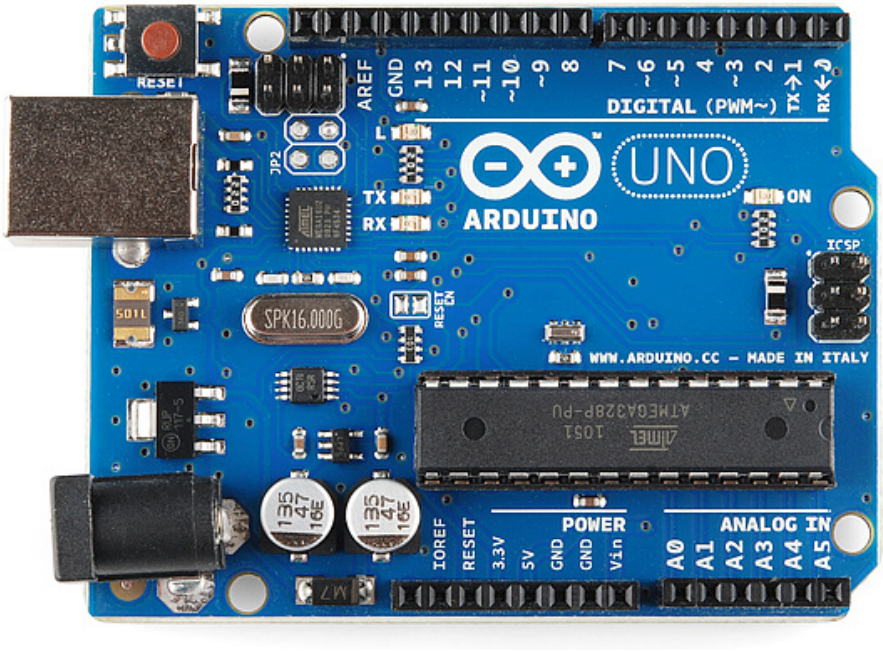
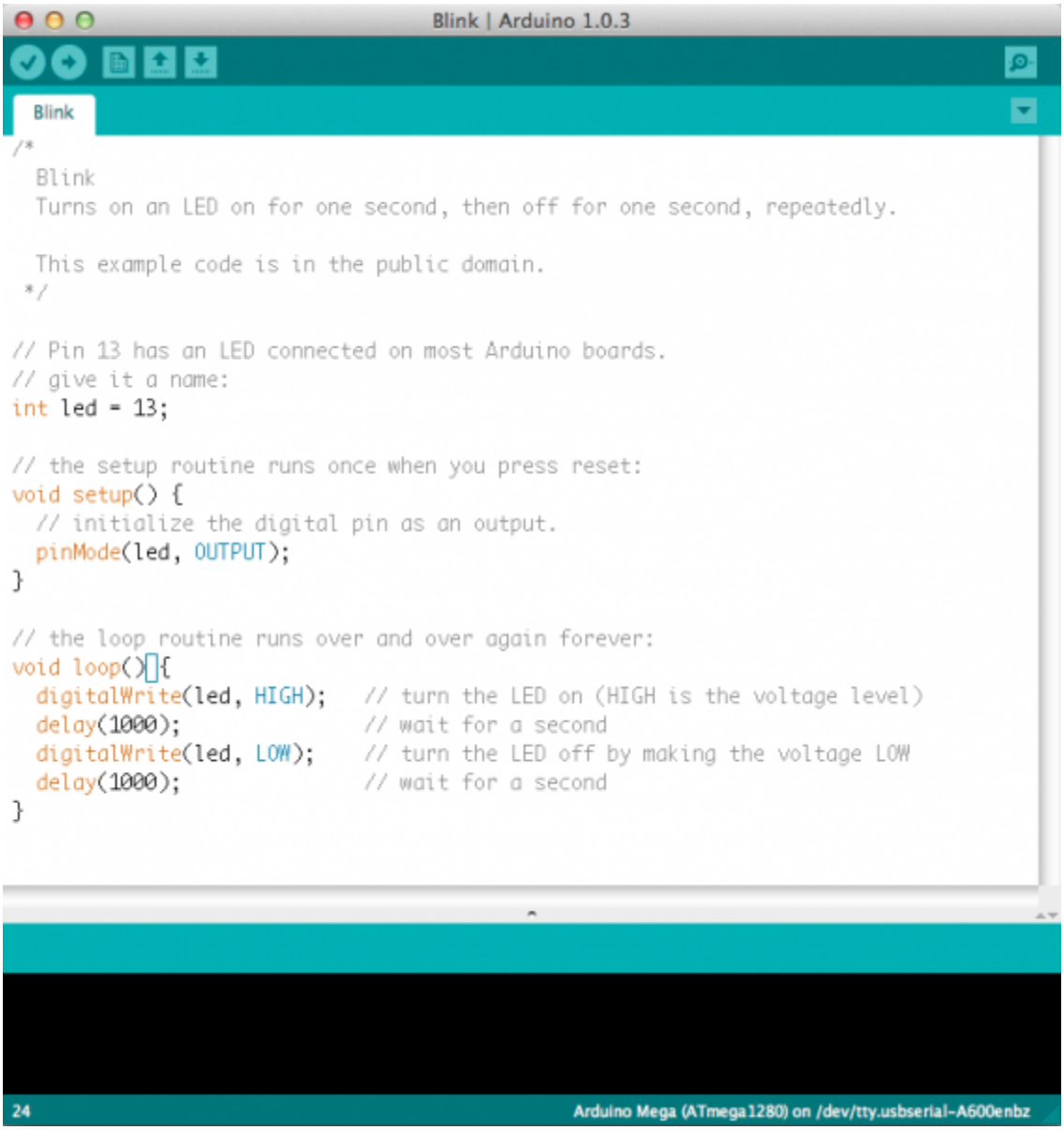
Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.



This is an Arduino Uno

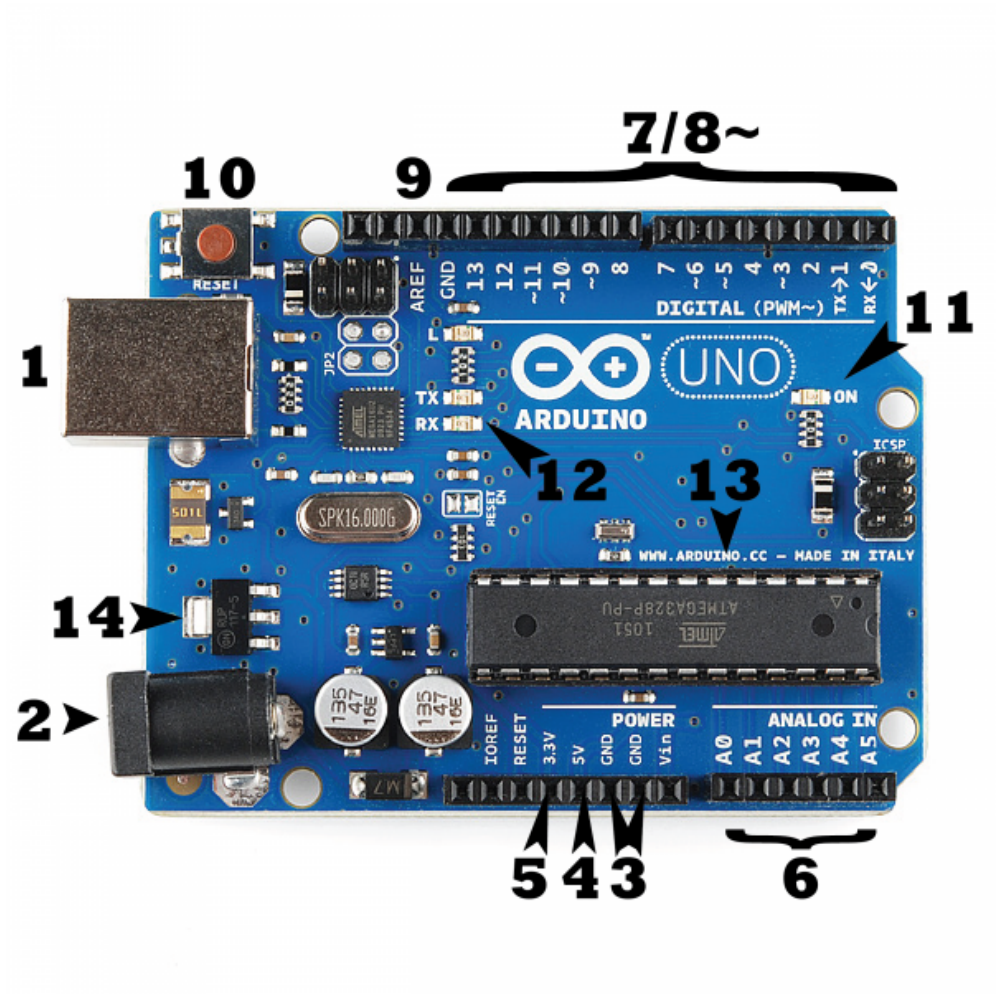
The Uno is one of the more popular boards in the Arduino family and a great choice for beginners. We'll talk about what's on it and what it can do later in the tutorial.



This is a screenshot of the Arduino IDE.

What's on the board?

There are many varieties of Arduino boards (explained on the next page) that can be used for different purposes. Some boards look a bit different from the one below, but most Arduinos have the majority of these components in common:



Power (USB / Barrel Jack)

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 or a wall power supply (like this) that is terminated in a barrel jack. In the picture above the USB connection is labeled **(1)** and the barrel jack is labeled **(2)**.

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 Installing and Programming Arduino tutorial.

Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjuction with abreadboard and some wire. They usually have black plastic ‘headers’ that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- **GND (3)**: Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- **5V (4) & 3.3V (5)**: As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- **Analog (6)**: 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 (like a temperature sensor) and convert it into a digital value that we can read.
- **Digital (7)**: 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).
- **PWM (8)**: 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). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).
- **AREF (9)**: 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.

Reset Button

Just like the original Nintendo, the Arduino has a reset button **(10)**. Pushing it 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. Unlike the original Nintendo however, blowing on the Arduino doesn’t usually fix any problems.

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’ **(11)**. 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!

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 **(12)**. 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).

Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit **(13)**. 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.

Voltage Regulator

The voltage regulator **(14)** 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.

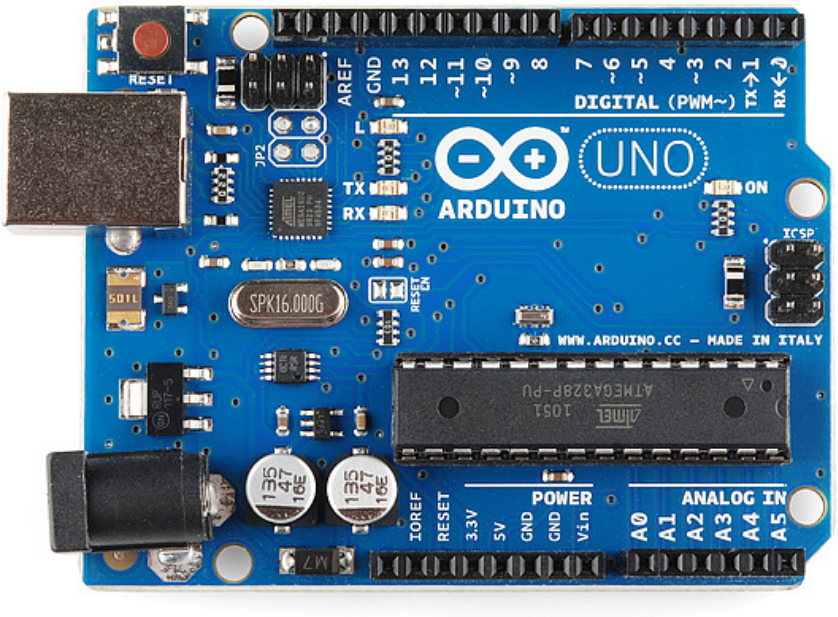
The Arduino Family

Arduino makes several different boards, each with different capabilities. In addition, part of being open source hardware means that others can modify and produce derivatives of Arduino boards that provide even more form factors and functionality. If you're not sure which one is right for your project, check this guide for some helpful hints. Here are a few options that are well-suited to someone new to the world of Arduino:

Arduino Uno (R3)

The Uno is a great choice for your first Arduino. It's got everything you need to get started, and nothing you don't. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6

analog inputs, a USB connection, a power jack, a reset button and more. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

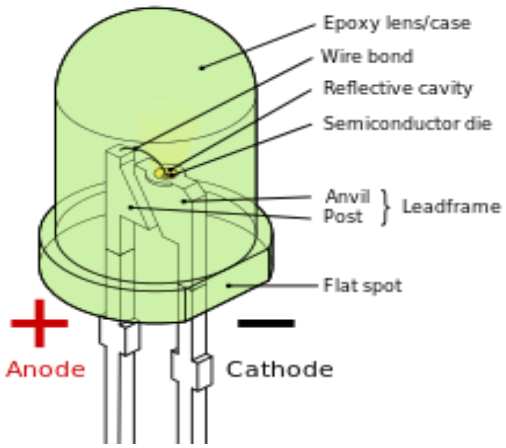


LED INTERFACING WITH ARDUINO

LED (Light Emitting Diode)

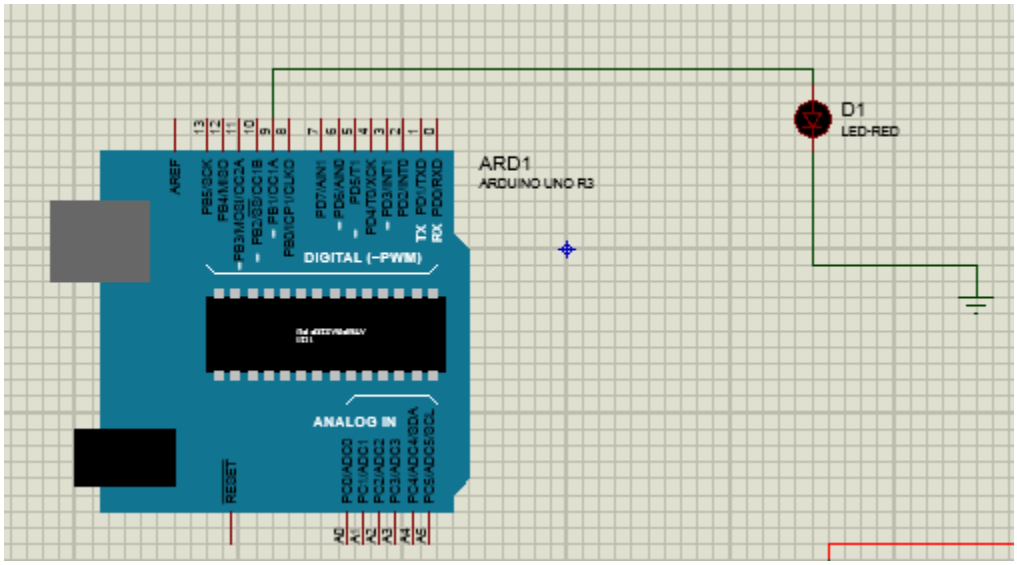
A light-emitting diode (LED) is a two-lead semiconductor light source. It is a p–n junctiondiode, which emits light when activated. When a suitable voltage is applied to the leads,electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor.

An LED is often small in area (less than 1 mm^2) and integrated optical components may be used to shape its radiation pattern.



1. INTERFACING SNGLE LED

HARDWARE CONNECTIONS WITH ARDUINO

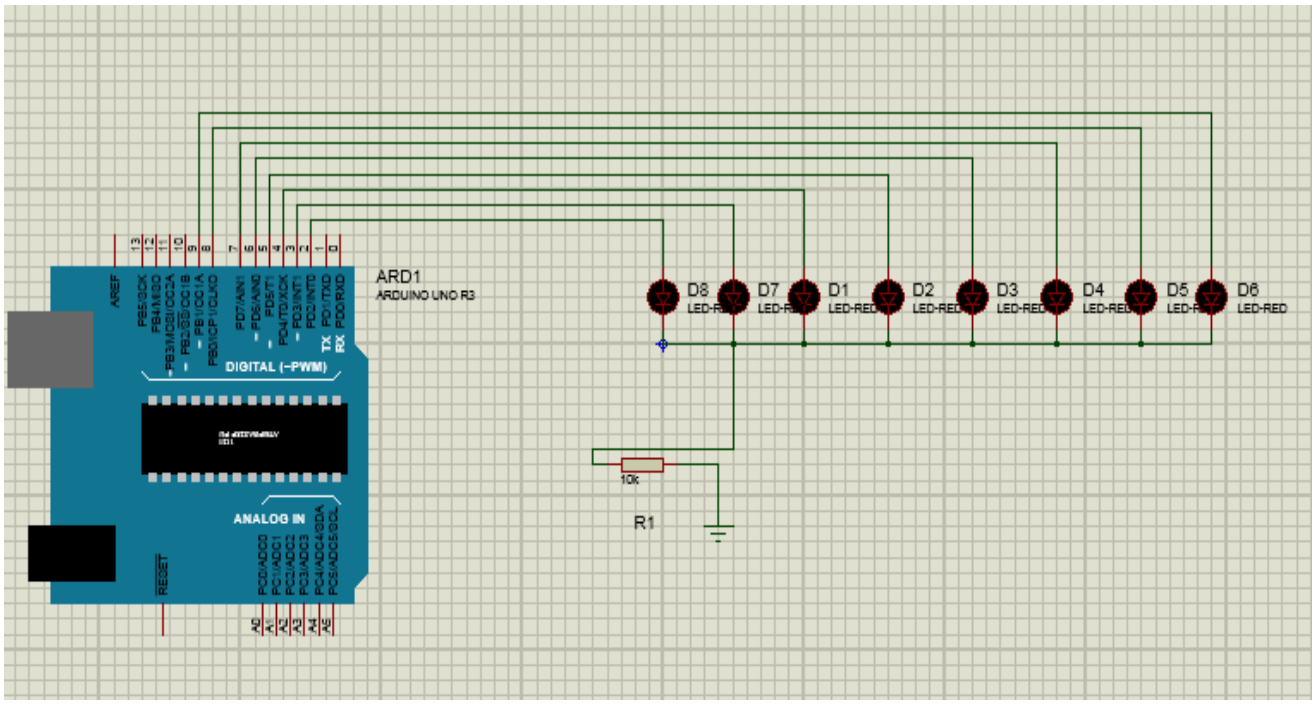


CODE TO BLINK SINGLE LED:-

```
void setup() {
  // put your setup code here, to run once:
pinMode(7,OUTPUT);
}

void loop()
 digitalWrite(7,HIGH);
 delay(1000);
 digitalWrite(7,LOW);
 delay(1000);

}
```

2.INTERFACING 8 LED’s WITH ARDUINO TO GENERATE DIFFERENT PATTERNS

HARDWARE CONNECTIONS



CODE TO GENERATE DIFFERENT PATTERNS:-

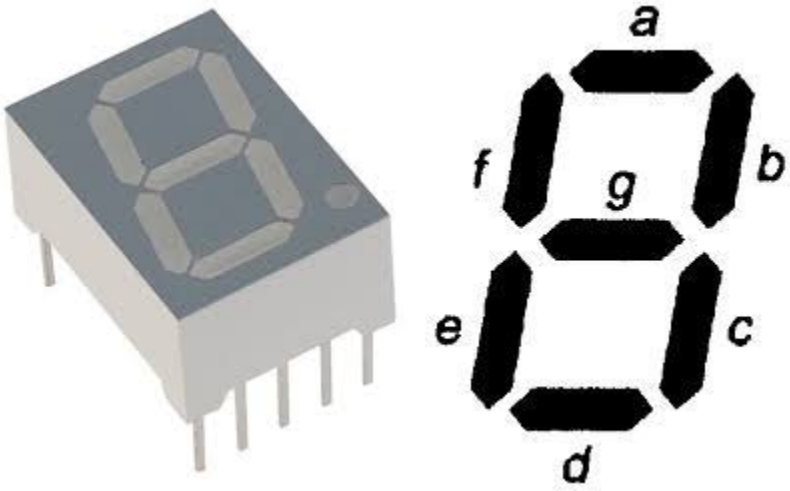
```
unsigned char arr[8][8]={{1,1,1,1,1,1,1,1},
{0,0,0,0,0,0,0,0},
{1,1,1,1,0,0,0,0},
{1,0,1,0,1,0,1,0},
{0,0,1,1,0,0,1,1},
{0,0,0,0,1,1,1,1},
{0,1,0,1,0,1,0,1},
{1,1,1,0,0,0,0,1},

  };

 int i=0,j=0,k=2;
void setup() {
for(i=2;i<10;i++)
{
  pinMode(i,OUTPUT);
  }


void loop() {
  for(i=0;i<8;i++)
  {
    k=2;
for(j=0;j<8;j++)
{
 digitalWrite(k,arr[i][j]);
  }
    delay(500);
    }

}
```

SEVEN SEGMENT INTERFACING WITH ARDUINO

Seven Segment Display is used in a variety of embedded application to display the numbers. Arduino 7 seg display is used in a different application like a digital counter, digital watch, lift, and oven etc.

We use seven segment display that will display number from 0-9 in a single segment.

The seven segment is constructed using the 8 led’s connected in a specific pattern.

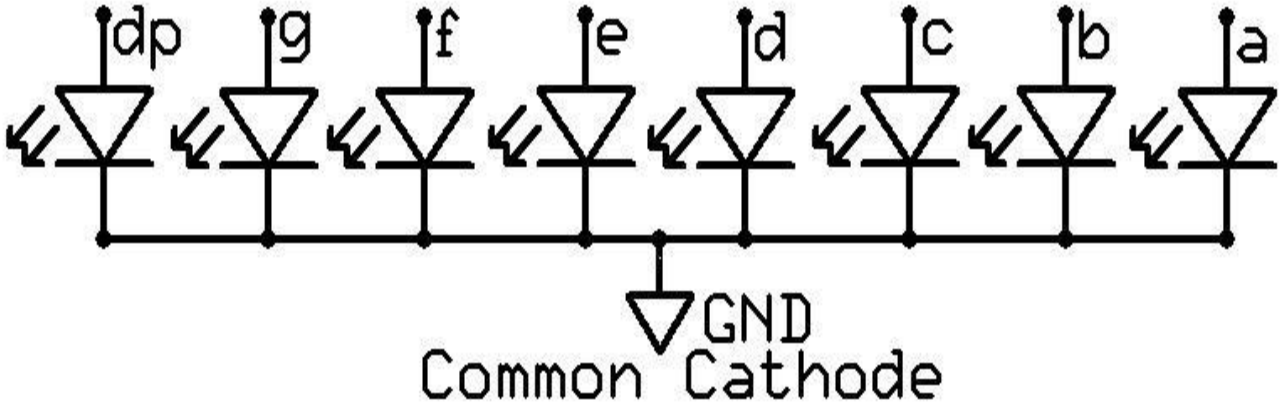


Types of Seven Segment Display

- Common Cathode
- Common Anode

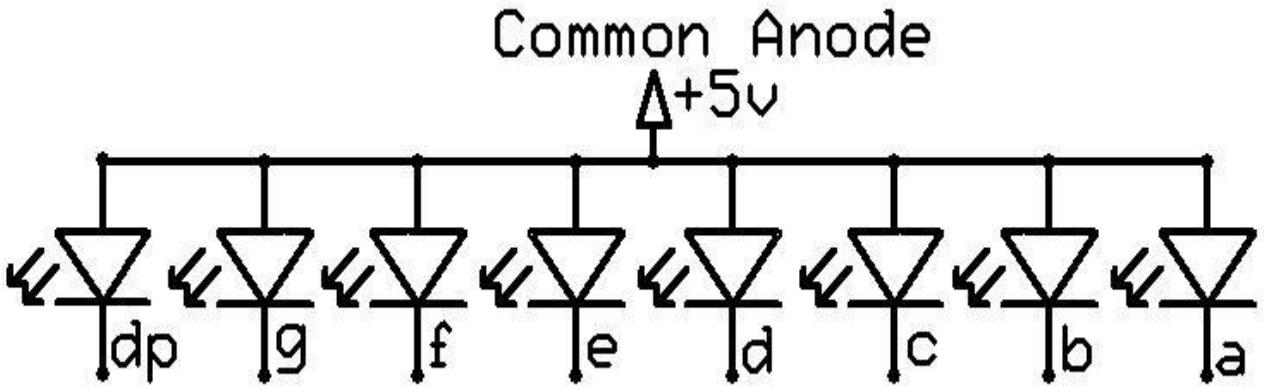
Common Cathode Seven Segment Display

In this type, the cathode of all 8-led’s is connected together and GND is applied to it. And all anode of 8-led’s assigned name a,b,c,d,e,f,g and dp. To glow a specific led in the segment we apply a logic HIGH signal to the segment pin.



Common Anode Seven Segment Display

On the other side, it is just opposite to common cathode. All the anode of 8-led's are connected to each other and to glow a particular led we apply logic LOW to respective pin.



7 Segment Display Truth Table Common Cathode

Digit	Hex Value	dp	g	f	e	d	c	b	a
"0"	0x3F	0	0	1	1	1	1	1	1

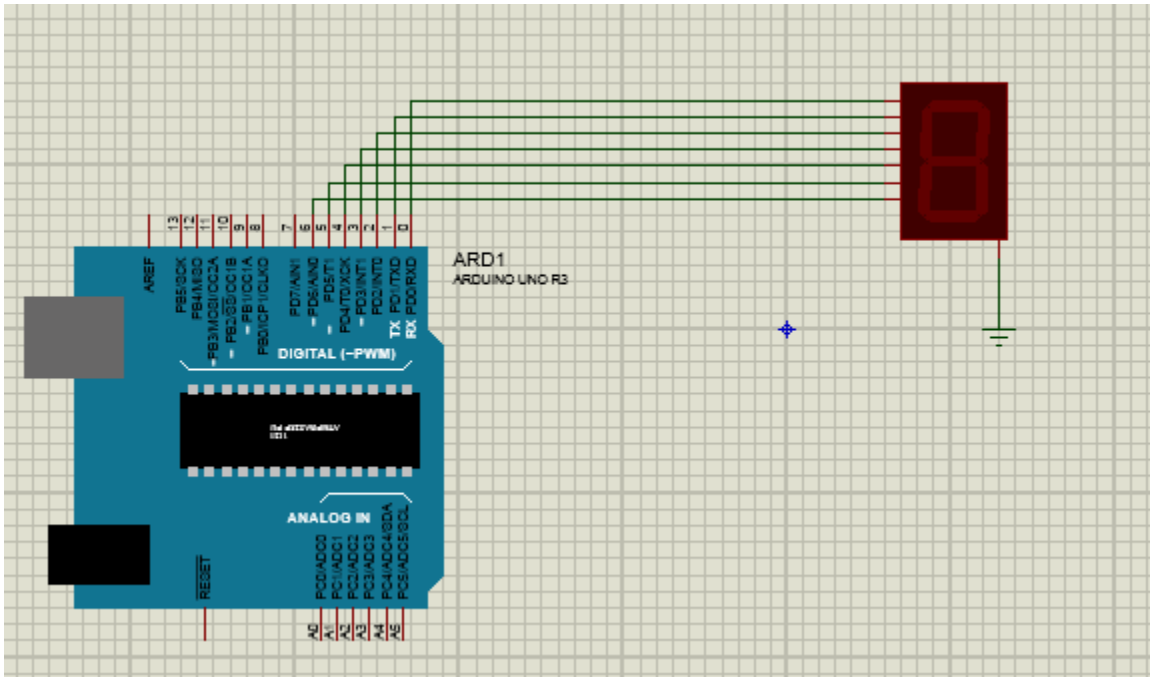
Digit	Hex Value	dp	g	f	e	d	c	b	a
“1”	0x06	0	0	0	0	0	1	1	0
“2”	0x5B	0	1	0	1	1	0	1	1
“3”	0x4F	0	1	0	0	1	1	1	1
“4”	0x66	0	1	1	0	0	1	1	0
“5”	0x6D	0	1	1	0	1	1	0	1
“6”	0x7D	0	1	1	1	1	1	0	1
“7”	0x07	0	0	0	0	0	1	1	1
“8”	0x7F	0	1	1	1	1	1	1	1
“9”	0x6F	0	1	1	0	1	1	1	1

How to display numbers on 7 segment display

Hardware Required

- Arduino Uno
- Common Cathode Seven Segment Display
- Resistor 220 ohm
- Breadboard

HARDWARE CONECTIONS OF SEGMENT WITH ARDUINO



CODE TO DISPLAY COUNT FROM 0-9

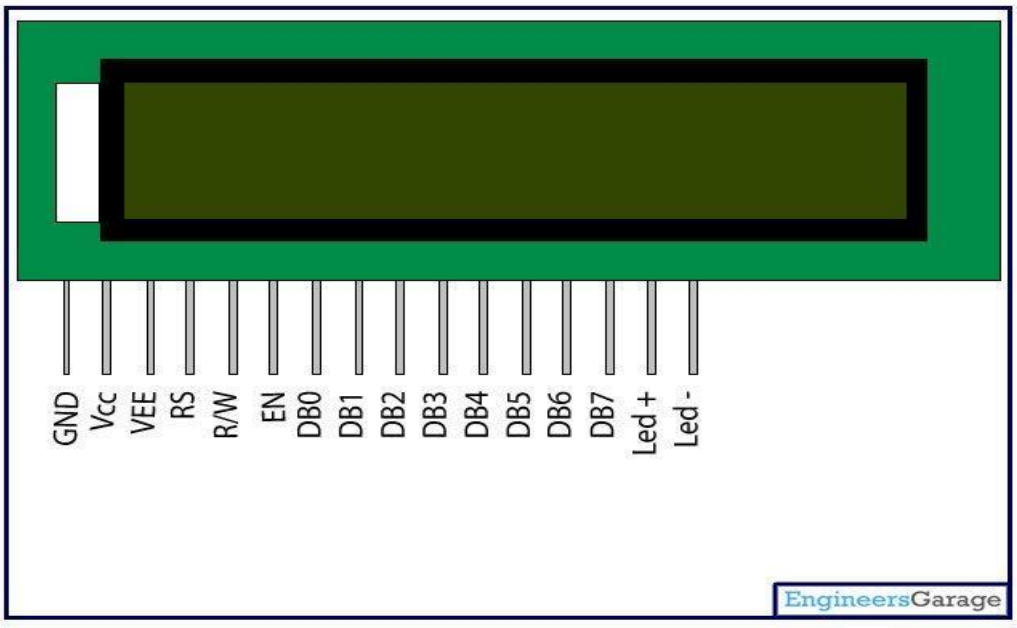
```
unsigned char arr[10][7]={{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0},
{1,1,0,1,1,0,1},
{1,1,1,1,0,0,1},
{0,1,1,0,0,1,1},
{1,0,1,1,0,1,1},
{1,0,1,1,1,1,1},
{1,1,1,0,0,0,0},
{1,1,1,1,1,1,1},
{1,1,1,1,0,1,1}
  };

 int i=0,j=0;
void setup() {
for(i=0;i<7;i++)
{
  pinMode(i,OUTPUT);
  }

}

void loop() {
  for(i=0;i<10;i++)
  {
for(j=0;j<7;j++)
{

 digitalWrite(j,arr[i][j]);
  }
    delay(500);
    }

}
```

LCD INTERFACING WITH ARDUINO

LCD 16*2–

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special &evencustom characters (unlike in seven segments), animations and so on.
A **16x2 LCD** means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.
The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD.

Pin Diagram:



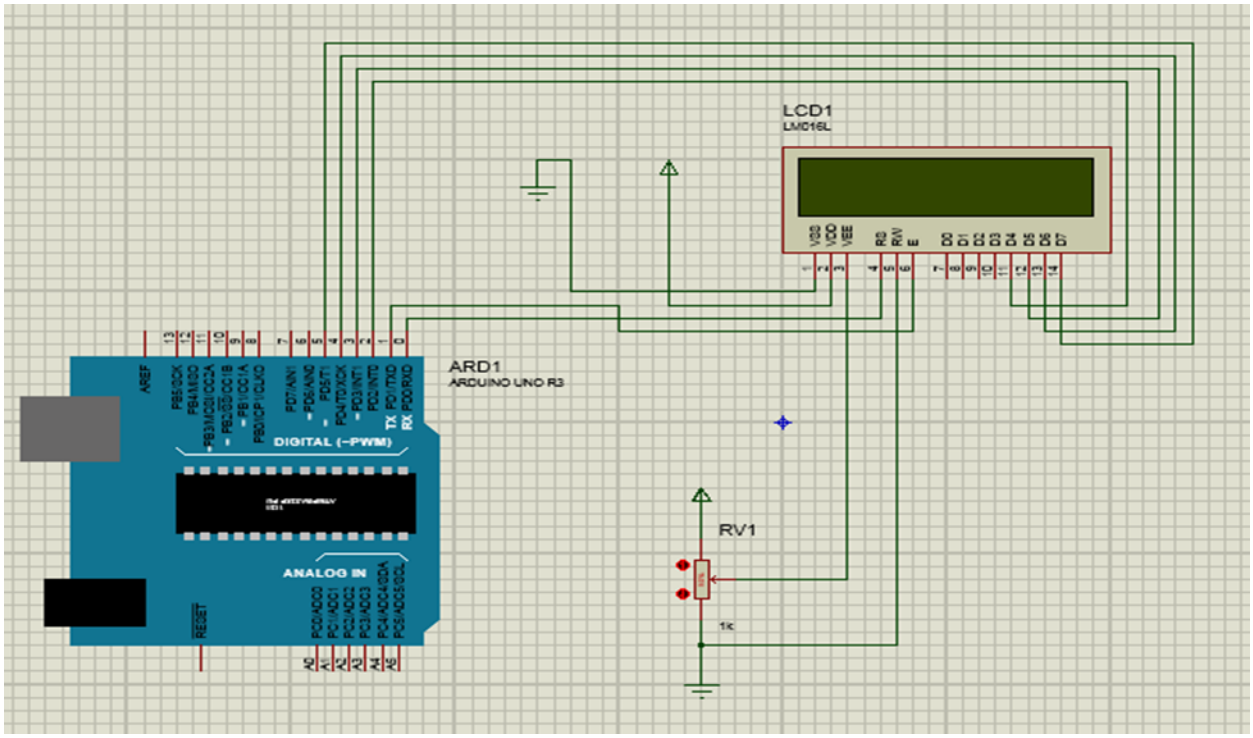
Pin Description:

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

LCD Command Codes

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5×7 matrix

HARDWARE CONNECTIONS FOR LCD WITH ARDUINO



CODE TO PRINT DATA ON LCD

```
 #include<LiquidCrystal.h>                  // header file for lcd

LiquidCrystal lcd(0,1,2,3,4,5);    ////CONNECTIONS OF RS,EN,D4,D5,D6,D7 RESPECTIVELY
void setup() {
 lcd.begin(16,2);                 ////function to tell how many rows and columns are there in our lcd
 lcd.setCursor(5,0);                //// function to set position where data is to be displayed
 lcd.print("hello");                  ///function to print data on lcd
delay(2000);
lcd.clear();                           //// function to clear data on lcd
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

BLUETOOTH INTERFACING WITH ARDUINO

HC-05 module is an easy to use **Bluetooth SPP (Serial Port Protocol) module**,designed for transparent wireless serial connection setup.The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication.

The Bluetooth module HC-05 is a MASTER/SLAVE module.By default the factory setting is SLAVE.The Role of the module (Master or Slave) can be configured only by AT COMMANDS.The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections.Master module can initiate a connection to other devices.

Software Features

- Slave default Baud rate: 9600, Data bits:8, Stop bit:1,Parity:No parity.
- Auto-connect to the last device on power as default.

- Permit pairing device to connect as default.
- Auto-pairing PINCODE:”1234” as default.

Pin Description

The HC-05 Bluetooth Module has 6pins. They are as follows:

ENABLE:

When enable is pulled **LOW**, the module is disabled which means the module will **not turn on** and it **fails to communicate**.When enable is **left open or connected to 3.3V**, the module is enabled i.e the module **remains on**and **communication also takes place**.

Vcc:

Supply Voltage 3.3V to 5V

GND:

Ground pin

TXD & RXD:

These two pins acts as an UART interface for communication

STATE:

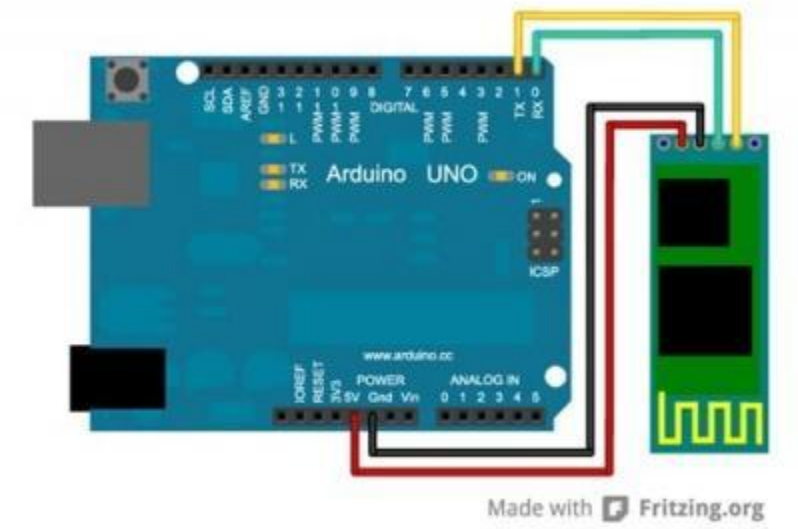
It acts as a status indicator.When the module is **not connected to / paired** with any other bluetooth device,signal goes **Low**.At this **low state**,the **led flashes continuously** which denotes that the module is **not paired** with other device.When this module is **connected to/paired** with any other bluetooth device,the signal goes **High**.At this **high state**,the **led blinks with a constant delay** say for example 2s delay which indicates that the module is **paired**.

BUTTON SWITCH:

This is used to switch the module into AT command mode.To enable AT command mode,press the button switch for a second.With the help of AT commands,the user can change the parameters of this module but only when the module is not paired with any other BT device.If the module is connected to any other bluetooth device, it starts to communicate with that device and fails to work in AT command mode.

Hardware Connections

As we know that Vcc and Gnd of the module goes to Vcc and Gnd of Arduino.The TXD pin goes to RXD pin of Arduino and RXD pin goes to TXD pin of Arduinoi.e(digital pin 0



PRACTICALS DONE

1) **Connect arduino Board with any android device , to send and receive data and control an led from android device**

2) **Design Bluetooth based notice board.**

To make a link between your Arduino and bluetooth , do the following :

1) Go to the bluetooth icon , right click and select Add a Device

2) Search for new device , Our bluetooth module will appear as HC-05 , and add it

3) The pairing code will be 1234 .

4)after make a pairing , we can now program the arduino and upload a sketch

TO START WITH, FIRST OF ALL INSTALL ANY BLUETOOTH BASED APPLICATION FROM PLAY STORE. FOR EXAMPLE :- we used ARDUINO BLUETOOTH CONTROLLER and BLUETOOTH TERMINAL.

UPLOAD THE FOLLOWING CODE TO CHECK WHAT BLUETOOTH RECEIVES FROM A APPLICATION

Code1

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

}

void loop() {
if(Serial.available()>0)
  {
char a=Serial.read();
Serial.print(a);
    }

}
```

CODE TO MAKE AN LED TURN ON AND OFF

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

void loop() {
if(Serial.available()>0)
 {
char a=Serial.read();
Serial.print(a);
if(a=='A')
   {
digitalWrite(13,HIGH);
     }
if(a=='B')
   {
digitalWrite(13,LOW);
     }
}
}
```

BLUETOOTH BASED NOTICE BOARD CODE

```
#include<LiquidCrystal.h>

LiquidCrystallcd(12,11,9,8,7,6);

int count;

charblankarr[32];

void setup(){
```

```
Serial.begin(9600);
lcd.begin(16,2);
lcd.setCursor(0,0);
lcd.print(" BLUTOOTH BASED ");
lcd.setCursor(0,1);
lcd.print("  NOTICE BOARD");
delay(3000);
lcd.clear();
}
void loop()
{
if(Serial.available()>0)
  {
lcd.clear();
count=0;
while(Serial.available() )         // Read 12 characters and store them in input array
     {
blankarr[count] = Serial.read();
count++;
delay(5);
```

```
Serial.print(blankarr[count]);
    }
    }
if(count<16)
    {
lcd.setCursor(0,0);
for(char i=0;i<count;i++)
   {
lcd.print(blankarr[i]);
   }
    }

if(count>16)
    {
lcd.setCursor(0,0);
for(char i=0;i<16;i++)
   {
lcd.print(blankarr[i]);
   }
lcd.setCursor(0,1);
for(char i=16;i<count;i++)
```

```
  {
lcd.print(blankarr[i]);
  }
  }}
```

GSM INTERFACING WITH ARDUINO

GSM module here is used for sim to sim communication .GSM module can send messages received from arduino to any specified mobile number. And can also receive messages from any number and forward those messages to arduino. Thus it acts as a medium to initiate communication between mobile and arduino. In short, GSM actually acts as a mobile. It can also be used to dial calls and receive incoming calls.

We use SIM900 GSM Module – This means the module supports communication in 900MHz band. We are from India and most of the mobile network providers in this country operate in the 900Mhz band. If you are from another country, you have to check the mobile network band in your area. A majority of **United States** mobile networks operate in 850Mhz band (the band is either 850Mhz or 1900Mhz). **Canada** operates primarily on 1900 Mhz band.

Check the power requirements of GSM module – GSM modules are manufactured by different companies. They all have different input power supply specs. You need to double check your GSM modules power requirements. In this tutorial, our gsm module requires a 12 volts input. So we feed it using a 12V,1A DC power supply.

START the GSM Module:-

1. Insert the SIM card to GSM module and lock it.
2. Connect the adapter to GSM module and turn it ON!

3. Now wait for some time (say 1 minute) and see the blinking rate of 'status LED' or 'network LED' (GSM module will take some time to establish connection with mobile network)

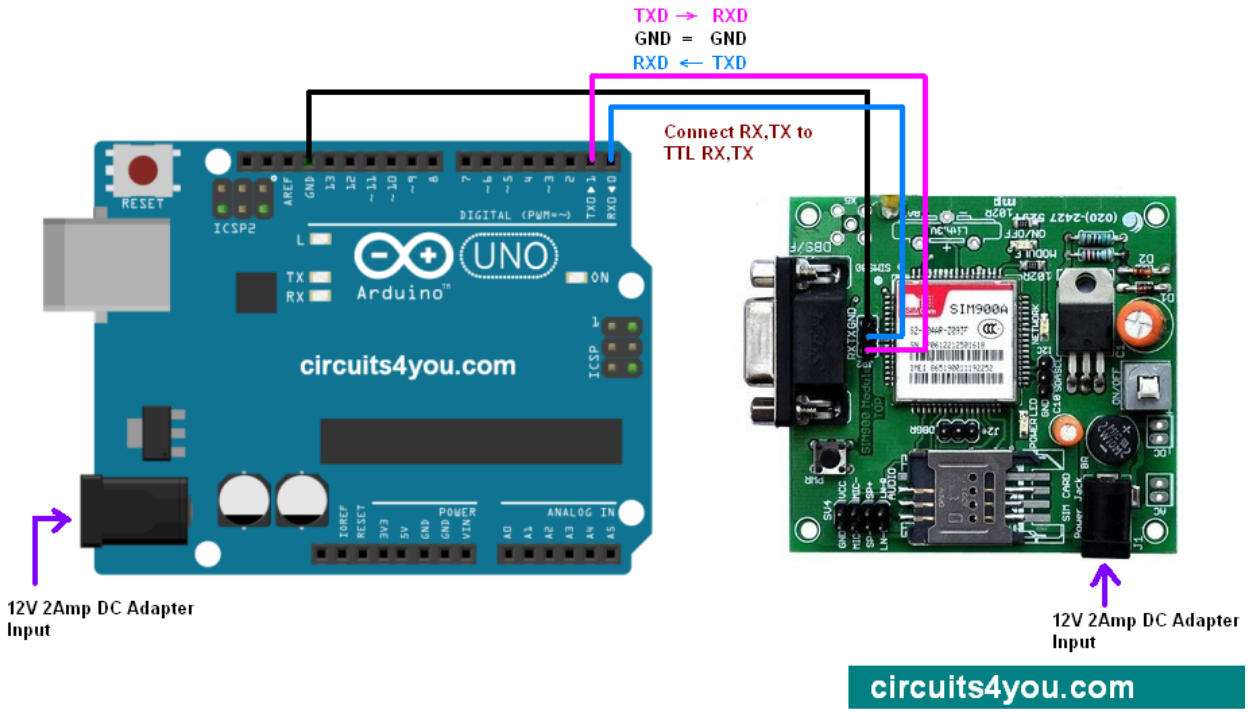
4. Once the connection is established successfully, the status/network LED will blink continuously every 3 seconds. You may try making a call to the mobile number of the sim card inside GSM module. If you hear a ring back, the gsm module has successfully established network connection.

Connecting GSM Module to Arduino

For connecting GSM module to arduino.The communication between Arduino and GSM module is serial. So we are supposed to use serial pins of Arduino (Rx and Tx). So if you are going with this method, you may connect the Tx pin of GSM module to Rx pin of Arduino and Rx pin of GSM module to Tx pin of Arduino. You read it right ? **GSM Tx –>Arduino Rx** and **GSM Rx –>ArduinoTx**. Now connect the ground pin of arduino to ground pin of gsm module

Note:- The problem with this connection is that, while programming Arduino uses serial ports to load program from the Arduino IDE. If these pins are used in wiring, the program will not be loaded successfully to Arduino. So you have to disconnect wiring in Rx and Tx each time you burn the program to arduino. Once the program is loaded successfully, you can reconnect these pins and have the system working!

HARDWARE CONNECTIONS



PRACTICALS PERFORMED

1. Dialing a number using gsm.
2. Attend an incoming call on gsm
3. Send a message to a particular number example:alert messages with help of gsm
4. Recive a message from a number with help of gsm and forward it to arduino.

CODES FOR GSM

GSM mainly works on AT commands. With the help of at commands we can put gsm in call mode or

message mode. In case we are tosend or receive messages the gsm has to be in message mode. And in order to receive or dial calls gsm has to be in call mode.

SOME BASIC AT COMANDS FOR GSM:-

AT
AT+CMGF=1 /// to put gsm in message mode
AT+CMGF=0/// to put gsm in call mode
ATDXXXXXXXXXX; ////to dial a call using gsm.
Xxxxxxxxxxx here is a 10 digit number to be dialed
ATA ////to attend a received call
ATH ///// to halt a received call
AT+CMGS=”+91xxxxxxxxxx” /////command to send a message on particular number

CODE TO DIAL A NUMBER:-

```
void setup()
{
  Serial.begin(9600);
  Serial.println("AT");      ///attention command
  delay(1000);
   Serial.println("AT+CMGF=0");   ///command to put gsm in call mode
  delay(1000);
  Serial.println("ATD8196013364;");    /////number to be dialled is given using this command
  }
  void loop()
  {
   ;
    }
```

CODE TO ATTEND A CALL

```
void setup()
{
  Serial.begin(9600);
  Serial.println("AT");      ///attention command
  delay(1000);
   Serial.println("AT+CMGF=0");   ///command to put gsm in call mode
  delay(1000);

  }
  void loop()
  {
   Serial.println("ATA");   ///command attend an incoming call
  delay(5000);
    }
```

CODE TO SEND A MESSAGE

```
void setup()
{
  Serial.begin(9600);
  Serial.println("AT");
    delay(1200);
   Serial.println("AT+CMGF=1"); // set gsm in message mode
    delay(1200);
   Serial.println();
    Serial.print("AT+CMGS=\""); // to set number on which mesae is to be sent
    Serial.print("+918198004484");
    Serial.println("\"");
    delay(1000);
    Serial.println("HELLO");    ////////message to be sent it can be anything like alert message

     delay(500);
     //delay(500);

    Serial.write(0x1A);
    Serial.write(0x0D);
    Serial.write(0x0A);    //////////three commands ox1a,0x0d and 0x0a are for message
                              ////forwarding from gsm to above mentioned mobile number
    delay(2000);
  }
  void loop()
  {
  ;
    }
```

WIFI MODULE(ESP8266) INTERFACING WITH ARDUINO

The ESP8266 Wifi module is a complete WiFi network where you can easily connect as a serving Wi- Fi adapter, wireless internet access interface to any microcontroller based design on its simple connectivity through Serial Communication or UART interface.

Communication with ESP8266 is via Attention Command or AT Commands.

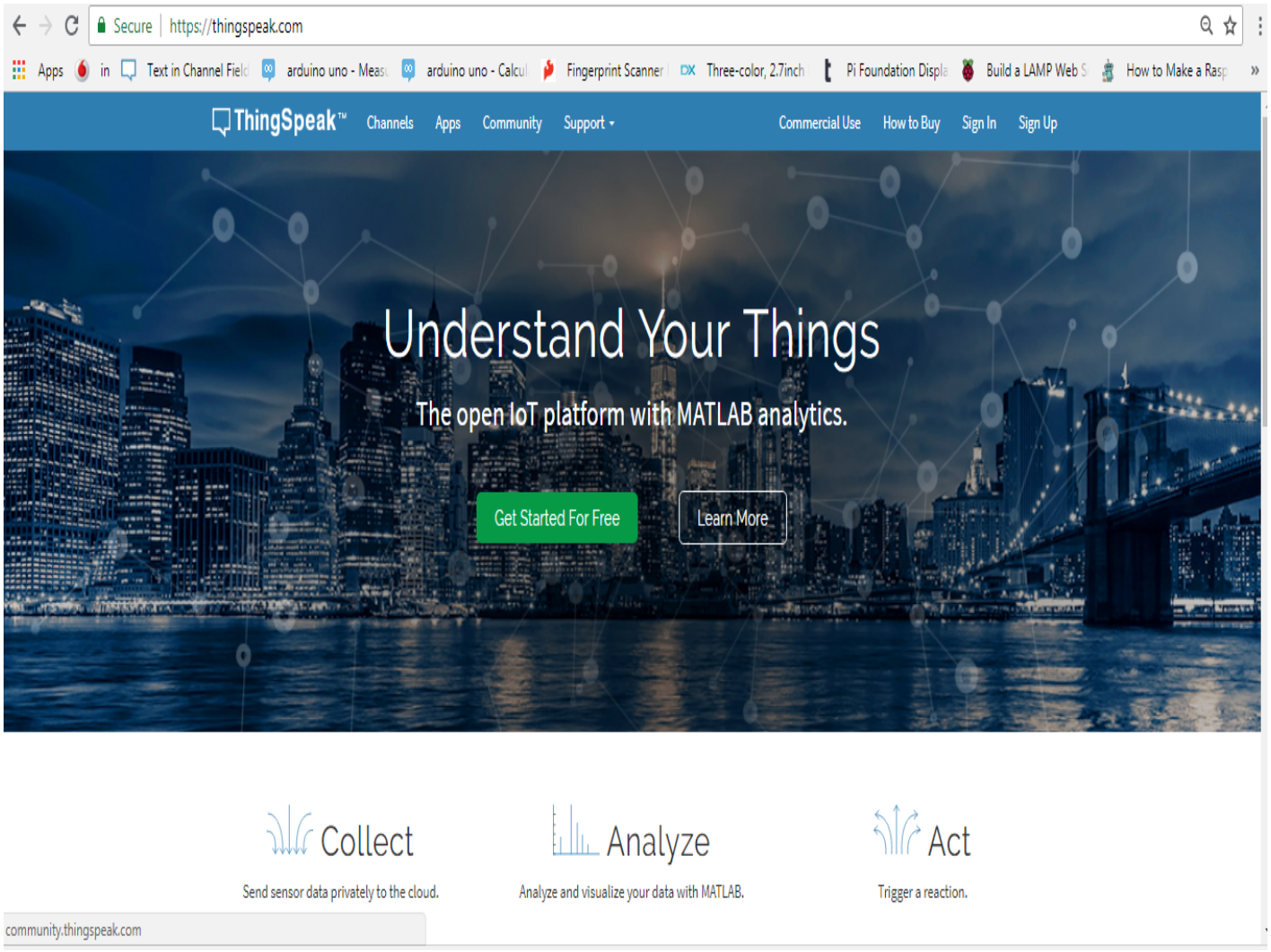
Commands	Description	Type
AT+RST	restart module	basic
AT+CWMODE	wifi mode	wifi
AT+CWJAP	join AP	wifi
AT+CWLAP	list AP	wif
AT+CWQAP	quit AP	wifi
AT+CIPSTATUS	get status	TCP/IP
AT+CIPSTART	set up TCP or UDP	TCP/IP
AT+CIPSEND	send data	TCP/IP
AT+CIPCLOSE	close TCP or UDP	TCP/IP
AT+CIFSR	get IP	TCP/IP
AT+CIPMUX	set multiple connections	TCP/IP
AT+CIPSERVER	set as server	TCP/IP

PRACTICALS DONE

1.Uploading data on THINGSPEAK IOT PLATFORM(CLOUD) with the help of esp8266.

For performing this practical we need to sign up on thingspeak

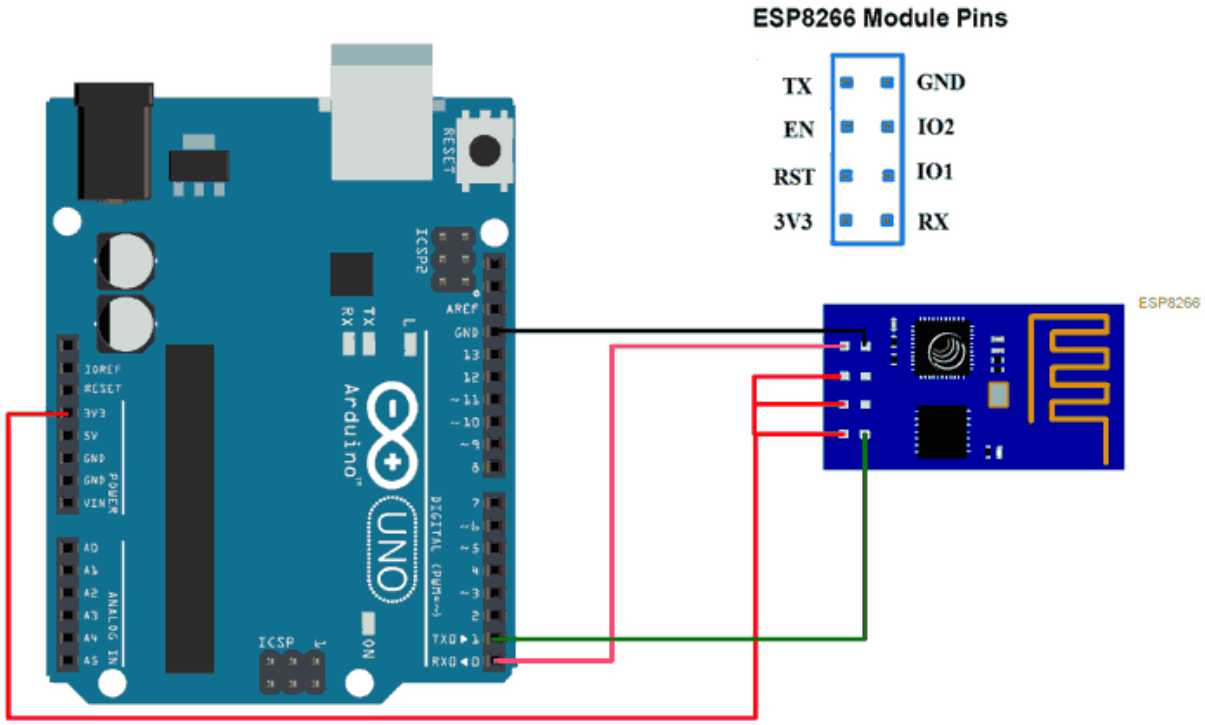
Go to link :-thingspeak.com



After signing up on thingspeak we will get CHANNEL ID,READ API KEY,WRITE API KEY AND API REQUESTS . With help of above parameters we can write

data on thingspeak cloud. These parameters are to provided in the software .

HARDWARE CONNECTIONS OF ESP8266



CODE TO UPLOAD DATA TO THINSPEAK CLOUD

```
#include <stdlib.h>

#define SSID "ZAKBEE"     // "SSID-WiFiname" (on this very network esp module will connect itself)
#define PASS "aaiza123"       // "password"
#define IP "184.106.153.149"// thingspeak.com ip
String msg = "GET /update?key=3EAEVSC69ST019IZ"; //change it with your key...
/*-------------------------------------------------------------*/

//Variables
float temp;

void setup()
{
  Serial.begin(115200); //or use default 115200.

  Serial.println("AT");
  delay(5000);
  lcd.clear();
  if(Serial.find("OK")){
     delay(2000);

     delay(5000);
    connectWiFi();
    delay(5000);
  }
}
```

```
void loop(){

  start: //label
 temp++;
 delay(1000);


  updateTemp();
  //Resend if transmission is not completed
  if (error==1){
    goto start; //go to label "start"
  }

  delay(100); //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 += temp;

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;
  }
}
```