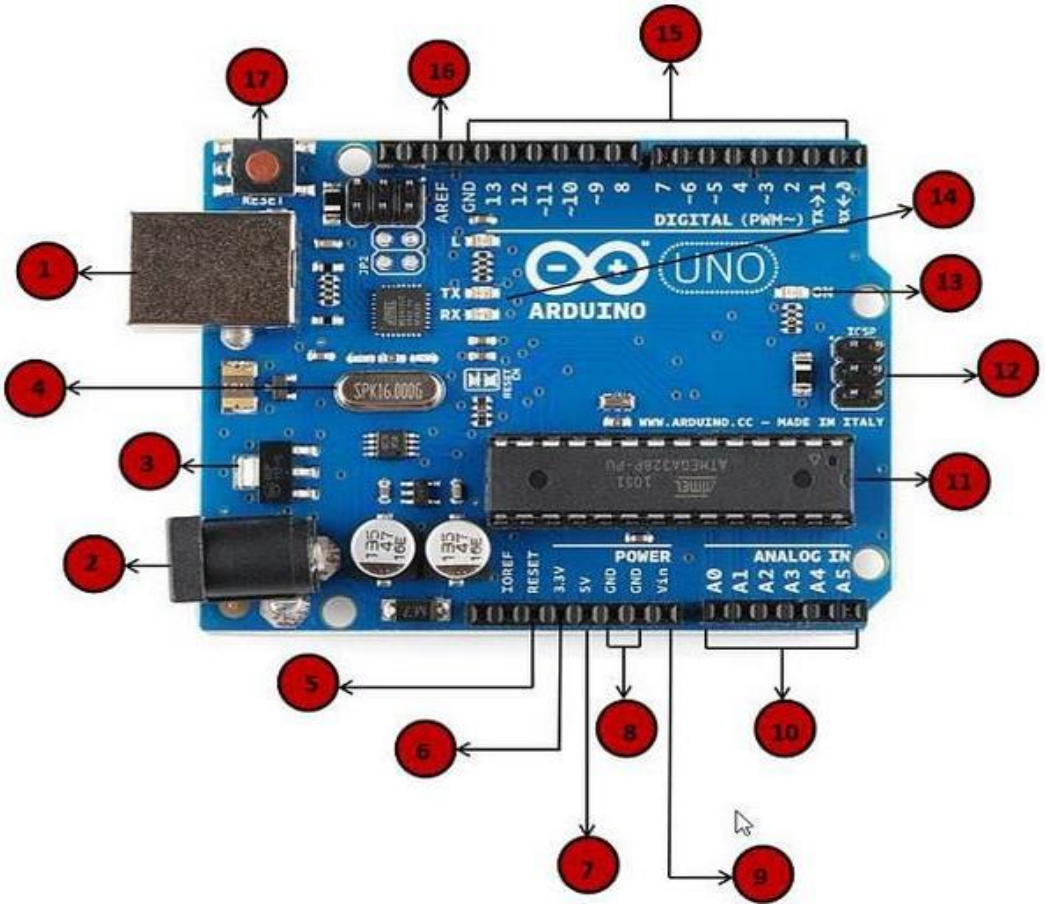
Understanding Arduino UNO Board and Components

Demonstrating Arduino UNO Board and its components.

Arduino UNO Board

Arduino UNO is an Micro controller which is the most popular board in the Arduino board family. It is the best board to get started with electronics and coding.

The components of the Arduino UNO are



Power USB

Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).

Power (Barrel Jack)

Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

Voltage Regulator

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

Crystal Oscillator

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

Arduino Reset

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

Pins (3.3, 5, GND, Vin)

- 3.3V (6) − Supply 3.3 output volt
- 5V (7) − Supply 5 output volt
- Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.
- GND (8)(Ground) − There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- Vin (9) − This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

Analog pins

The Arduino UNO board has 6- analog input pins A0 through A5. These pins can read the signal from an analog sensor like the **humidity sensor or temperature sensor** and convert it into a **digital value** that can be read by the microprocessor.

Main microcontroller(IC-ATmega328P)

Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

ICSP pin

Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

Power LED indicator

This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

TX and RX LEDs

On your board, you will find **two** labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

Digital I/O

The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.

AREF

AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Installing and work with Arduino IDE

Demonstrating the installing of arduino IDE in laptops or Desktops.

REQUIREMENTS:

- Arduino UNO Board,
- USB A to USB-B Cable.
- Arduino Ide

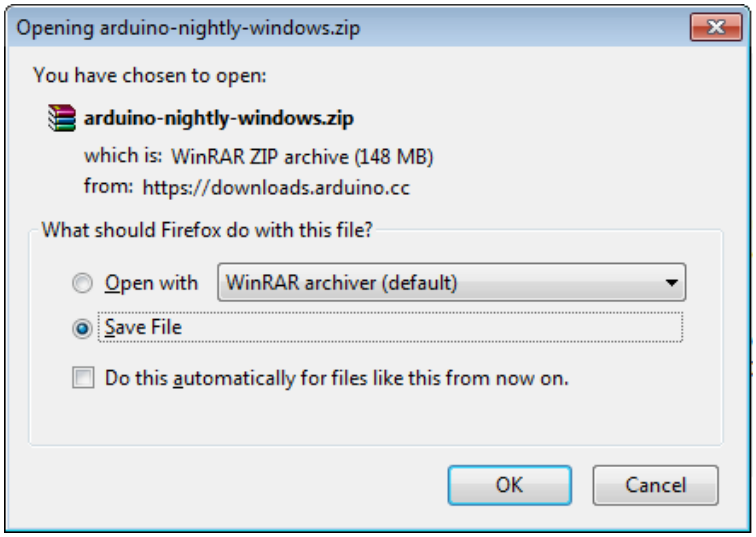
DESCRIPTION:

We will learn how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1 − First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, ArduinoDuemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

Step 2 – Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



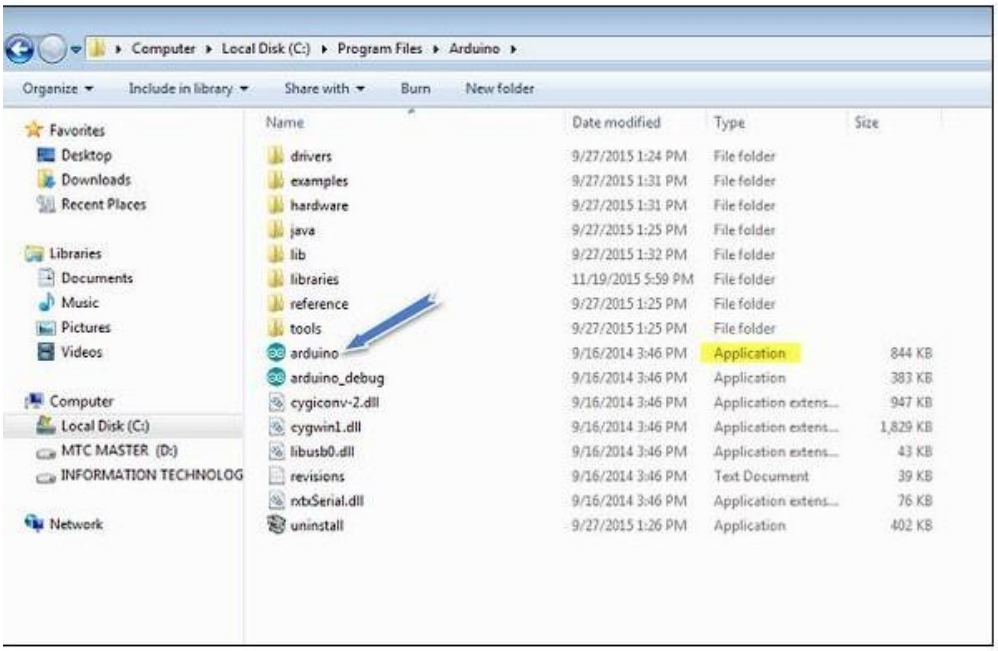
Step 3 − Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an ArduinoDiecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 − Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



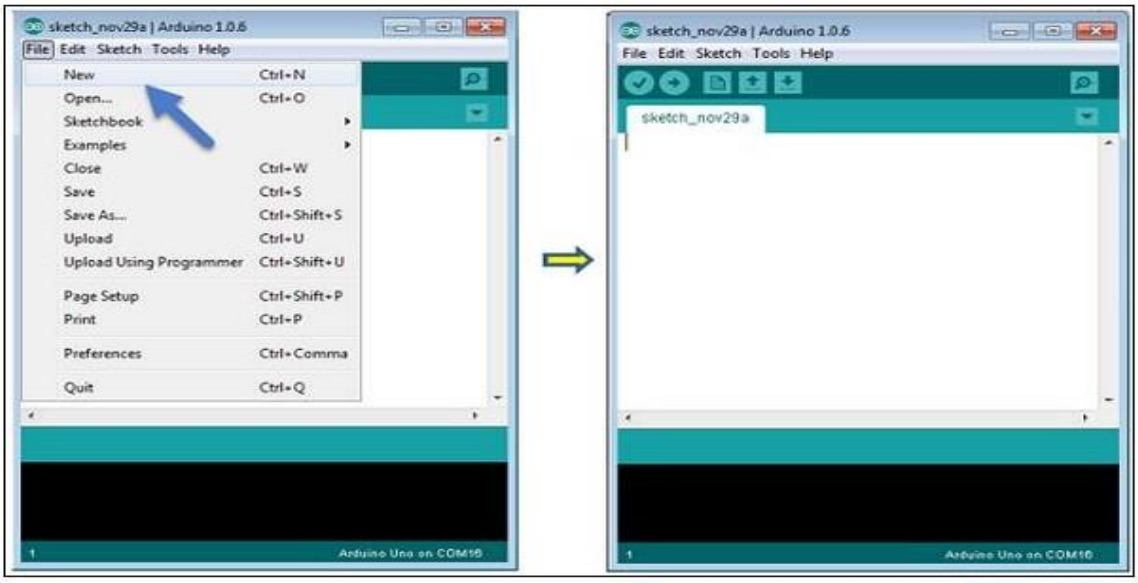
Step 5 − Open your first project.

Once the software starts, you have two options −

Create a new project.

Open an existing project example.

To create a new project, select File → New.



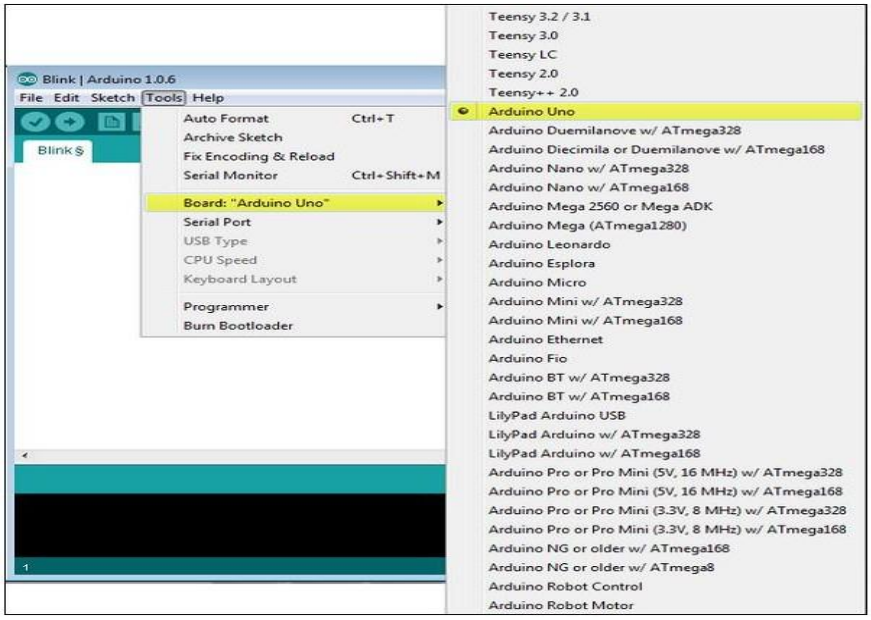
To open an existing project example, select File → Example → Basics → Blink.

Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6 − Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

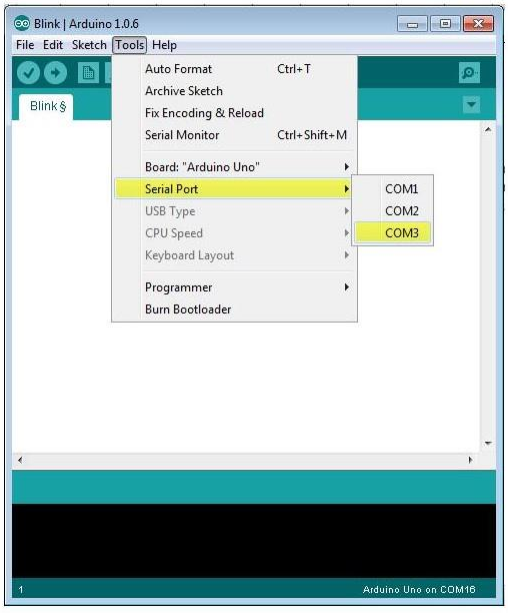
Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

Step 7 − Select your serial port.

Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 − Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.

- **A** − Used to check if there is any compilation error.
- **B** − Used to upload a program to the Arduino board.
- **C** − Shortcut used to create a new sketch.
- **D** − Used to directly open one of the example sketch.
- **E** − Used to save your sketch.
- **F** − Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Blinking LED sketch with Arduino

Demonstrating the Blinking LED with Arduino.

HARDWARE

- Arduino UNO Board,
- USB A to USB-B Cable.
- 5mm LED
- Jumper wires
- Breadboard

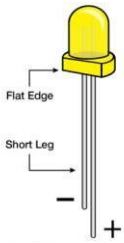
SOFTWARE

Arduino IDE

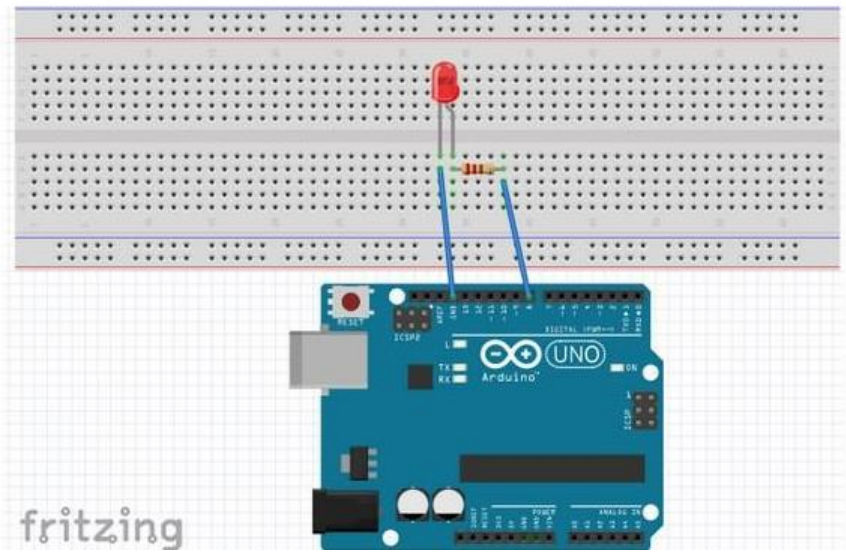
PROCEDURE:

Follow the circuit diagram and hook up the components on the breadboard as shown in the image given below.

Note – To find out the polarity of an LED, look at it closely. The shorter of the two legs, towards the flat edge of the bulb indicates the negative terminal.



CIRCUIT DIAGRAM



SOURCE CODE

```
constintledPin=8; //definition digital 8 pins as pin to control the LED

void setup()
{
pinMode(ledPin,OUTPUT);     //Set the digital 8 port mode, OUTPUT: Output mode
}

void loop()
{
digitalWrite(ledPin,HIGH); //HIGH is set to about 5V PIN8
delay(1000);               //Set the delay time, 1000ms = 1S
digitalWrite(ledPin,LOW); //LOW is set to about 5V PIN8
delay(1000);               //Set the delay time, 1000ms = 1S
}
```

Simulation of 4-Way Traffic Light with Arduino

Demonsrating the simulation of traffic lights with Arduino

HARDWARE

- Arduino UNO Board,
- USB A to USB-B Cable.
- 5mm LED -6 (Red-2, Green-2, Yellow-2)
- Jumper wires
- Breadboard

SOFTWARE

Arduino IDE

PROCEDURE:

This project is done to give you an idea of how the traffic light controller works. This is not the real time traffic light controller.

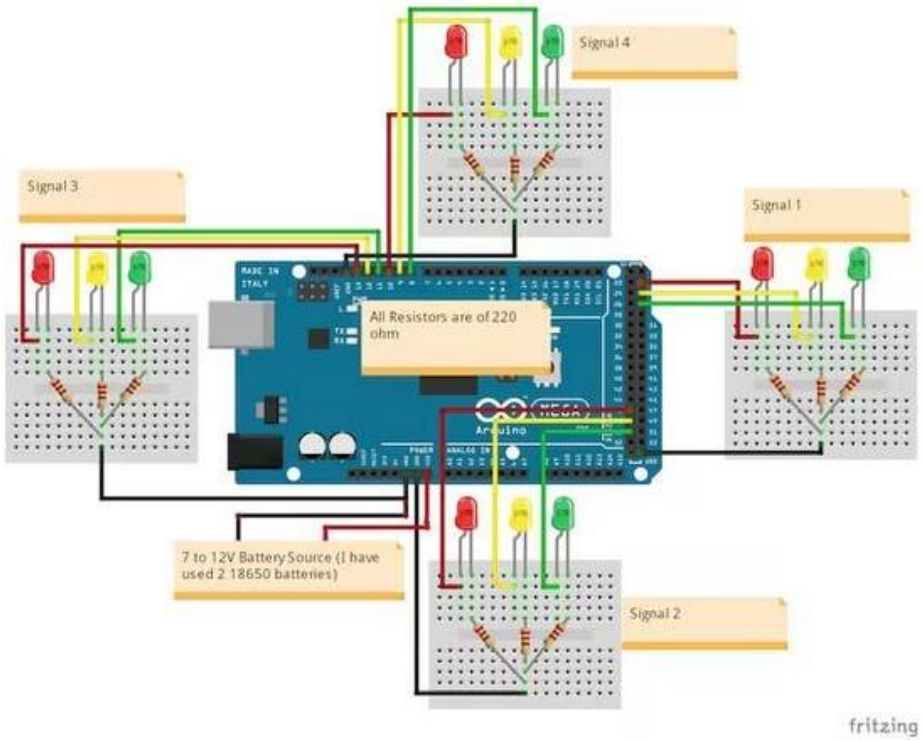
So at start, green light of signal 1 and red lights at other signals will light up to give time to the vehicles at signal 1 to pass.

After 5 seconds, the yellow light at signal 1 will light up to give an indication that the red light at signal 1 is about to come up and also to give an indication to the vehicles at signal 2 that the green light is about to light up.

So after 2 seconds, red light at signal 1 will come up and green light at signal will come up meaning vehicles at signal 1 must stop and vehicles at signal 2 can move.

Similarly the traffic light controller will work for the signal 3, signal 4 and the system will keep looping.

CIRCUIT DIAGRAM



SOURCE CODE

```
int  rled1=2;
int  yled1=3;
int gled1=4;

int    rled2=8;
int    yled2=9;
int gled2=10;

void setup()
{
            Serial.begin(9600);
            pinMode(rled1,OUTPUT);
            pinMode(yled1,OUTPUT);
            pinMode(gled1,OUTPUT);

            pinMode(rled2,OUTPUT);
            pinMode(yled2,OUTPUT);
            pinMode(gled2,OUTPUT);

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

            digitalWrite(rled1,HIGH); //1-way red light high
            digitalWrite(gled2,HIGH);//2-way green light high
            Serial.println("giving second path to move vehicles");
            delay(5000);
```

```
        digitalWrite(yled2,HIGH);
        digitalWrite(gled2,LOW);
        Serial.println("signal yellow is on process of way-2");
        delay(2000);

        digitalWrite(rled1,LOW);
        digitalWrite(rled2,HIGH);
        digitalWrite(gled1,HIGH);
        digitalWrite(yled2,LOW);
        Serial.println("way-2 is stopped and way-1 is start");
        delay(5000);

        digitalWrite(gled1,LOW);
        digitalWrite(yled1,HIGH);
        digitalWrite(rled2,HIGH);
        Serial.println("way-1 yellow signal is on stop the vehciles");
        delay(2000);

        digitalWrite(rled1,LOW);
        digitalWrite(yled1,LOW);
        digitalWrite(gled1,LOW);
        digitalWrite(rled2,LOW);
        digitalWrite(yled2,LOW);
        digitalWrite(gled2,LOW);
        Serial.println("System is resseting");
}
```

Using Pulse Width Modulation in Arduino

Fading of led using PWM (Pulse with Modulation) Pin with arduino for increasing and decreasing intensity of led.

HARDWARE

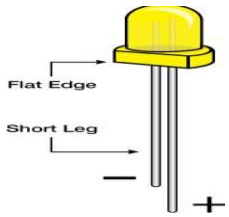
- Arduino UNO Board,
- USB A to USB-B Cable.
- 5mm LED -1
- Jumper wires
- Breadboard
- 330Ω resistor

SOFTWARE

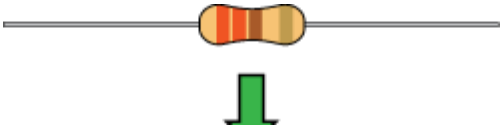
Arduino IDE

PROCEDURE:

find out the polarity of an LED, look at it closely. The shorter of the two legs, towards the flat edge of the bulb indicates the negative terminal.



Components like resistors need to have their terminals bent into 90° angles in order to fit the breadboard sockets properly. You can also cut the terminals shorter.

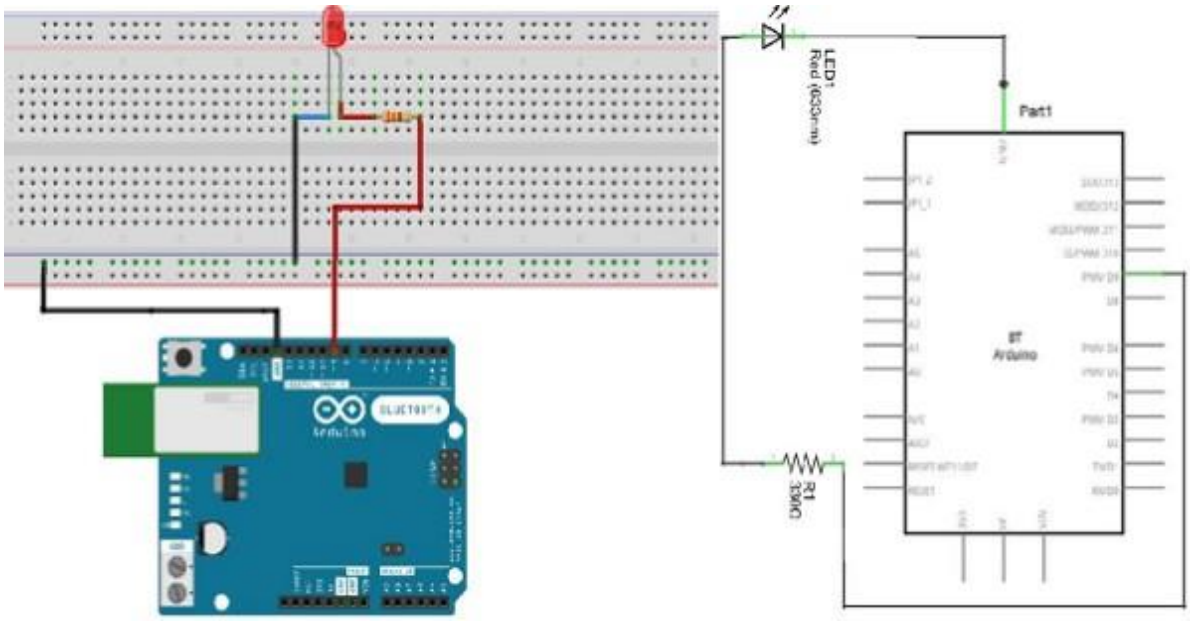


In order to fade the LED off and on, gradually increase the PWM values from 0 (all the way off) to 255 (all the way on), and then back to 0, to complete the cycle. In the sketch given above, the PWM value is set using a variable called brightness. Each time through the loop, it increases by the value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

If brightness is at either extreme of its value (either 0 or 255), then fadeAmount is changed to its negative. In other words, if fadeAmount is 5, then it is set to -5. If it is -5, then it is set to 5. The next time through the loop, this change causes brightness to change direction as well.

analogWrite() can change the PWM value very fast, so the delay at the end of the sketch controls the speed of the fade. Try changing the value of the delay and see how it changes the fading effect.

CIRCUIT DIAGRAM



SOURCE CODE

```
int led = 9; // the PWM pin the LED is attached to
int brightness = 0; // how bright the LED is
intfadeAmount = 5; // how many points to fade the LED by
// the setup routine runs once when you press reset:

void setup() {
  // declare pin 9 to be an output:
pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:

void loop()
 {
  // set the brightness of pin 9:
analogWrite(led, brightness);
  // change the brightness for next time through the loop:
brightness = brightness + fadeAmount;
  // 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(300);
}
```

LED Fade Sketch and Button Sketch

Fading of led when button pressed to maximum intensity and decreasing intensity of led

HARDWARE

- Arduino UNO Board,
- USB A to USB-B Cable.
- 5mm LED
- Jumper wires
- Breadboard
- Push button switch
- 330Ω resistor

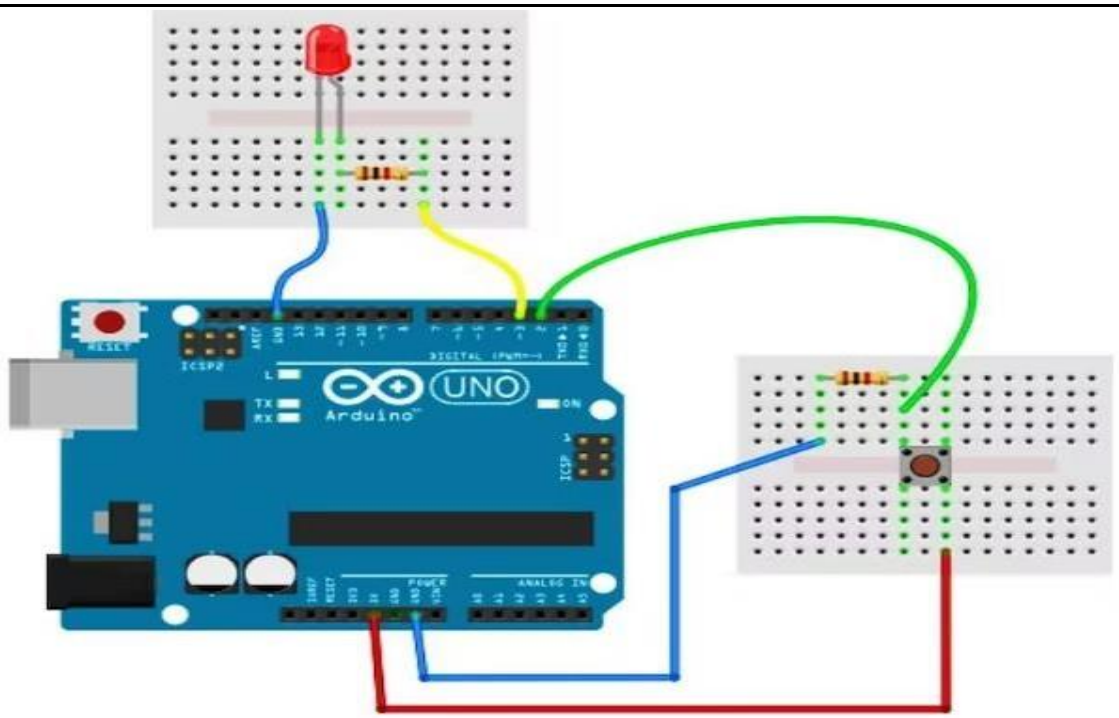
SOFTWARE

Arduino IDE

PROCEDURE:

- First, make sure to power off your Arduino – remove any USB cable.
- Plug a black wire between the blue line of the breadboard and a ground (GND) pin on the Arduino board.
- Plug the LED. You can notice that the LED has a leg shorter than the other. Plug this shorter leg to the ground (blue line here) of the circuit.
- Connect the longer leg of the LED to a digital pin (here pin no 8, you can change it). Add a 220 Ohm resistor in between to limit the current going through the LED.
- Add the push button to the breadboard, like in the picture.
- Connect one leg of the button to the ground, and put a 10k Ohm resistor in between. This resistor will act as a "pull down" resistor, which means that the default button's state will be LOW.
- Add a red wire between another leg of the button and VCC (5V).
- Finally, connect a leg of the button (same side as the pull down resistor) to a digital pin (here 7).

CIRCUIT DIAGRAM



SOURCE CODE

```
constint BUTTON = 2;
constint LED = 3;
intBUTTONstate = 0;

void setup()
{
pinMode(BUTTON, INPUT);
pinMode(LED, OUTPUT);
}

void loop()
{
BUTTONstate = digitalRead(BUTTON);
if (BUTTONstate == HIGH)
  {
digitalWrite(LED, HIGH);
  }
else
{
digitalWrite(LED, LOW);
  }
}
```

Analog Input Sketch (Bar Graph with LEDs and Potentiometre)

Using potentiometer Bar Graph representing with LED's using arduino.

HARDWARE

- Arduino UNO Board,
- USB A to USB-B Cable.
- 5mm LED's -10
- 10 220 ohm resistors
- Jumper wires
- Breadboard

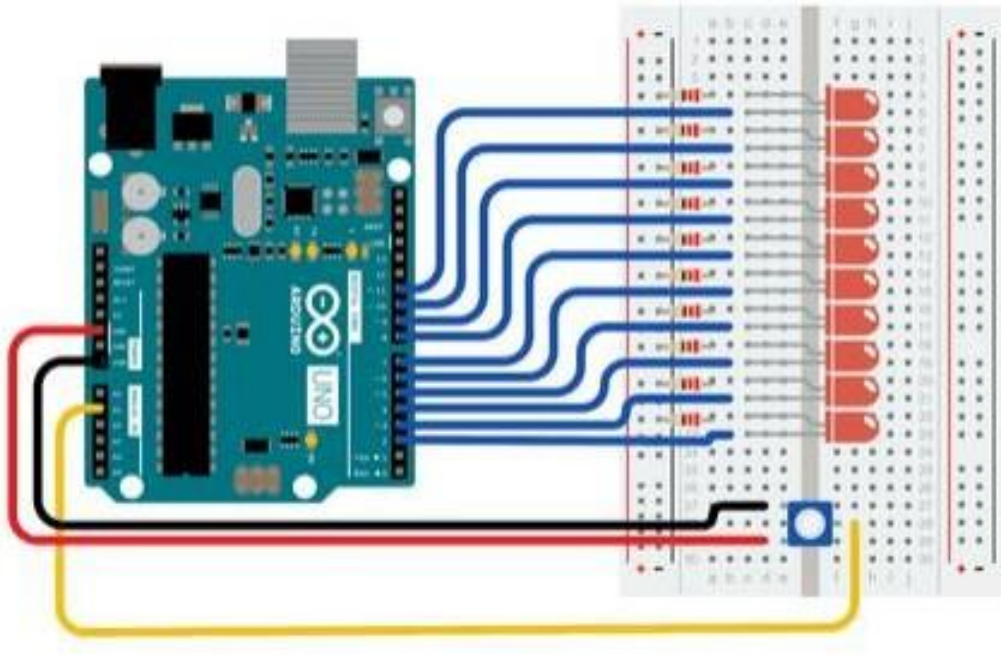
SOFTWARE

Arduino IDE

PROCEDURE:

The bar graph - a series of LEDs in a line, such as you see on an audio display - is a common hardware display for analog sensors. It's made up of a series of LEDs in a row, an analog input like a potentiometer, and a little code in between

CIRCUIT DIAGRAM



SOURCE CODE

```
intptm=A0,count=100;
int  led[]={2,3,4,5,6,7,8,9,10,11};
void setup()
{
  // put your main code here, to run repeatedly:
for(int i=0;i<count;i++)
{
pinMode(led[i],OUTPUT);
}
pinMode(ptm,INPUT);
}
void loop()
{
intptmvalue=analogRead(ptm);
intledLevel = map(ptmvalue, 0, 1023, 0, count);

  // loop over the LED array:
for (int i = 0; i < count; i++)
  {
    // if the array element's index is less than ledLevel,
    // turn the pin for this element on:
if (i <ledLevel)
        {
digitalWrite(led[i],  HIGH);
        }
    // turn off all pins higher than the ledLevel:
else
    {
```

```
digitalWrite(led[i],  LOW);
    }
  }
}
```

Digital Read Serial Sketch (Working with DHT/IR/Gas or Any other Sensor)

Interfacing MQ5 Gas Sensor WithArduino and knowing whether any leakage of gas is existed or not.

HARDWARE

- Arduino UNO Board,
- USB A to USB-B Cable.
- MQ-5 Gas Sensor
- Jumper wires
- Breadboard

SOFTWARE

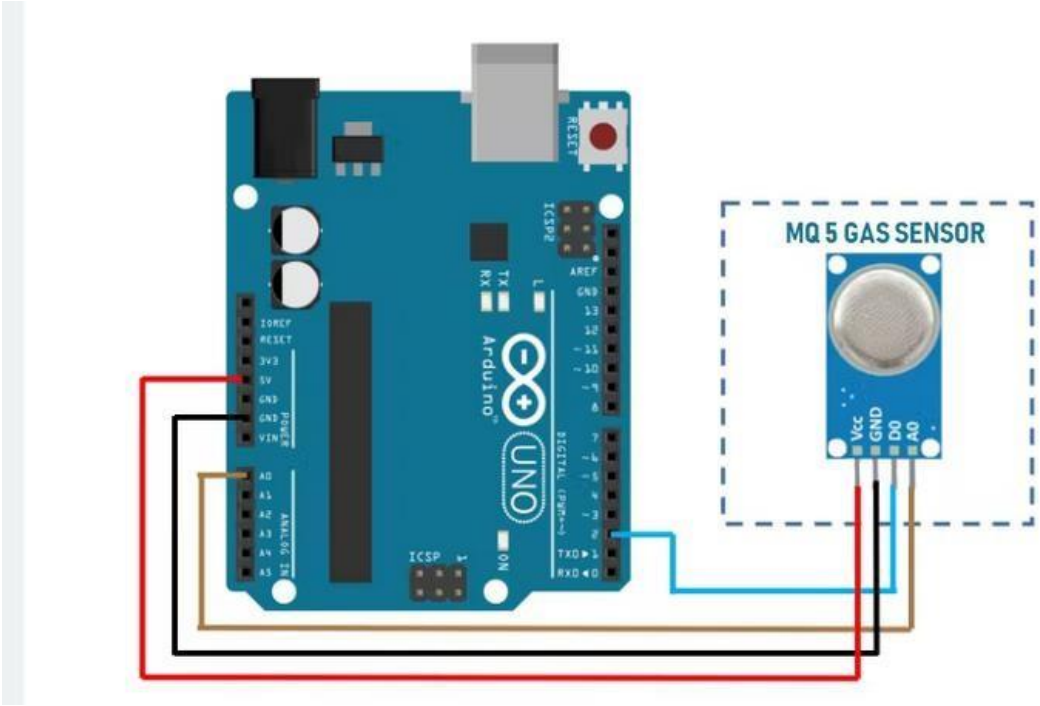
Arduino IDE

PROCEDURE:

The most basic connection for an MQ5 is between a digital pin on theArduinoUnoand its power supply. You'll need to connect a +5V line from the Arduino to pin 4 (the "VCC" connector) for power. And another GND line from the Arduino to pin 6 (the "GND" connector) for grounding. I have used both analogue and Digital Pins You .can use either digital or analogue

MQ5 Arduino sensors can be used in different situations even though it may be hard to believe. For instance, to test if there is a gas leak when starting a car, you could use an MQ5 sensor. With the help of a cigarette lighter, pressure should rise as one presses the trigger and provides the needed gas leakage.

CIRCUIT DIAGRAM



SOURCE CODE

```
long mq6_sensor=A0;
intled_indicator=13;
intthreshold_leak=100;
void setup()
{
  // put your setup code here, to run once:
pinMode(mq6_sensor,INPUT);
pinMode(led_indicator,OUTPUT);
Serial.begin(9600);
}

void loop()
{
  // put your main code here, to run repeatedly:
intsensor_data=analogRead(mq6_sensor);
Serial.print("the leakage of gas value is");
Serial.println(value);

if(sensor_data>threshold_leak)
 {
Serial.println("huge amount of leakge exist");
digitalWrite(led_indicator,HIGH);
 }
else
 {

digitalWrite(led,LOW);
 }
delay(2000);
}
```

Working with Adafruit Libraries in Arduino

working with Adafruitlibraires with arduino and installing libraries for OLED display for Adafuit.

HARDWARE

- Arduino UNO Board,
- USB A to USB-B Cable.
- 5mm LED -6 (Red-2, Green-2, Yellow-2)
- Jumper wires
- Breadboard

SOFTWARE

Arduino IDE

PROCEDURE:

The Arduino IDE comes with a set of standard libraries for commonly used functionality. These libraries support all the examples included with the IDE. Standard library functionality includes basic communication functions and support for some of the most common types of hardware like servo motors and character LCD displays.

Standard Libraries are pre-installed in the "Libraries" folder of the Arduino install. If you have multiple versions of the IDE installed, each version will have its own set of libraries. For the most part, it is not a good idea to change the Standard Libraries or install your libraries in the same folder.

AdaFruit

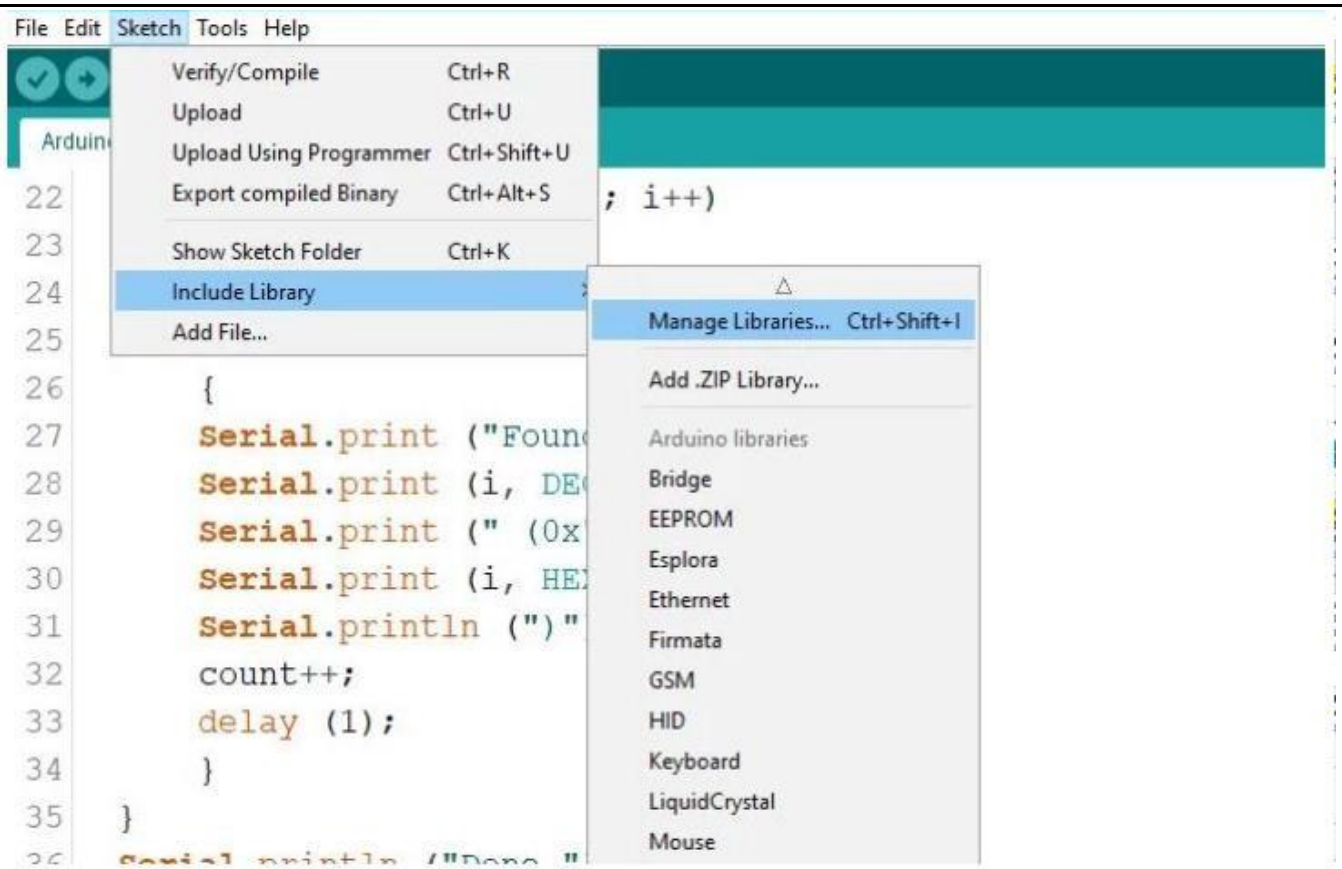
The Arduino IDE has a Library Manager which facilitates installing third-party libraries submitted to Arduino for use. Adafruit has most of its libraries and drivers in the Library Manager for easy use.

INSTALLATION OF ADAFRUIT LIBRARIES FOR INTEGRATING OLED DISPLAY WITH ARDUINO

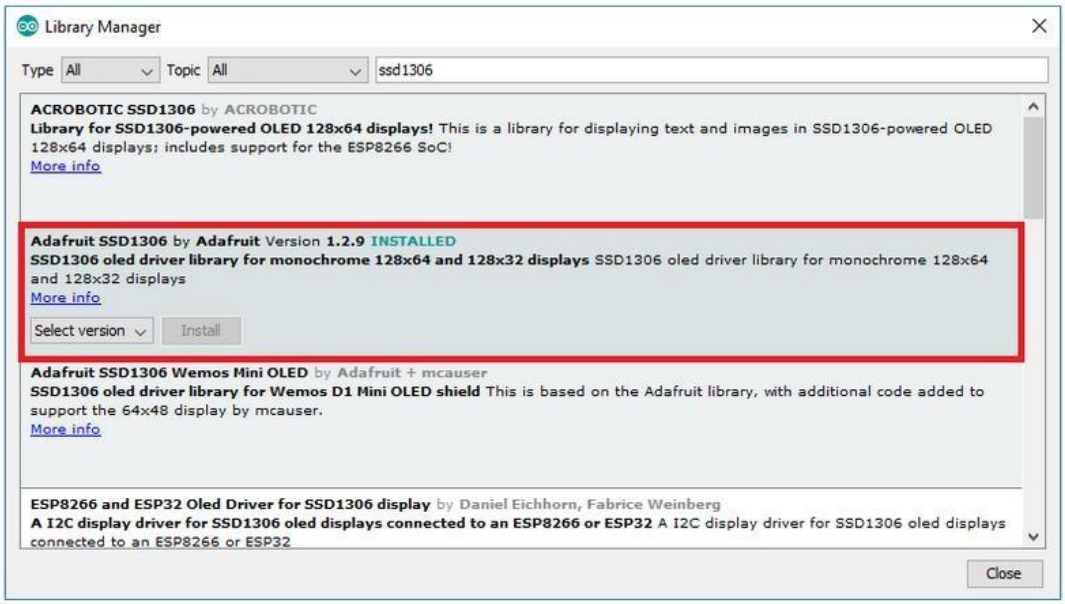
you can include these libraries by following the steps.

1. go to **Sketch** menu.
2. select**Include Libraries**.
3. go to **Manage Libraries**.
4. search for **ADAFRUIT GFX** and **INSTALL** it.
5. search for **ADAFRUIT SSD1306** and **INSTALL** it.

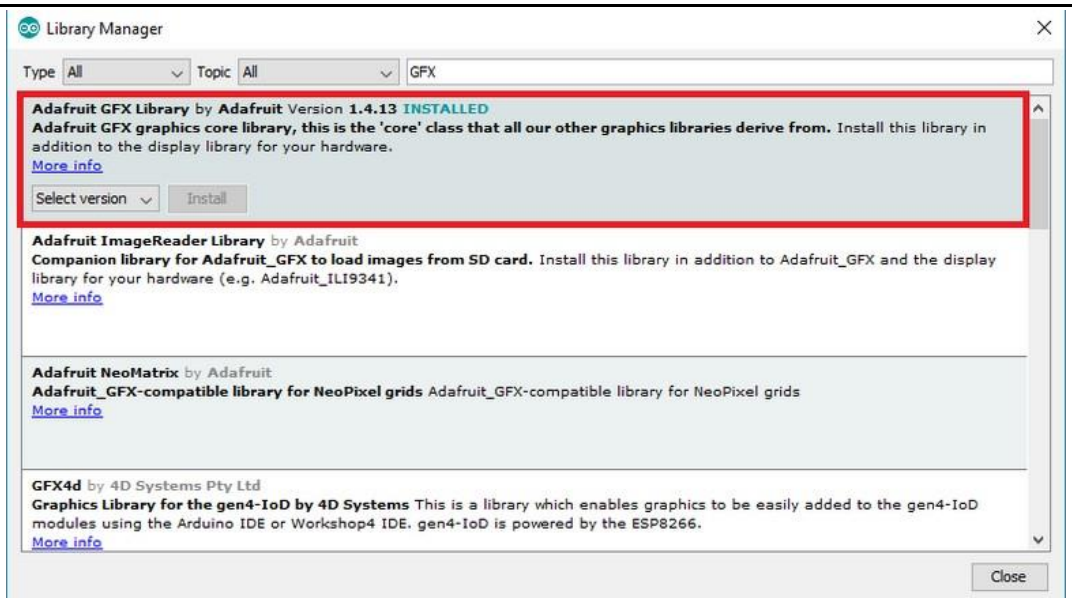
Going to **Manage Libraries**



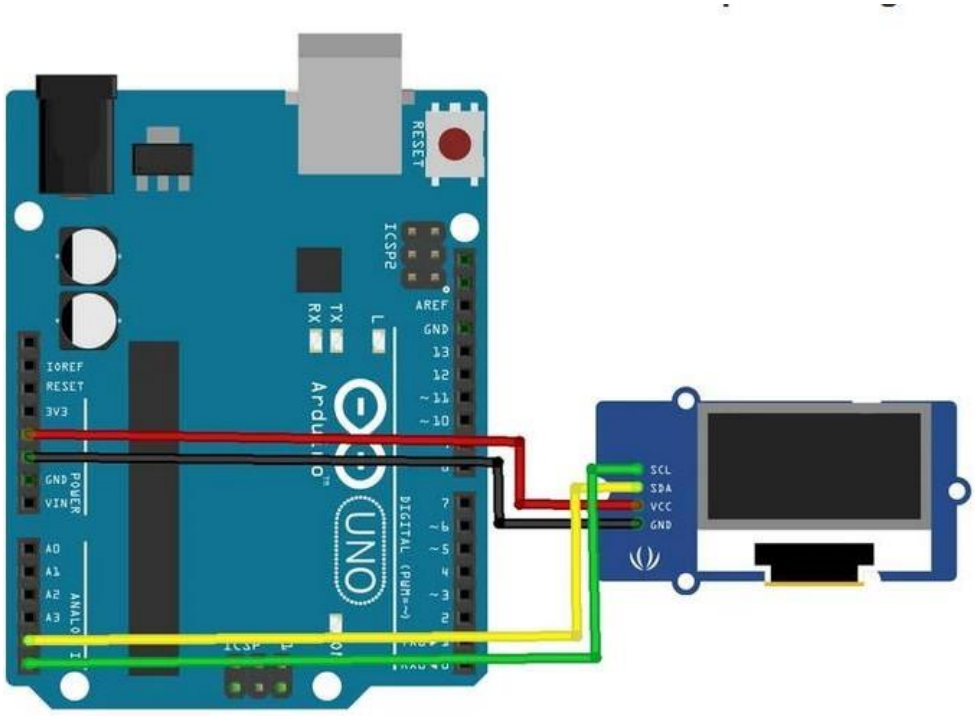
Search for **ADAFRUIT SSD1306** and **INSTALL** it.



Search for **ADAFRUIT GFX** and **INSTALL** it



CIRCUIT DIAGRAM



SOURCE CODE

```
#include <SPI.h>
#include  <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>

#define  OLED_WIDTH  128
#define  OLED_HEIGHT  64

#define OLED_ADDR  0x3C
#define OLED_RESET 4
Adafruit_SSD1306 oled(OLED_RESET);
intgas_leak=A0;
```

```
void setup()
{
oled.begin(SSD1306_SWITCHCAPVCC,  OLED_ADDR);
oled.clearDisplay();


Serial.begin(9600);
oled.display();
pinMode(gas_leak,INPUT);

}

void loop()
{
oled.clearDisplay();
oled.setTextSize(2);
oled.setTextColor(WHITE);

oled.setCursor(0,0);
oled.println("GAS-DETECT");
oled.setCursor(0,13);
oled.println(" ---------- ");
oled.setTextSize(1);
oled.setTextColor(WHITE);
oled.setCursor(0, 15);
int value=analogRead(gas_leak);
oled.setCursor(0, 30);
oled.print("Sensor value:");
oled.println(value);
oled.setTextSize(1);
oled.setTextColor(WHITE);
oled.setCursor(0, 45);
if(value > 75)
  {
oled.println("GAS LEAKAGE DETECTED");
oled.println("**** BE ALERT ****");
Serial.println("GAS  is  GETTING  LEAKED");
  }
else
  {
oled.println("NO LEKAGE DETECTED");
oled.println("**** BE-COOL ****");
Serial.println("GAS  IS  not  GETTING  LEAKED");
  }

oled.display();
delay(2000);
}
```

Spinning a DC Motor and Motor Speed Control Sketch

making spinning of dc motors with arduino and kotor speed controlling

HARDWARE

- Arduino UNO Board,
- USB A to USB-B Cable.
- L298 Motor Driver module
- DC Motor
- 9/12 V Battery
- Jumper wires
- Breadboard

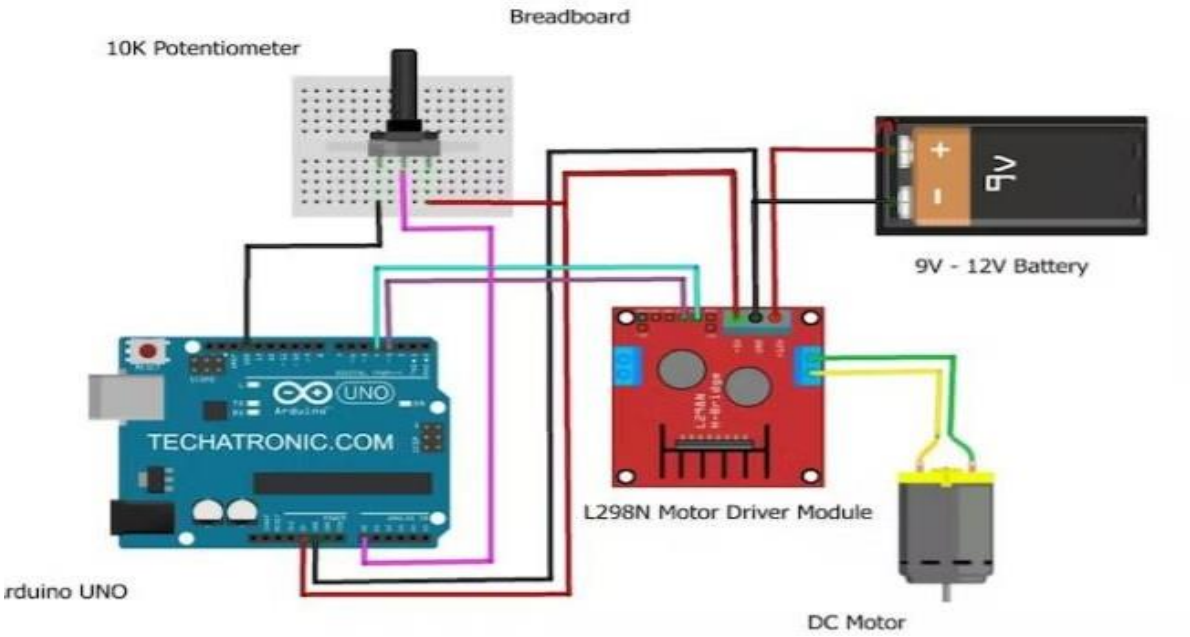
SOFTWARE

Arduino IDE

PROCEDURE:

Connect 5 volts pin of the Arduino with the 5 volts pin of the L298n motor driver module and one side pin of the 10K ohm potentiometer. Attach the GND pin of the Arduino with the GND pin of the L298n module and another side pin of the 10K potentiometer. Join the middle pin of the potentiometer with the analog-0 pin of the Arduino. Take a 9 volts battery and connect its positive wire with the 12 volts pin of the L298n module and its negative wire with the GND pin of the Arduino. Join the IN1 pin of the motor driver module with the digital-4 pin of the Arduino and the IN2 pin of the driver module with the digital-3 pin of the Arduino. At last, take a DC motor and connect its wires with the output pins of the driver module.

CIRCUIT DIAGRAM



SOURCE CODE

```
intmotorpinforward= 3;
intmotorpinback= 4;
void setup()
{
Serial.begin(9600);
pinMode(motorpinforward,OUTPUT); // Motor pin 1
pinMode(motorpinback,OUTPUT); // Motor pin 2
digitalWrite(motorpinback,LOW); // Normally LOW inthis pin
pinMode(A0,INPUT); //10k Potentiometer
}
void loop()
{
int s=analogRead(A0); //10k Potentiometer
int z=map(s,0,1024,0,255);
Serial.println(z);
analogWrite(motorpinforward,z);
}
```

Working various shields with Arduino

working with various shields witharduino

HARDWARE

- Arduino UNO Board,
- USB A to USB-B Cable.
- Jumper wires
- Ethernet shield or Any other shield device
- Breadboard

SOFTWARE

Arduino IDE

PROCEDURE:

Arduino shields are the boards, which are plugged over the Arduino board to expand its functionalities. There are different varieties of shields used for various tasks, such as Arduino motor shields, Arduino communication shields, etc.

Shield is defined as the hardware device that can be mounted over the board to increase the capabilities of the projects. It also makes our work easy. For example, Ethernet shields are used to connect the Arduino board to the Internet.

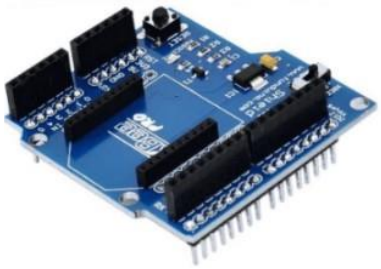
The pin position of the shields is similar to the Arduino boards. We can also connect the modules and sensors to the shields with the help of the connection cable.

popular Arduino shields are listed below:

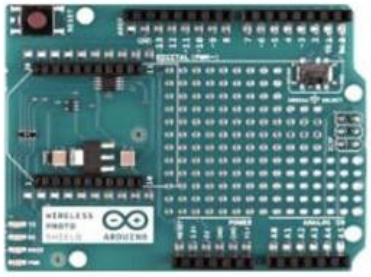
- **Ethernet shield**
- **Xbee Shield**
- **Proto shield**
- **Relay shield**
- **Motor shield**
- **LCD shield**
- **Bluetooth shield**
- **Capacitive Touchpad Shield**

Ethernet Shield

Bluetooth Shield



Xbee shield



Proto Shield

Relay Shield

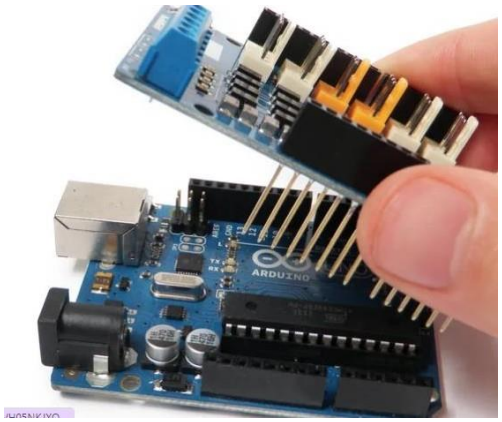
LCD Shield

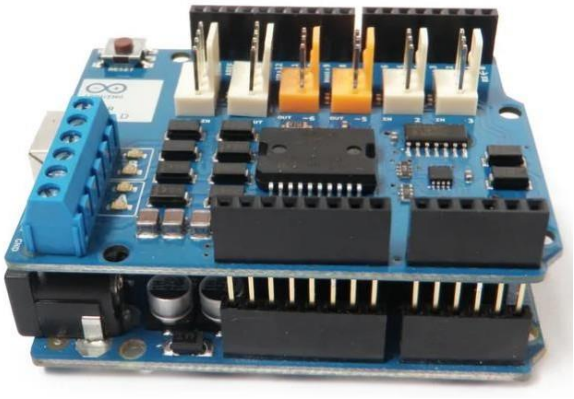
Capactive touch shield

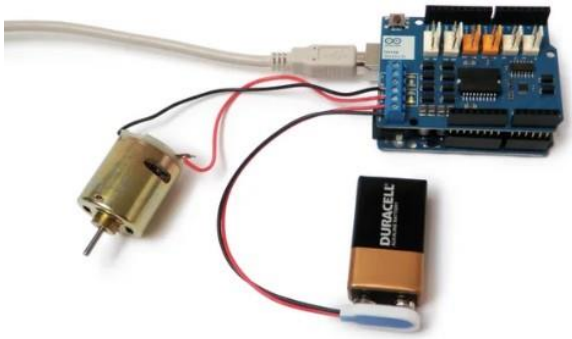
Motor Shield

CIRCUIT DIAGRAM

DC Motor connected with arduino using Motor Shield







SOURCE CODE

```
void setup()
 {
            //Setup Channel A
          pinMode(12, OUTPUT); //Initiates Motor Channel A pin
          pinMode(9, OUTPUT); //Initiates Brake Channel A pin

}
void loop()
{

           //forward @ full speed
          digitalWrite(12, HIGH); //Establishes forward direction of Channel A
          digitalWrite(9, LOW); //Disengage the Brake for Channel A
          analogWrite(3, 255); //Spins the motor on Channel A at full speed

          delay(3000);

          digitalWrite(9, HIGH); //Eengage the Brake for Channel A
          delay(1000);

          //backward @ half speed
          digitalWrite(12, LOW); //Establishes backward direction of Channel A
          digitalWrite(9, LOW); //Disengage the Brake for Channel A
          analogWrite(3, 123); //Spins the motor on Channel A at half speed
```

```
        delay(3000);
        digitalWrite(9, HIGH); //Eengage the Brake for Channel A

        delay(1000);

}
```

Interfacing Node McU with Cloud (Thingspeak API)

Interfacing Node Mcu with Cloud (Thingspeak API) and sending temperature and humidity data to visualize in Thingspeak channel.

HARDWARE

- Node MCU,
- USB-A to USB-C cable.
- DHT11 (temperature & Humidity) Sensor
- Jumper wires
- Breadboard

SOFTWARE

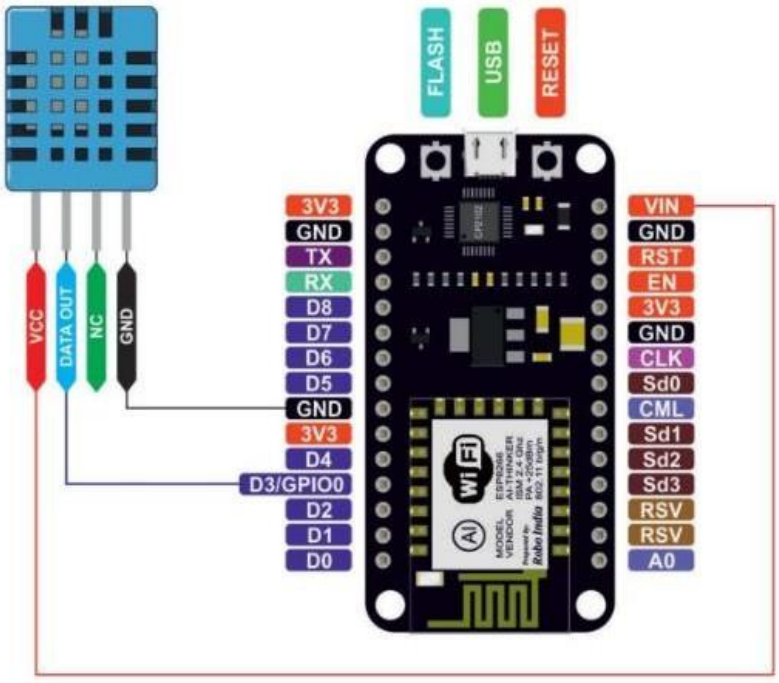
Arduino IDE, Thing Speak (Cloud) Channel.

ABOUT THING SPEAK

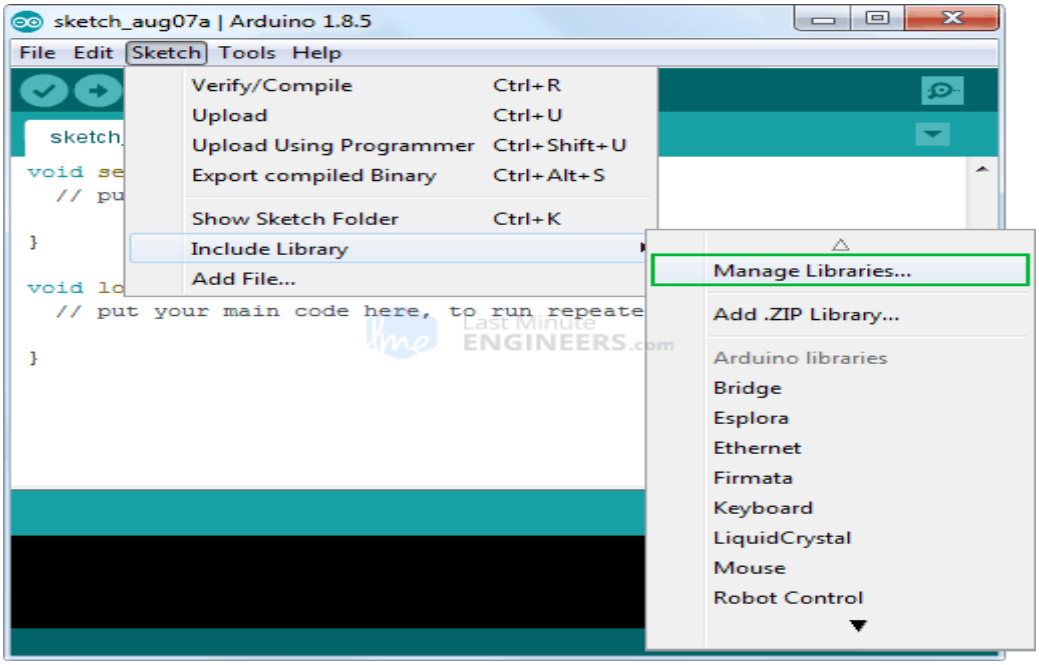
ThingSpeak is an IoT analytics service that allows you to aggregate, visualize, and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak.ThingSpeak is often used for prototyping and proof-of-concept IoT systems that require analytics.

You can send data from any internet-connected device directly to ThingSpeak using a Rest API or MQTT. In addition, cloud-to-cloud integrations with The Things Network, Senet, the LibeliumMeshlium gateway, and Particle.io enable sensor data to reach ThingSpeak over LoRaWAN® and 4G/3G cellular connections.

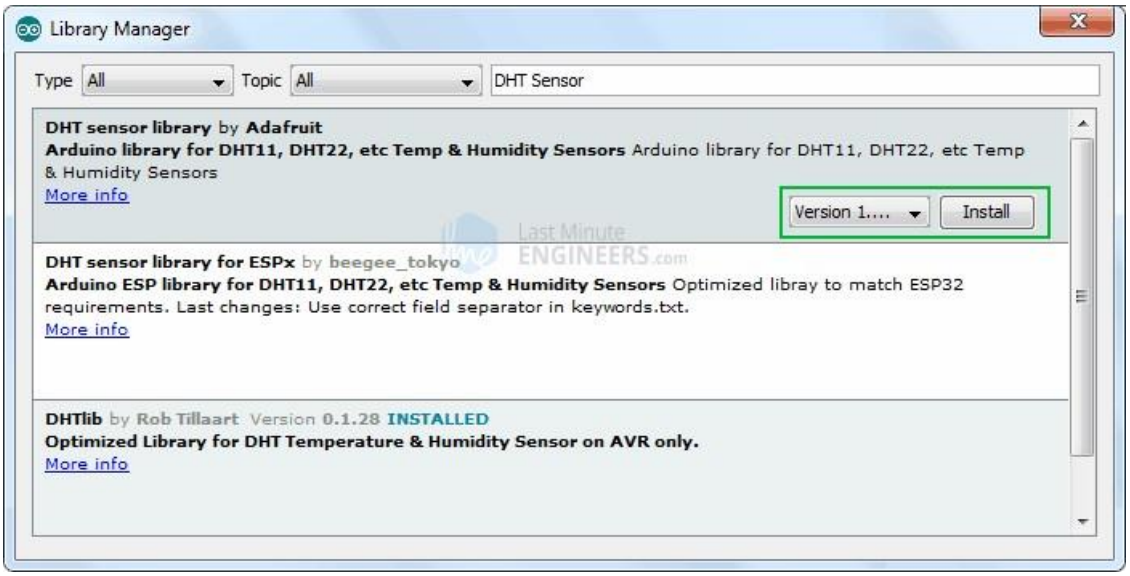
CIRCUIT DIAGRAM



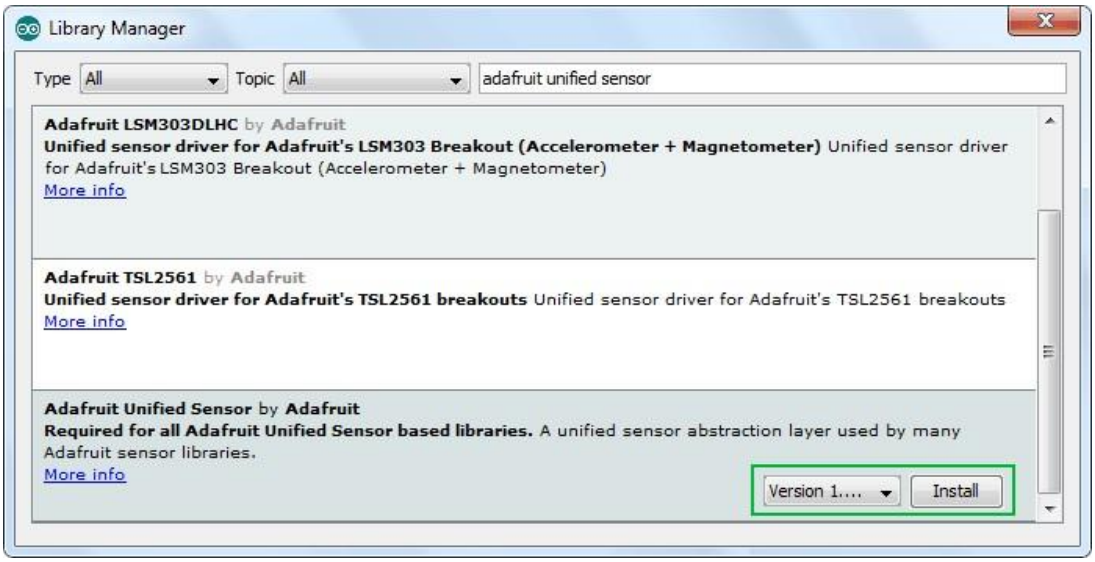
Installing DHT Sensor Library



Filter your search by entering 'DHT sensor'. Look for the DHT sensor library by Adafruit. Click on that entry and then choose Install.



The DHT sensor library makes use of the [Adafruit Sensor support backend]. So, look for Adafruit Unified Sensor in the library manager and install it as well.



PROCEDURE

Creating an ESP8266 Web Server using WiFi Station (STA) mode

1. Go to https://thingspeak.com/ and create an account if you do not have one. Login to your account.

2. Create a new channel by clicking on the button.Enter basic details of the channel.Than Scroll down and save the channel.

3. Channel Id is the identity of your channel. Note down this. Than go to API keys copy and paste this key to a separate notepad file will need it later.

Once the circuit part is done, NodeMCU is needed to be programmed

SOURCE CODE

```
#include <DHT.h>  // Including library for dht

#include <ESP8266WiFi.h>

String apiKey = " 60XU84DUSJAIOZ2R
";     // Enter your Write API key from ThingSpeak

const char *ssid =  "H-BOTS";     // replace with your wifissid and wpa2 key
const char *pass = "password";
const char* server = "api.thingspeak.com";

#define DHTPIN 0          //pin where the dht11 is connected

DHT dht(DHTPIN, DHT11);

WiFiClient client;

void setup()
{
Serial.begin(115200);
delay(10);
dht.begin();
Serial.println("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, pass);

while (WiFi.status() != WL_CONNECTED)
    {
delay(500);
Serial.print(".");
    }
Serial.println("");
Serial.println("WiFi connected");

}

void loop()
{

float h = dht.readHumidity();
float t = dht.readTemperature();

if (isnan(h) || isnan(t))
              {
Serial.println("Failed to read from DHT sensor!");
return;
              }

if (client.connect(server,80))    //    "184.106.153.149" or api.thingspeak.com
```

```
                {

                        String postStr = apiKey;
postStr +="&field1=";
postStr += String(t);
postStr +="&field2=";
postStr += String(h);
postStr += "\r\n\r\n";

client.print("POST /update HTTP/1.1\n");
client.print("Host: api.thingspeak.com\n");
client.print("Connection: close\n");
client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
client.print("Content-Type: application/x-www-form-urlencoded\n");
client.print("Content-Length: ");
client.print(postStr.length());
client.print("\n\n");
client.print(postStr);

Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" degrees Celcius, Humidity: ");
Serial.print(h);
Serial.println("%. Send to Thingspeak.");
                  }
client.stop();

Serial.println("Waiting...");

  // thingspeak needs minimum 15 sec delay between updates, i've set it to 30 seconds
delay(10000);
}
```

THINGSPEAK VIEW

dht11

Channel ID: **1756546**
Author: mwa0000026683343
Access: Private

temperature and humidity

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATLAB Visualization

Channel Stats

Created: 5 months ago
Last entry: 19 minutes ago
Entries: 3