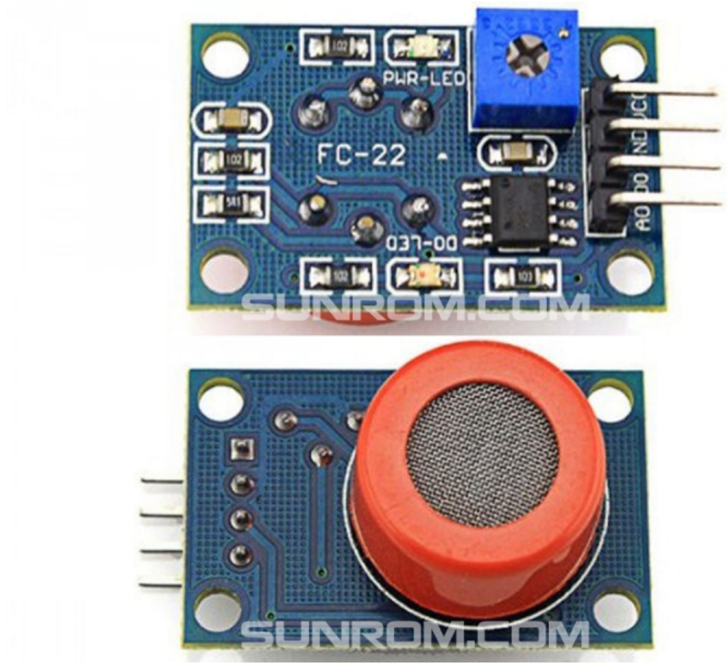
```
#include "DHT.h"
DHT dht(2,DHT11);
void setup() {
  dht.begin();
  Serial.begin(9600);
  }
void loop() {
 float t=dht.readTemperature();
 float h=dht.readHumidity();
 Serial.print(t);
 Serial.print("  ");
 Serial.println(h);
}
```

CODE 2: Measure temperature and humidity using DHT11 and if the temperature is greater than 27 degree celcius led connected should turn ON else led should remain OFF.

```
#include "DHT.h"
DHT dht(2,DHT11);
void setup() {
  dht.begin();
  Serial.begin(9600);
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
}
void loop() {
 float t=dht.readTemperature();
 float h=dht.readHumidity();
 Serial.print(t);
 Serial.print("  ");
 Serial.println(h);
if(t>28)

 {
  digitalWrite(13,HIGH);
 }
 else
 {
  digitalWrite(13,LOW);
 }
}
```

ALCOHOL SENSOR (MQ3)

MQ3 Alcohol Sensor is alcohol detector sensor which is used to detect the alcohol concentration on your breath.This sensor provides an analog resistive output based on alcohol concentration. When the alcohol gas exists, the sensor's conductivity gets higher along with the gas concentration rising.It is suitable for various applications of detecting alcohol at different concentration.It is widely used in domestic alcohol gas alarm, industrial alcohol gas alarm, and portable alcohol detector.



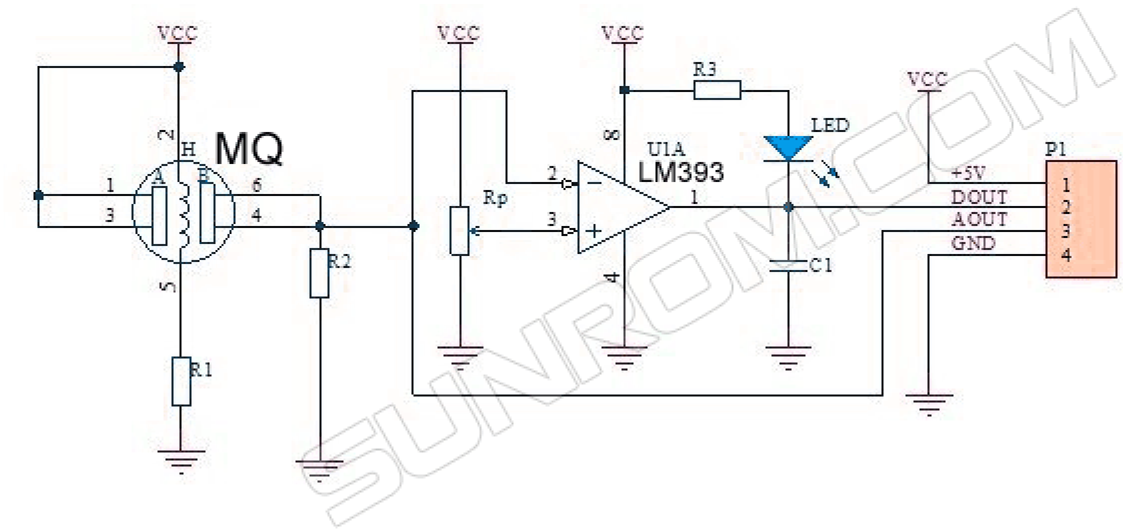
Alcohol Sensor module can provide digital output i.e 0 or 1

Also it can provide analog output depending on the concentration of Alcohol and that varies from 0v to 5volt. As the concentration of alcohol increases analog output voltage increases. We measure the analog voltage by connecting the AO pin of sensor on analog pin of arduino.

Features

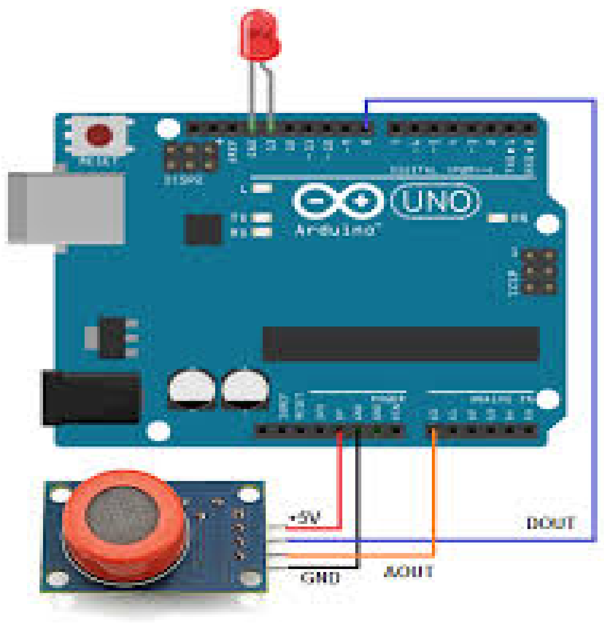
- 5V operation
- Simple to use
- LEDs for output and power
- Output sensitivity adjustable
- Analog output 0V to 5V
- Digital output 0V or 5V
- Low Cost
- Fast Response
- Stable and Long Life
- Good Sensitivity to Alcohol Gas
- Both Digital and Analog Outputs
- On-board LED Indicator

MQ3 Sensor internal circuitry(only for knowledge not required for examination point of view)



HARDWARE CONNECTIONS AND CODE

HARDWARE

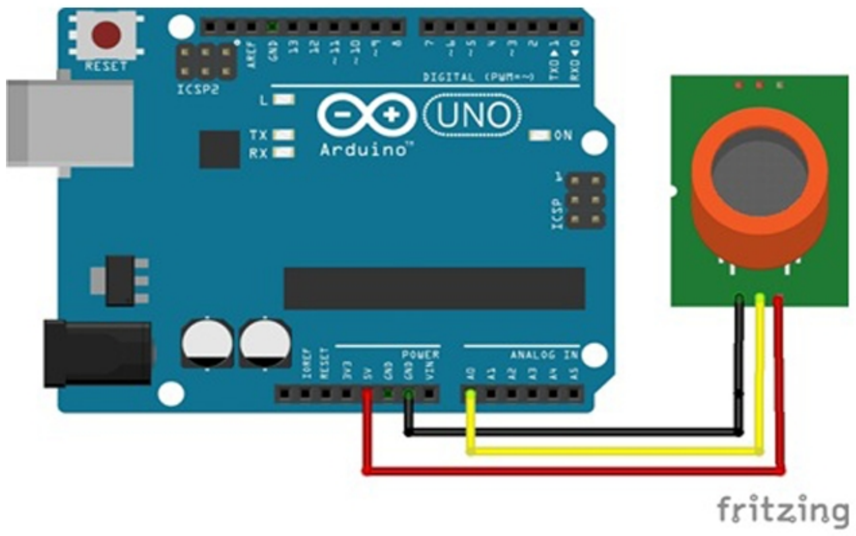


CODE:

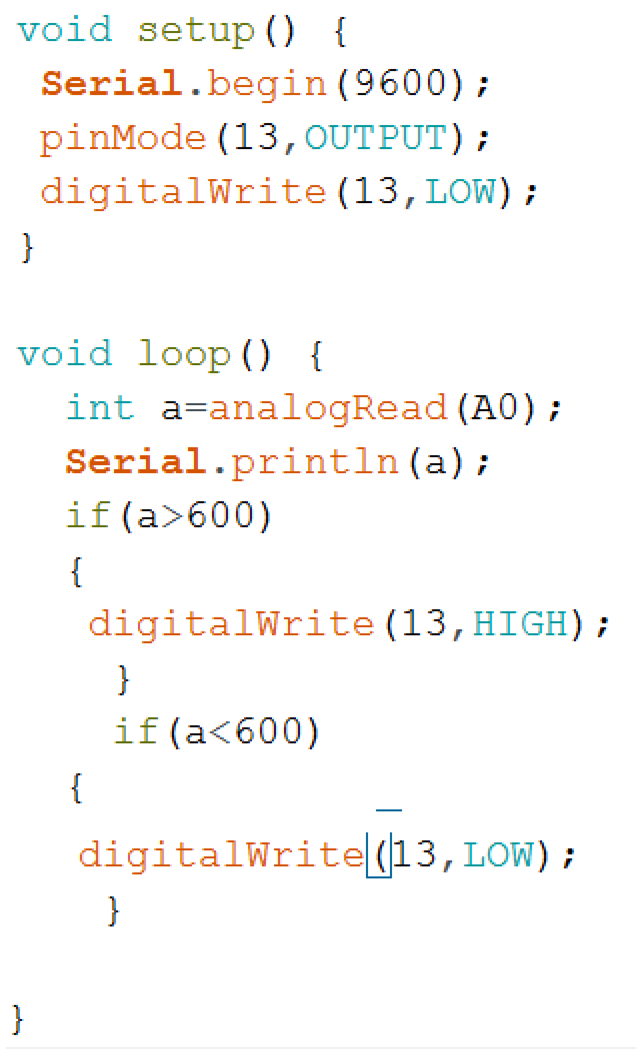
```
void setup() {
  pinMode(8,INPUT);
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
 }

void loop() {
  int a=digitalRead(8);
  if(a==0)
  {
    digitalWrite(13,HIGH);
  }
  else
  {
    digitalWrite(13,LOW);
    }
}
```

CODE 2: To take analog readings of alohol sensor and make the led on when alcohol concentration is high and turn it off when the concentration is low

HARDWARE:



CODE:



```
void setup() {
 Serial.begin(9600);
 pinMode(13,OUTPUT);
 digitalWrite(13,LOW);
}

void loop() {
   int a=analogRead(A0);
   Serial.println(a);
   if(a>600)
   {
    digitalWrite(13,HIGH);
     }
     if(a<600)
   {
    digitalWrite(13,LOW);
     }

}
```

ESP8266 CODES

CODE1: To upload the readings of DHT11 sensor on thing speak using ESP8266.

```
#include <DHT.h>
DHT dht(2, DHT11); // Initialize the DHT sensor
#define SSID "ZAKBEE"     // "SSID-WiFiname"
#define PASS "aaiza123"       // "password"
#define IP "184.106.153.149"// thingspeak.com ip
String msg = "GET /update?key=3EAEVSC69ST019IZ"; //change it with your key...
float t;
float h;
int error;
void setup()
{
  Serial.begin(115200); //or use default 115200.

  Serial.println("AT");
  delay(5000);
  if(Serial.find("OK")){
    connectWiFi();
  }
}

void loop(){
  //Read temperature and humidity values from DHT sensor:
  start: //label
  error=0;
  t = dht.readTemperature();
  h= dht.readHumidity();
 updateTemp();
  if (error==1){
    goto start; //go to label "start"
  }
  delay(2000); //Update every 1 hour
}

void updateTemp(){
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += IP;
  cmd += "\",80";
  Serial.println(cmd);
  delay(2000);
  if(Serial.find("Error")){
    return;
  }
  cmd = msg ;
  cmd += "&field1=";    //field 1 for temperature

  cmd += t;
  cmd += "&field2=";  //field 2 for humidity
  cmd += h;
  cmd += "\r\n";
  Serial.print("AT+CIPSEND=");
  Serial.println(cmd.length());
  if(Serial.find(">")){
    Serial.print(cmd);
  }
  else{
    Serial.println("AT+CIPCLOSE");
    //Resend...
    error=1;
  }
}
```

```
boolean connectWiFi(){
  Serial.println("AT+CWMODE=1");
  delay(2000);
  String cmd="AT+CWJAP=\"";
  cmd+=SSID;
  cmd+="\",\"";
  cmd+=PASS;
  cmd+="\"";
  Serial.println(cmd);
  delay(5000);
  if(Serial.find("OK")){
    return true;
  }else{
    return false;
  }

}
```

CODE2: To upload the readings of Ultrasonic sensor on thing speak using ESP8266.

```
#include <stdlib.h>
const int trigPin = 8;
const int echoPin = 9;
#define SSID "Harshita"     // "SSID-WiFiname"
#define PASS "harshita"       // "password"
#define IP "184.106.153.149"// thingspeak.com ip
String msg = "GET /update?key=WUOTE6C1004S7TB3"; //change it with your key...
int error;
 long duration, inches, cm;
long microsecondsToCentimeters(long microseconds)
{
  return microseconds / 29 / 2;
}
void setup()
{
   Serial.begin(115200); //or use default 115200.
   Serial.println("AT");
   delay(5000);

   if(Serial.find("OK")){
     connectWiFi();
   }
}

void loop(){
   start: //label
error=0;
 pinMode(trigPin, OUTPUT);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = microsecondsToCentimeters(duration);
  updateTemp();
  //Resend if transmission is not completed
  if (error==1){
    goto start; //go to label "start"
  }

  delay(1000); //Update every 1 hour
}

void updateTemp(){
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += IP;
  cmd += "\",80";
  Serial.println(cmd);
  delay(2000);
  if(Serial.find("Error")){
    return;
  }
  cmd = msg ;
  cmd += "&field1=";    //field 1 for temperature
  cmd += String(cm);;
  cmd += "\r\n";
  Serial.print("AT+CIPSEND=");
  Serial.println(cmd.length());
```

```
    if(Serial.find(">")){
      Serial.print(cmd);
    }
    else{
      Serial.println("AT+CIPCLOSE");
      //Resend...
      error=1;
    }
  }


  boolean connectWiFi(){
    Serial.println("AT+CWMODE=1");
    delay(2000);
    String cmd="AT+CWJAP=\"";
    cmd+=SSID;
    cmd+="\",\"";
    cmd+=PASS;
    cmd+="\"";
    Serial.println(cmd);
    delay(5000);
    if(Serial.find("OK")){
      return true;
    }else{
      return false;
    }
  }
```