Week 1:

1. Introduction to Arduino Uno – Sensors & Actuators
   a. Temperature & Humidity Sensors
   b. Air Quality Sensor
   c. PIR Motion Sensor
   d. Micro Servo Motor
   e. Stepper Motor
   f. 100 RPM Motor

Week 2:

2. Introduction to NodeMCU – Sensors & Actuators
   a. Temperature & Humidity Sensors
   b. Air Quality Sensor
   c. PIR Motion Sensor
   d. Micro Servo Motor
   e. Stepper Motor
   f. 100RPM Motor

Week 3:

3. Setting up your Raspberry Pi. Installation of software.
4. Introduction to Raspberry Pi – Sensors & Actuators
   a. Temperature & Humidity Sensor
   b. Ultrasonic Sensor
   c. Micro Servo Motor

Week 4:

5. Introduction to IoT & Sensor control with IFTTT

Week 5:

6. Open Source Cloud Platforms for IoT: thinger.io, Google Cloud Platform.

Week 6:

7. Introduction to Open Web Services for IoT
8. Experiments with Open Web Services for IoT:
   a. M2M Labs
   b. The ThingBox
   c. The Thing System
   d. Node-RED

Week 1:
Introduction to Arduino Uno – Sensors & Actuators

The Arduino Software (IDE) allows you to write programs and upload them to your board. In the Arduino Software page you will find two options:

1. If you have a reliable Internet connection, you should use the online IDE (Arduino Web Editor). It will allow you to save your sketches in the cloud, having them available from any device and backed up. You will always have the most up-to-date version of the IDE without the need to install updates or community generated libraries.

2. If you would rather work offline, you should use the latest version of the desktop IDE

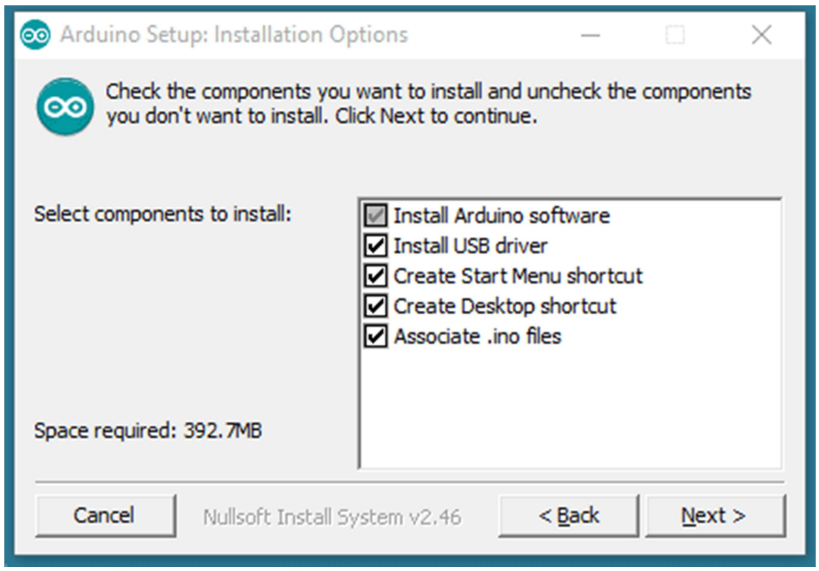
Install the Arduino Software (IDE) on Windows PCs

This document explains how to install the Arduino Software (IDE) on Windows machines

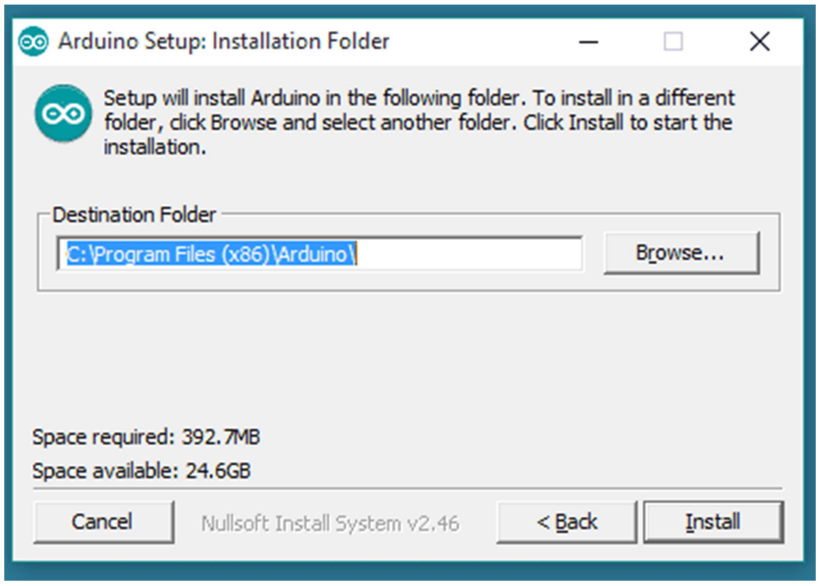
Download the Arduino Software (IDE)

Get the latest version from the download page. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.

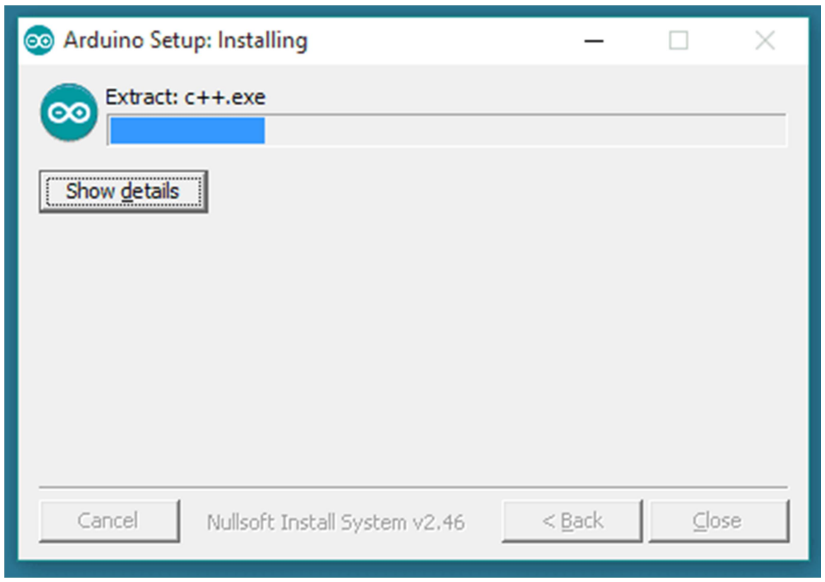
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



Choose the components to install



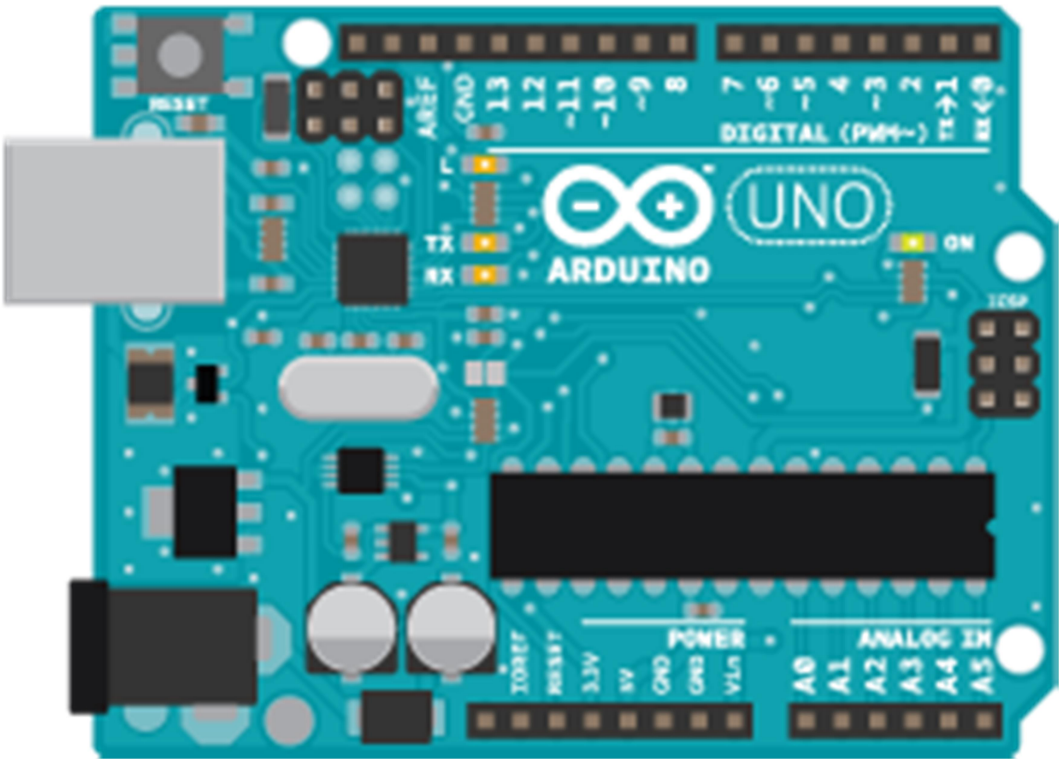
Choose the installation directory (we suggest to keep the default one)



The process will extract and install all the required files to execute properly the Arduino Software (IDE)

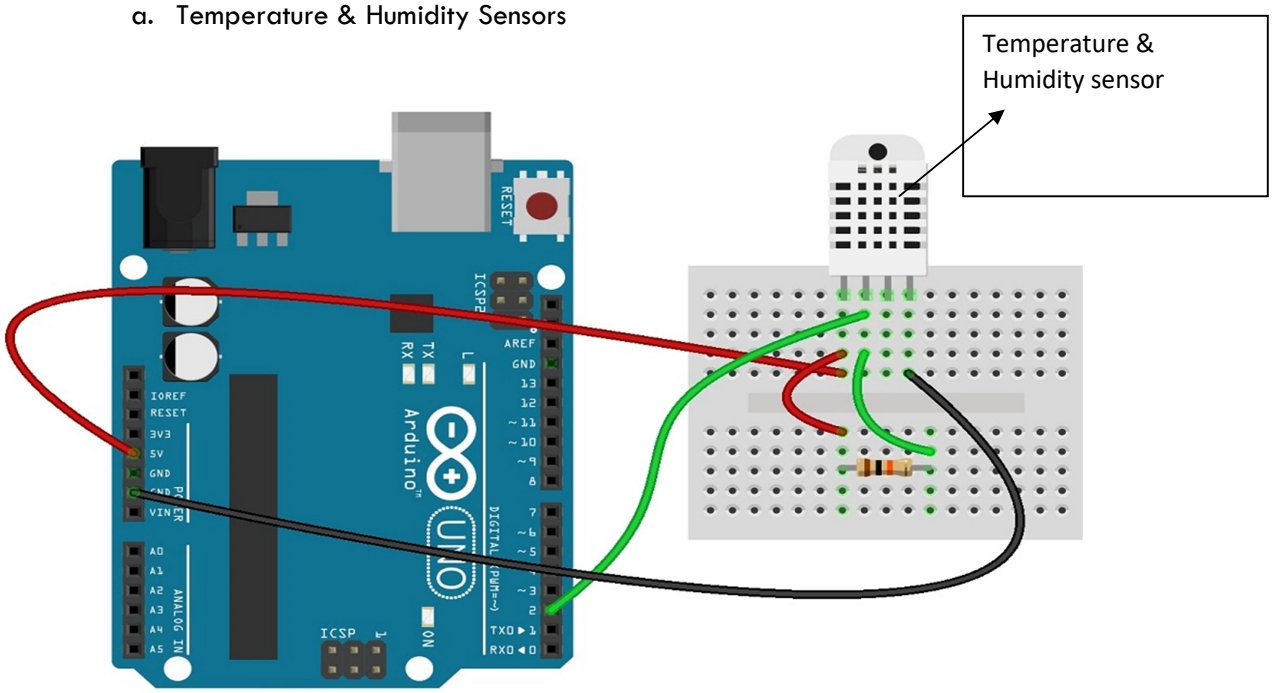
WHAT IS ARDUINO?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.



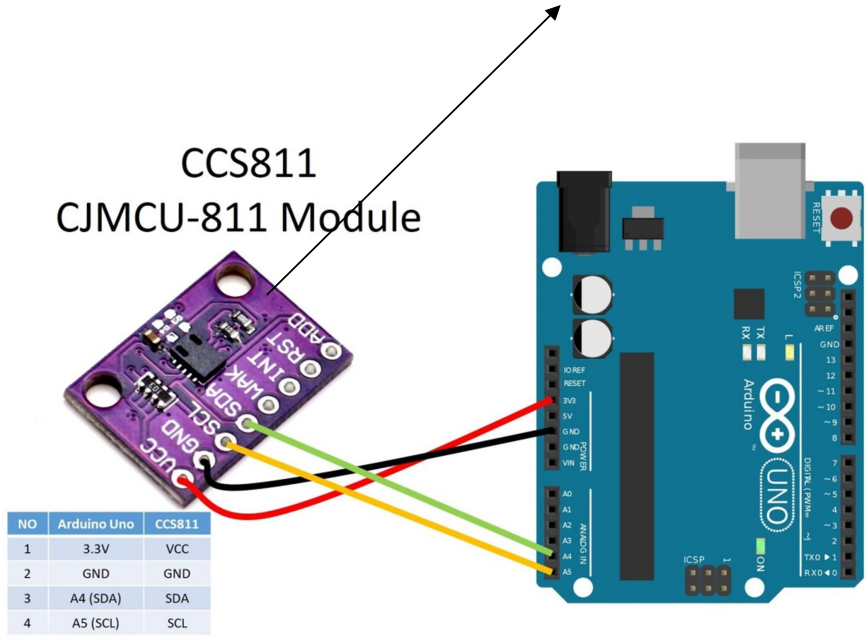
– Sensors & Actuators

a. Temperature & Humidity Sensors



Air Quality Sensor

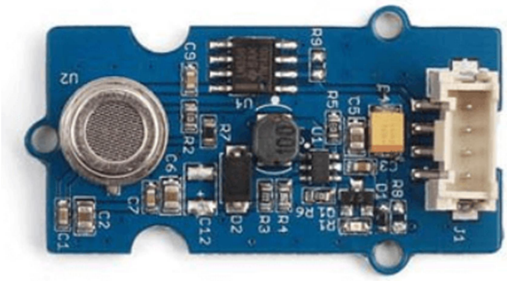
Air Quality Sensors



NO	Arduino Uno	CCS811
1	3.3V	VCC
2	GND	GND
3	A4 (SDA)	SDA
4	A5 (SCL)	SCL

Air Quality Sensor Modules

Grove – Air Quality Sensor v1.3 – Arduino Compatible



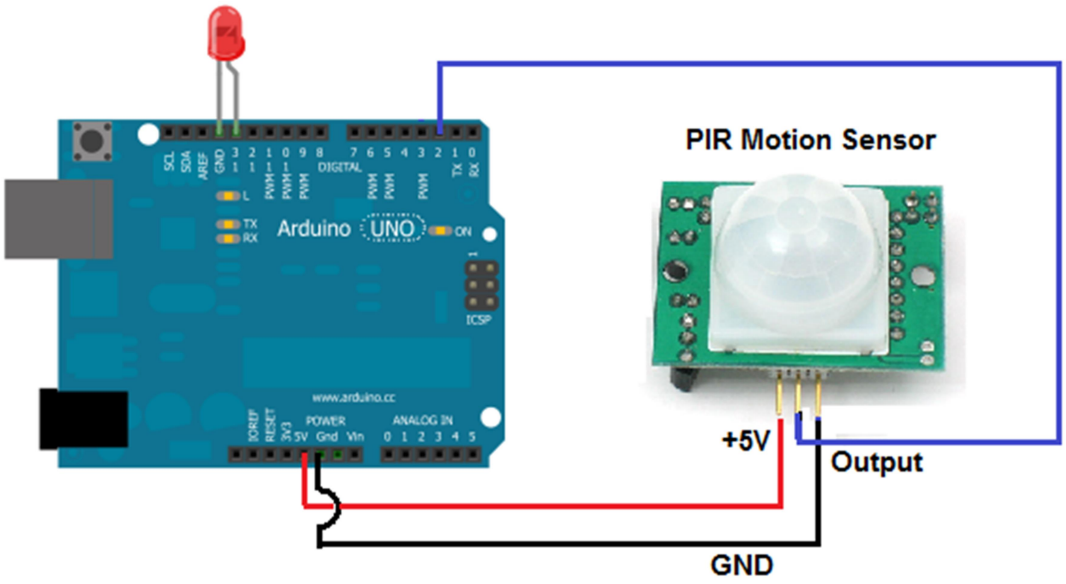
Grove – Air quality sensor v1.3 is designed for indoor air quality testing. It can respond to carbon monoxide, alcohol, acetone, thinner, formaldehyde, and other slightly toxic gases. Compatible with 5V and 3.3V power supply, it can work with Arduino and Raspberry Pi. With its long-term stability & low power consumption, it would be a perfect choice for air quality monitoring. With a tiny outline as well, you can easily integrate it into your air quality monitor or system.

This sensor performs better in providing qualitative results over a wide scope of target gases. Over time, do note that if this sensor is exposed to highly polluted air for a long time, it might weaken its sensitivity greatly.

Specification

Sensor	MP503
Operating Voltage	3.3V, 5V
Detecting Range	10-1000ppm(Alcohol)
Interface	Analog

PIR Motion Sensor

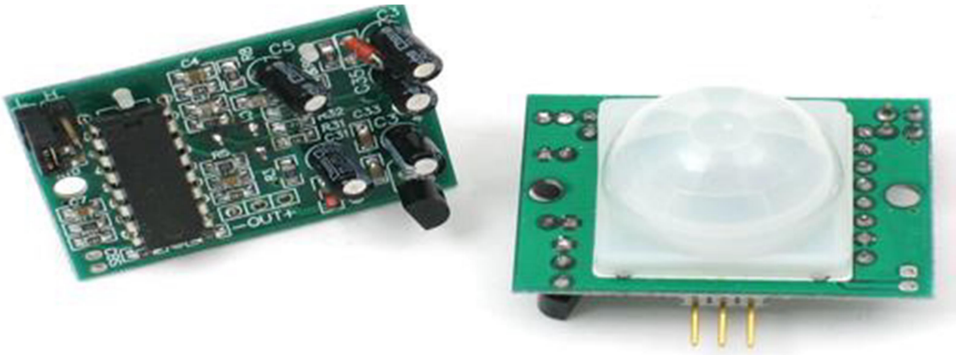


PIR sensors allow you to sense motion. They are used to detect whether a human has moved in or out of the sensor's range. They are commonly found in appliances and gadgets used at home or for businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.

Following are the advantages of PIR Sensors −

- Small in size
- Wide lens range
- Easy to interface
- Inexpensive

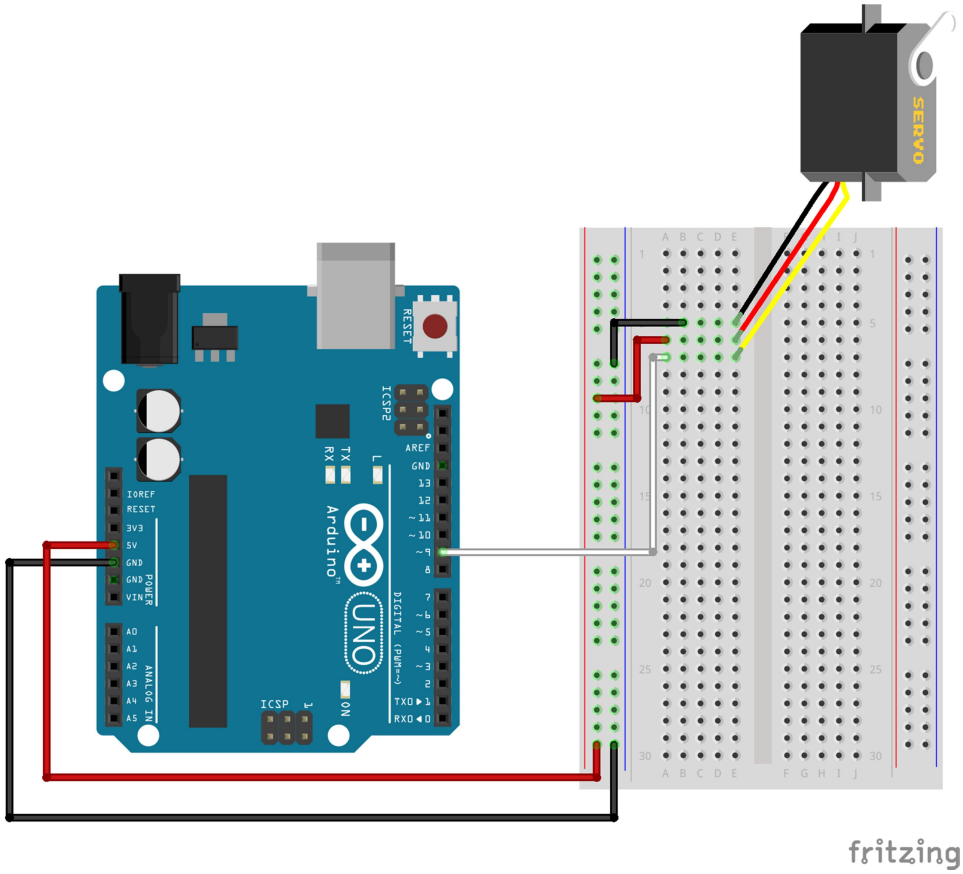
- Low-power
- Easy to use
- Do not wear out



PIRs are made of pyroelectric sensors, a round metal can with a rectangular crystal in the center, which can detect levels of infrared radiation. Everything emits low-level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is split in two halves. This is to detect motion (change) and not average IR levels. The two halves are connected so that they cancel out each other. If one-half sees more or less IR radiation than the other, the output will swing high or low.

Micro Servo Motor

Servo is a type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your Arduino Uno board. These pulses tell the servo what position it should move to.



Week 1 & 2:

1. Setting up your Raspberry Pi. Installation of software.
2. Introduction to Raspberry Pi & Sensors a. Temperature Sensors b. Proximity Sensors c. Pressure Sensors
3. Introduction to Sensors & Actuators
   a. Humidity Sensors
   b. Accelerometer & Gyroscope
   c. Actuators Ex: Motors

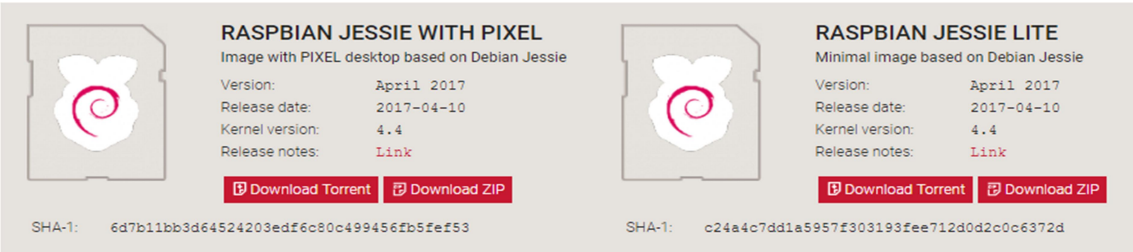
How to install Raspbian on the Raspberry Pi

Installing Raspbian on the Raspberry Pi is pretty straightforward. We'll be downloading Raspbian and writing the disc image to a microSD card, then booting the Raspberry Pi to that microSD card. For this project, you'll need a microSD card (go with at least 8 GB), a computer with a slot for it, and, of course, a Raspberry Pi and basic peripherals (a mouse, keyboard, screen, and power source). This isn't the only method for installing Raspbian (more on that in a moment), but it's a useful technique to learn because it can also be used to install so many other operating systems on the Raspberry Pi. Once you know how to write a disc image to a microSD card, you open up a lot of options for fun Raspberry Pi projects.

A word about NOOBS – An easier way

It's worth noting that the method described here isn't your only option for installing Raspbian. You can also opt to use NOOBS, an operating system installation manager that makes it easy to install Raspbian, as well as a few other operating systems. If you really want to make things easy, you can even buy SD cards that come pre-loaded with NOOBS. For a bit more on that, check out how to install NOOBS at page number 4.

Step 1: Download Raspbian



I promised to show you how to install Raspbian on the Raspberry Pi, so it's about time that we got started! First things first: hop onto your computer (Mac and PC are both fine) and download the Raspbian disc image. **You can find the latest version of Raspbian on the Raspberry Pi Foundation's website here.** Give yourself some time for this, especially if you plan to use the traditional download option rather than the torrent. It can easily take a half hour or more to download.

Raspbian OS Source: https://www.raspberrypi.org/downloads/raspbian/

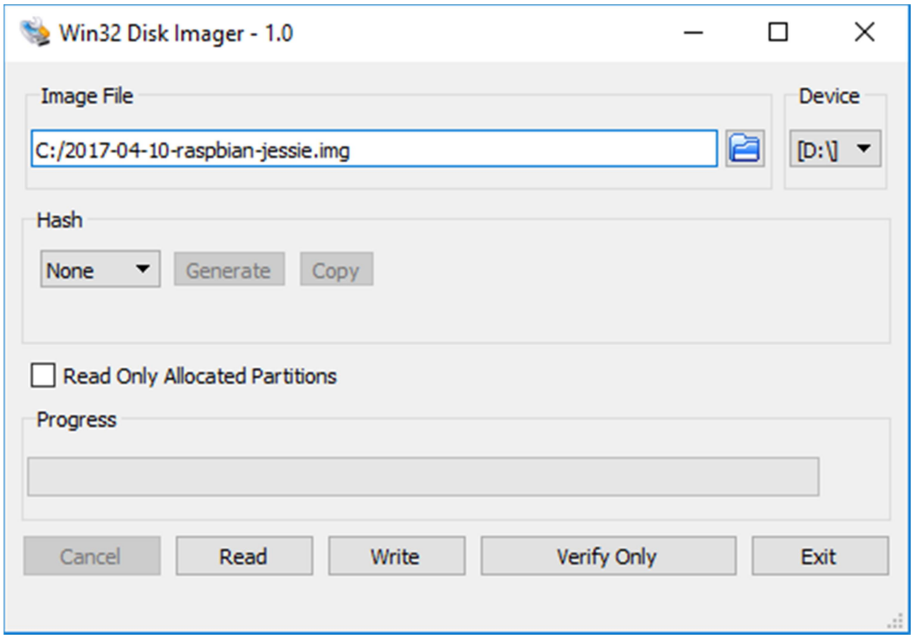
Step 2: Unzip the file

The Raspbian disc image is compressed, so you'll need to unzip it. The file uses the ZIP64 format, so depending on how current your built-in utilities are, you need to use certain programs to unzip it. If you have any trouble, try these programs recommended by the Raspberry Pi Foundation:

- Windows users, you'll want **7-Zip.**
- Mac users, **The Unarchiver** is your best bet.
- Linux users will use the appropriately named **Unzip**.

Sources:
7-Zip: http://www.7-zip.org/
The Unarchiver: http://unarchiver.c3.cx/unarchiver
Unzip: http://www.info-zip.org/mans/unzip.html

Step 3: Write the disc image to your microSD card



Next, pop your microSD card into your computer and write the disc image to it. You'll need a specific program to do this:

- Windows users, your answer is **Win32 Disk Imager**.
- Mac users, you can use the disk utility that's already on your machine.
- Linux people, **Etcher** – which also works on Mac and Windows – is what the Raspberry Pi Foundation recommends.

Sources:
Win32 Disk Imager: https://sourceforge.net/projects/win32diskimager/
Etcher: https://etcher.io/

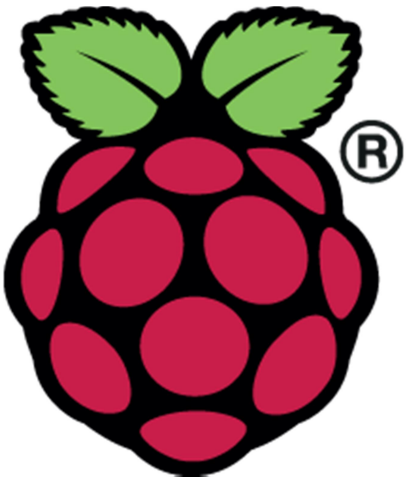
The process of actually writing the image will be slightly different across these programs, but it's pretty self-explanatory no matter what you're using. Each of these programs will have you

select the destination (make sure you've picked your microSD card!) and the disc image (the unzipped Raspbian file). Choose, double-check, and then hit the button to write.

Step 4: Put the microSD card in your Pi and boot up

Once the disc image has been written to the microSD card, you're ready to go! Put that into your Rasberry Pi, plug in the peripherals and power source, and enjoy. The current edition to Raspbian will boot directly to the desktop. Your default credentials are username **pi** and password **raspberry.**

How to install NOOBS on the Raspberry Pi



The Raspberry Pi is an incredible device, but it won't do much of anything without an operating system. Luckily, choosing and installing an appropriate operating system on your Raspberry Pi has never been easier. One simple method is to use NOOBS, or "New Out of Box Software." As the name suggests, NOOBS is perfect for Pi newbies. It lets you choose your preferred operating system and install it right then and there. But how do you load NOOBS itself? Here's our complete guide on how to install NOOBS on the Raspberry Pi.

Luckily for us, the process is extremely simple. All you'll need is a Raspberry Pi, a computer, and an SD or microSD card. Check out the complete instruction below.

How to install NOOBS on the Raspberry Pi

We've called our article "How to install NOOBS on the Raspberry Pi," but what we're technically doing is installing it on a flash drive, booting to the drive on the Raspberry Pi, and then using NOOBS to choose and install an operating system.

NOOBS has plenty of operating systems for us to choose from when we reach that step – the most notable of which is Raspbian. For now, though let's concentrate on how to install NOOBS on the Raspberry Pi. We will briefly discuss the operating system installations later, in our final step.

The optional easy route: buy a NOOBS SD card.

Installing NOOBS on an SD card isn't hard, but it also isn't necessary. If you'd like, you can choose to buy an SD card that comes pre-loaded with NOOBS. If you go that route, you can skip all the way to the final step!

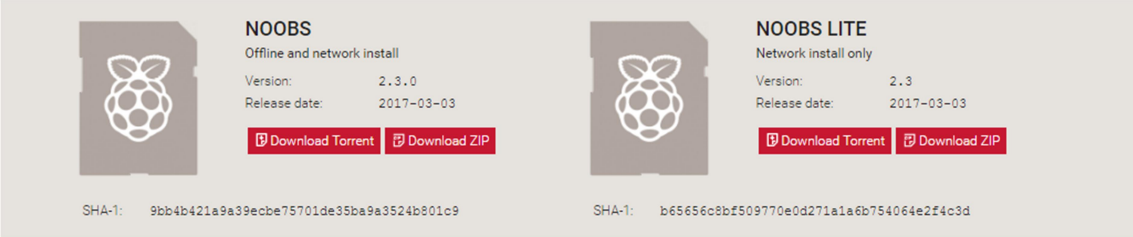
If you want to do things yourself, though, just read on.

What you'll need to install NOOBS on the Raspberry Pi

This project is pretty simple. Besides your Raspberry Pi and essential peripherals, here's all you'll need:

- A computer with an SD card slot
- An SD or microSD card of at least 8 GB

Step 1: Download NOOBS and extract it



You're going to use your computer to put NOOBS on an SD card – so step one is to get NOOBS onto your computer!

Link to NOOBS download page: https://www.raspberrypi.org/downloads/noobs/

The NOOBS download page will let you choose between NOOBS and "NOOBS Lite." NOOBS includes a full version of Raspbian, so you can install that particular operating system without using the internet at all. With NOOBS Lite, on the other hand, you'll need a network connection to install any of the operating systems NOOBS makes available – even Raspbian.

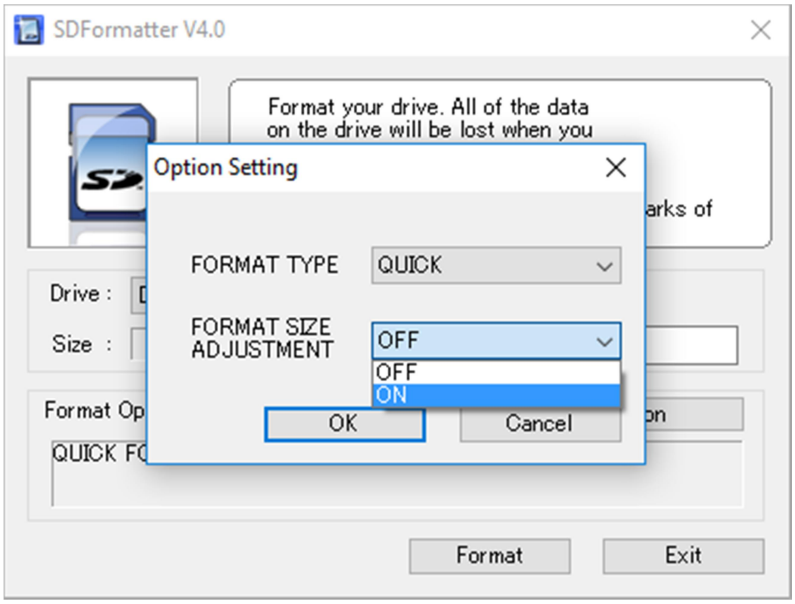
Go ahead and choose whichever version you would like. NOOBS will download as a .zip file, so before you do anything else, go ahead and extract it.

Step 2: Format an SD card

Now you're going to want to go ahead and stick your SD card into the corresponding slot on your computer. You're going to want to format it as FAT. There are a few ways to do this:

On Mac or Windows, use the **SD Association's Formatting Tool** (Mac users can also just use the disk utility). Make sure the "Format size adjustment" option is set to "on." Then erase it in FAT (or MS-DOS) format.

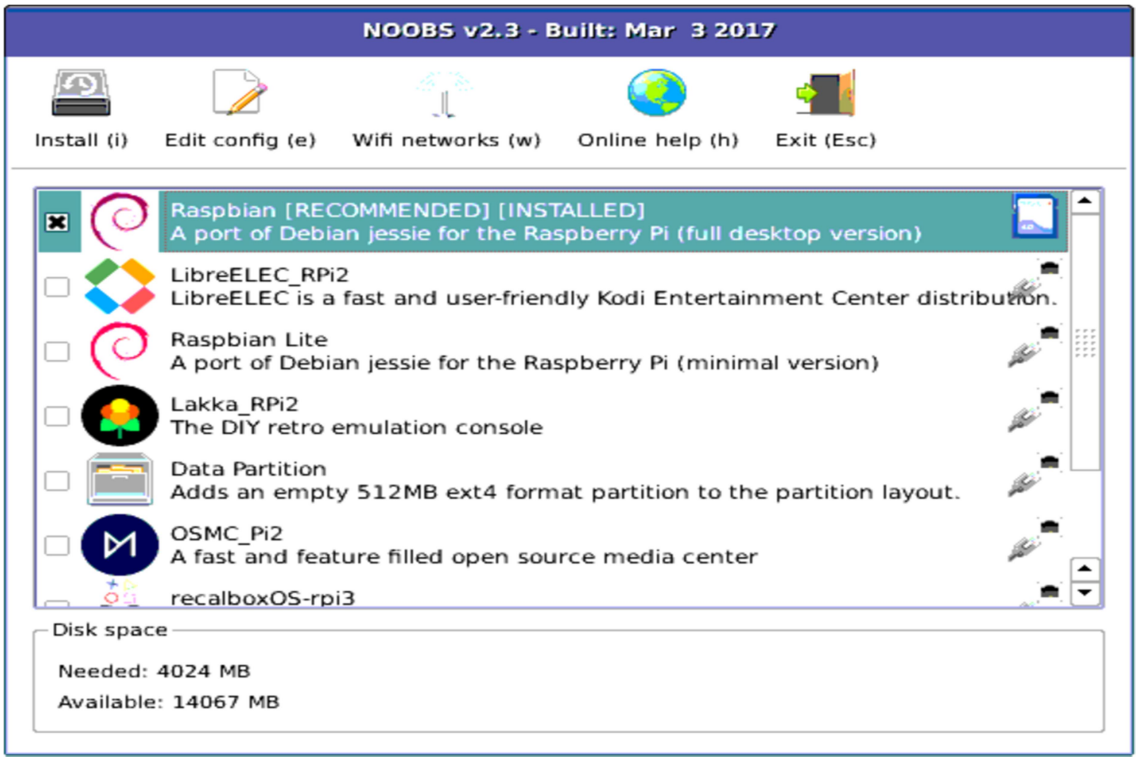
Source: https://www.sdcard.org/downloads/formatter_4/eula_windows/



Step 3: Put the NOOBS files on the SD card

Now, just drag and drop the NOOBS files into your newly formatted SD card. You want the files only, so if your .zip extracted to a folder, open that folder up and select only the stuff inside of it.

Step 4: Put your SD card into your Raspberry Pi and boot it up



Once you have NOOBS on your SD card, using it is incredibly easy. Just put the SD card into your Raspberry Pi and start that sucker up. As we said before, while this guide is called “How to install NOOBS on the Raspberry Pi,” the endgame here is actually to install an operating system like Raspbian, LibreELEC, OSMC, or any of the others NOOBS gives you access to.

This is the step in which that happens. After booting to NOOBS, you'll be greeted with a menu that will let you choose which operating system you'd like to install on your Pi. Your menu may look a little bit different than the one in the screenshot above, because NOOBS ingeniously adapts to your generation and model of Raspberry Pi.

Which OS should you choose? Well, that's up to you. Raspbian is probably the most frequently used, and you'll find plenty of projects here on our site that utilize it. OSMC acts as a media center, and LibreELEC boots directly to the popular media center app Kodi. Ultimately, it's all a matter of personal preference!

Once you've decided, just hit "Install" and sit back. From now on, your Pi will boot directly to that operating system. Easy, right?

And if you're not happy with the operating system you pick, you're not stuck. Just hold down the SHIFT key while booting up, and you'll be back in the NOOBS menu ready to try out a different option.

2.a)Temperature Sensors

Sensor Data Displaying on Serial Monitor Using Arduino UNO

Practical's Objective:

Tocapturedatafromdifferentsensorsanddisplaythesameon SerialmonitorusingArduinoUNO.

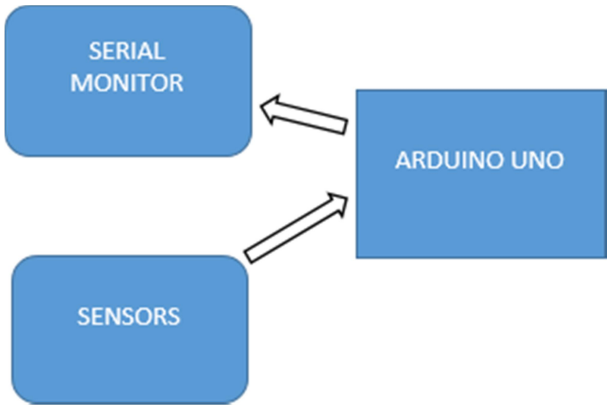
Description of Sensor Board Elements:

A**Temperaturesensor**(Sensorname:LM35D)isusedtomonitorthetemperatureofthematerial.

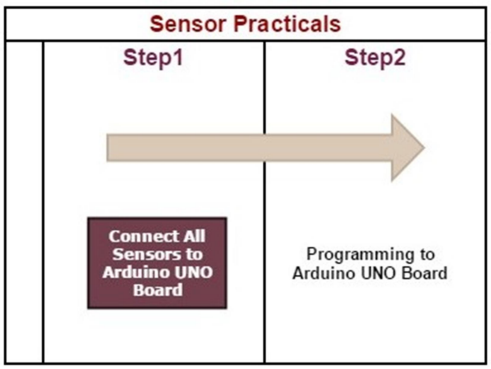
A**LightDependantResistor**((Sensorname:LDR)isusedtoknowtheamountoflightpresent.

Note:Asperitsnames,thetechnicaldatasheetsofthesensorscanbefoundintheinternetfor moreinformation.

1. **SensorPracticalHardwareflowDiagram:**



2. **SensorPracticalToDo:**



3. **H/w&S/wRequirements:**

Hardware required:

- Arduino UNOBoard
- Temperaturesensor
- HumiditySensor
- LDRsensor
- Connectors
- USBcable

Software Requirement:

- Aurdino IDE -**http://arduino.cc/en/Main/OldSoftwareReleases**

ConnectingSensorstoArduinoBoard:

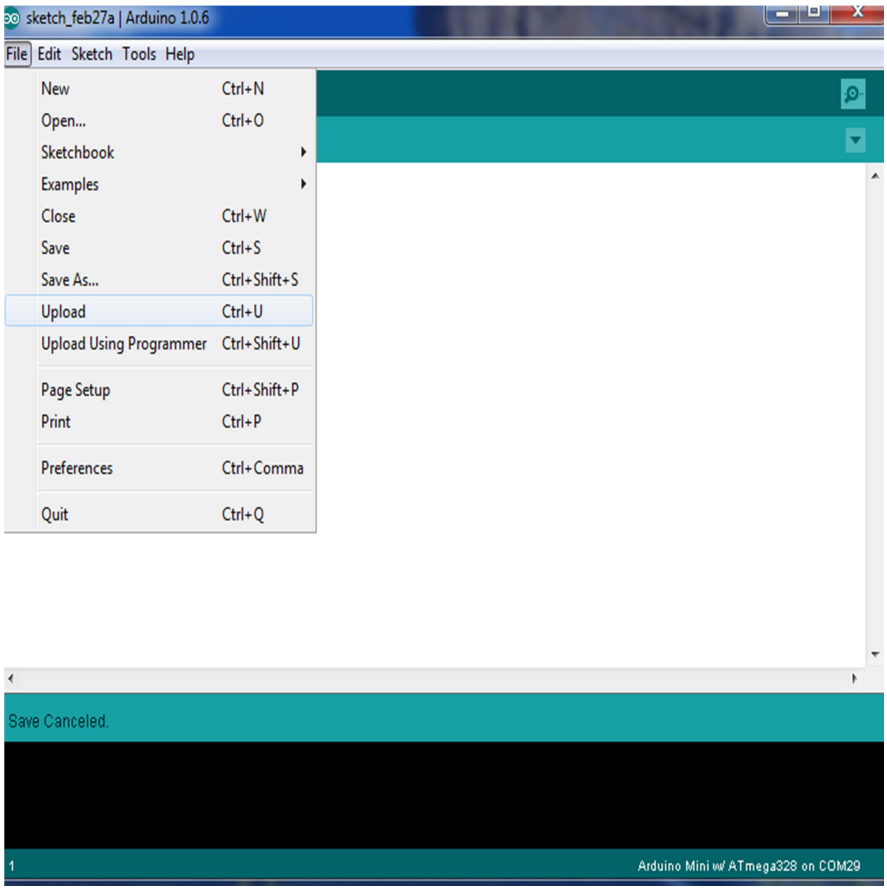
SensorBoard	LCDShield
T	A2
A0	A0
Vcc	5V
GND	GND

- ConnectTemperaturesensorOutputpinto**A2pin**onArduinoUNOBoardusingasinglepinwire. Also,connectlightsensoroutputpintoA0pinonArduinoUNOBoardusingsinglepinwire,
- ConnectUSBcablefromArduinoUNOboardtoPC.

Note: Beforegettingstarted,checkthattheArduinosoftwareisinstalled.Whenconnectedforthe firsttime,checktheArduinoUNOboardUSBdriversoftwareisupdatedaspertheInstallation document.

Programming:

1. StartArduinoIDEbyselectingtheprograminthewindowsstartmenu.YouwillgettheWindowas below.



2. Stepsforconstructingasimple‘C’programinArduino.

Step1: Create a new sketch [program]

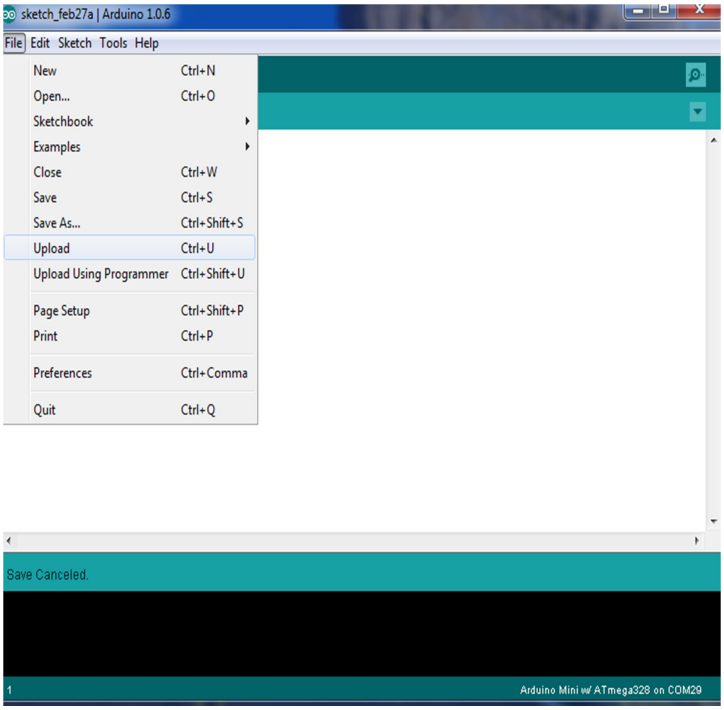
```
int T = A2;      // Power Switch
// A2 has already been defined as Analog Pin2 in Arduino Library
int L = A0;
void setup()
{
/********** Analog Read function*********/
void Temperature()
{
```

```
        int value_temp = analogRead(T);       // reading from A2 pin
        float millivolts_temp = ((value_temp/1023.0)*5000);
        float Celsius = millivolts_temp/10;
        Serial.println(Celsius);
}
void LIG()
{
        int value_lig = analogRead(T);          //reading from A2 pin
        delay(10);
        value_lig = analogRead(L);
        float volts_lig = (value_lig/1023.0)*5;
        // Calculate the Lux = 500/[R1*((Vin - Vsense)/Vsense)]
        int LIGHT = 500/(4*((5-volts_lig)/volts_lig));
        lcd.setCursor(0,1);
        lcd.print("L:");
        lcd.print(LIGHT);
        lcd.print("Lx");
        Serial.print("Light: ");
        Serial.print(LIGHT);
        Serial.println(" Lux");
        delay(2000);
        lcd.clear();
}
        void loop()
{
        Temperature();
        LIG();
        delay(2000);
}
```

Step10:

Now save theproject:Goto**File**menu,youcanfind**save**optionasthewindowsshownbelow Clickon**save**,giveaFilenameandclick**ok**.

Nowgotothe**File**menu,clickon**Upload**option.Thenthecodeiscompiledandloadedinto Arduino UNOboard.



IfyougetanyissueswiththeuploadingtheremightbeaprobleminttheselectionofCOMportor withthedrivers.RefertotheArdunioenvironmentsetupmanualtomakesurethattherearenо issues.

2.b) Proximity Sensors

Proximity Sensor Data Displaying on output screen Using Raspberry-Pi

Practical's Objective:

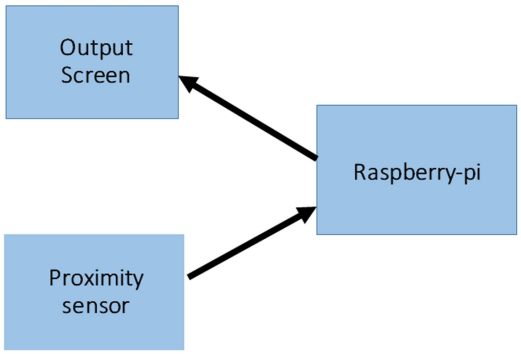
To capture data from sensor and display the same on output screen using raspberry-pi.

Description of Sensor Board Elements:

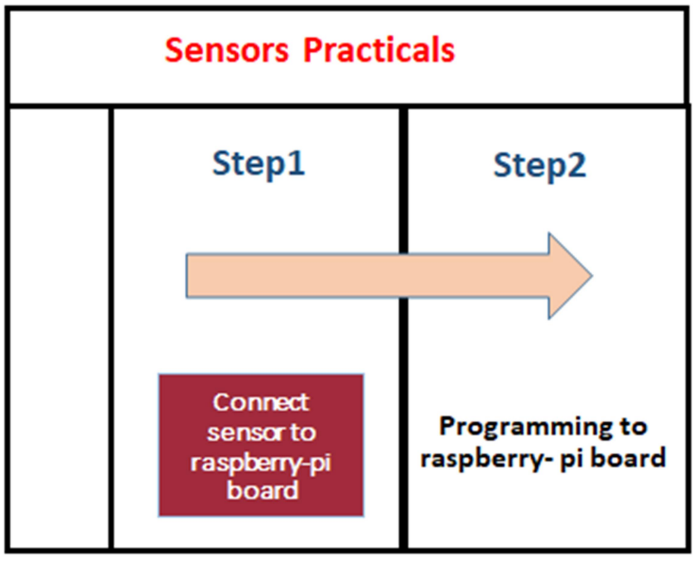
A Proximity sensor is used to detect presence of human beings.

Note: As per its names, the technical datasheets of the sensors can be found in the internet for more information.

1. **Sensor Practical Hardware flow Diagram:**



2. **Sensor Practical To Do:**



3. **H/w & S/w Requirements:**

Hardware required:

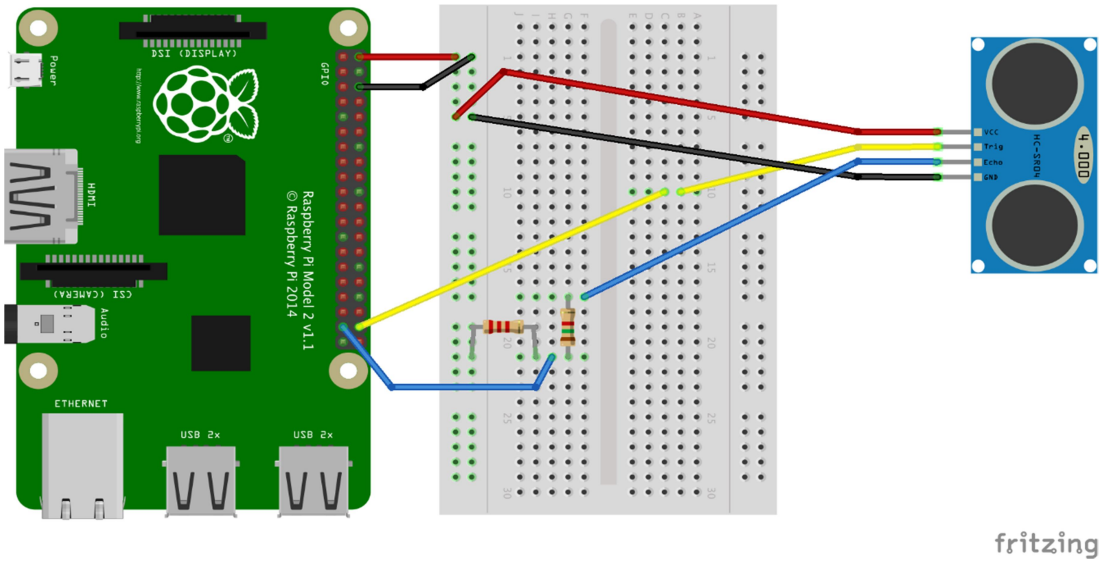
- Raspberry-pi Board
- Proximity sensor
- 2A charger
- Connectors
- LAN cable

4. **Software Requirement:**

Python IDE 2 –

5. **Connecting Sensors to Raspberry-Pi Board:**

Sensor	Raspberry-pi
Vcc	5v(pin 2)
GND	GND(pin 6)
OUT	GPIO17(pin 11)



6. **Programming:**

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
GPIO.setup( 17, GPIO.IN )
while(1):
pir = GPIO.input(17)
if(pir):
print 'Motion Detected'
pir_s = True
```

```
sleep(1)
else:
print 'No Motion'
pir_s = False
sleep(1)
```

2.c)Pressure Sensors

Pressure Sensor Data Displaying on output screen Using Raspberry-pi

Practical's Objective:

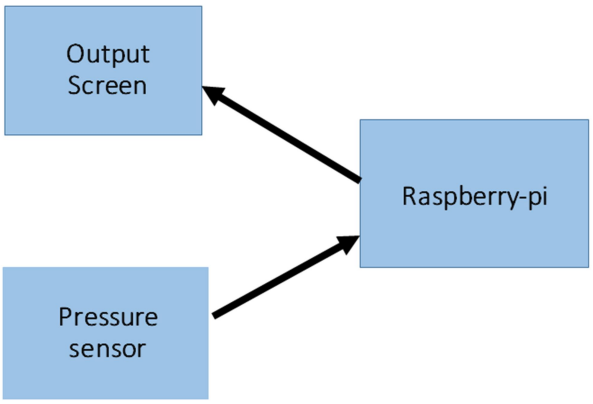
To capture data from sensor and display the same on output screen using raspberry-pi.

Description of Sensor Board Elements:

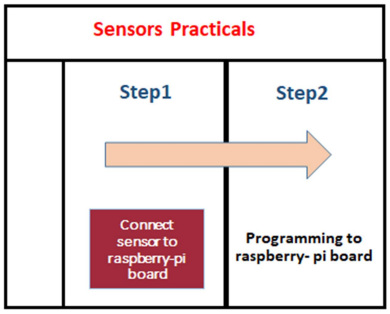
A Pressure sensor is used to measures the **pressure** of a gas or a liquid against a diaphragm made of stainless steel, silicon, etc.

Note: As per its names, the technical datasheets of the sensors can be found in the internet for more information.

1. **Sensor Practical Hardware flow Diagram:**



2. **Sensor Practical To Do:**



3. **H/w & S/w Requirements:**

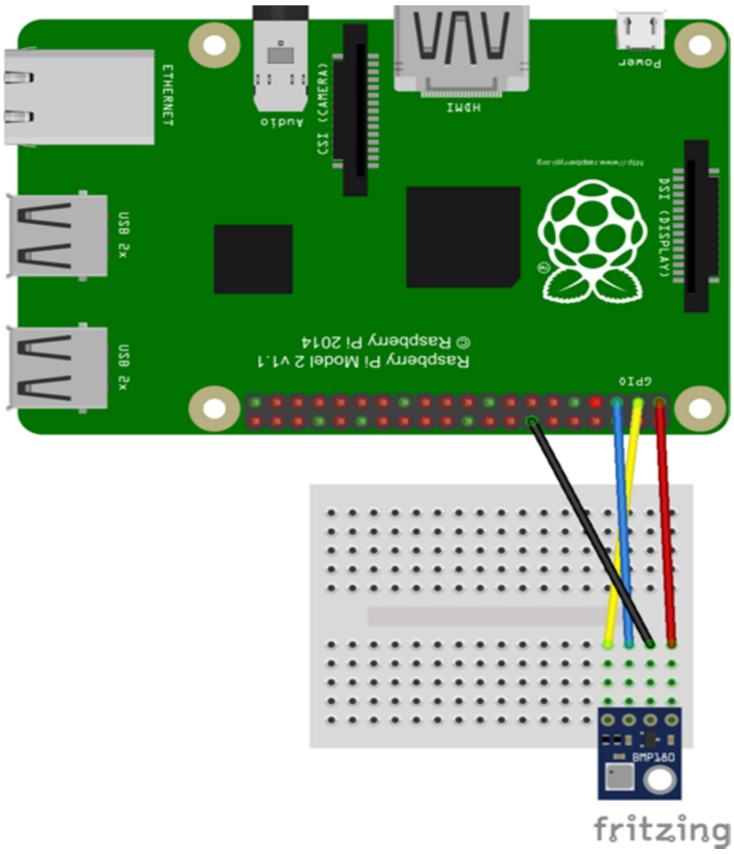
   Hardware required:
   - Raspberry-pi Board
   - Pressure sensor
   - 2A charger
   - Connectors
   - LAN cable

4. **Software Requirement:**

   Python IDE 3 –

5. **Connecting Sensors to Raspbery-pi Board:**

Sensor	Raspberry-pi
Vcc	3.3v(pin 1)
GND	GND(pin 6)
SCL	SCL (PIN 5)
SDA	SDA(piN 3)



Using the BMP180

- If everything has been installed okay, and everything has been connected okay, you are now ready to turn on your Pi and start seeing what the BMP180 is telling you about the world around you.
- The first thing to do is to check that the Pi sees your BMP180. Try the following in a terminal window:

  sudo i2cdetect -y 1
- If you receive an error message and have a very early Raspberry Pi Model B, try:

  sudo i2cdetect -y 0
- If the command worked, you should see the following:

```
pi@PiHutTutorials ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
pi@PiHutTutorials ~ $
```

This is showing that the BMP180 is on channel '77'.

Now we need to install some libraries so that Python knows how to read the output of the BMP180. The libraries were actually written by Adafruit for use with the predecessor of the BMP180, the BMP085, but since the chips were designed to be compatible with each other, the libraries work with this newer device.

To install these libraries and required Python libraries, type the following into a terminal window on your Pi:

- Install the GIT application which will be able to retrieve code from the GitHub (a website that is used to store and version control code):

```
sudo apt-get update
sudo apt-get install git build-essential python-dev python-smbus
```

- Create and move to a directory that will contain your code. This directory can be called anything you like:

```
mkdir ~/BMP180Code
cd ~/BMP180Code
```

- Download the GIT repository for the BMP180. This will download the contents of the Adafruit Python library to the current location:

```
git clone https://github.com/adafruit/Adafruit_Python_BMP.git
```

- The library now needs compiling so that you can use it. In the same terminal window as above, run the following:

```
cd Adafruit_Python_BMP
sudo python setup.py install
```

Now you are ready to use the BMP180.

6. Programming:

```
#!/usr/bin/python

import Adafruit_BMP.BMP085 as BMP085 # Imports the BMP library

# Create an 'object' containing the BMP180 data
sensor = BMP085.BMP085()

# Temperature in Celcius
print 'Temp = {0:0.2f} C'.format(sensor.read_temperature())

# The local pressure
print 'Pressure =
                {0:0.2f}Pa'.format(sensor.read_pressure())
```

```
# The current altitude
print 'Altitude = {0:0.2f}m'.format(sensor.read_altitude())

# The sea-level pressure
print 'Sealevel Pressure =
        {0:0.2f}Pa'.format(sensor.read_sealevel_pressure())
```

3.a) Humidity Sensors

Sensor Data Displaying on Serial Monitor Using Arduino UNO

Practical's Objective:

Tocapturedatafromdifferentsensorsanddisplaythesameon SerialmonitorusingArduinoUNO.

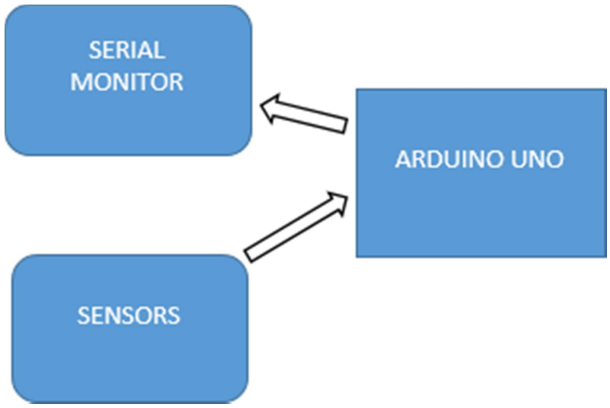
Description of Sensor Board Elements:

A**Temperaturesensor**(Sensorname:LM35D)isusedtomonitorthetemperatureofthematerial.

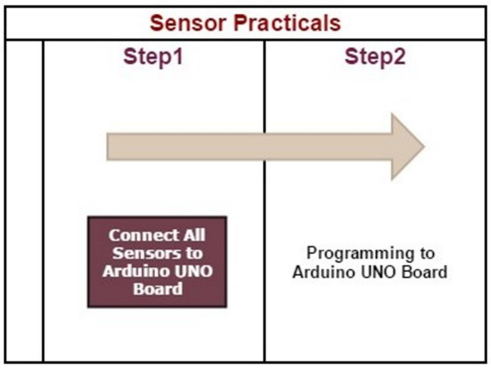
A**LightDependantResistor**((Sensorname:LDR)isusedtoknowtheamountoflightpresent.

Note:Asperitsnames,thetechnicaldatasheetsofthesensorscanbefoundintheinternetfor moreinformation.

4. SensorPracticalHardwareflowDiagram:



5. SensorPracticalToDo:



6. **H/w&S/wRequirements:**

Hardware required:

- Arduino UNOBoard
- Temperaturesensor
- HumiditySensor
- LDRsensor
- Connectors
- USBcable

Software Requirement:

- Aurdino IDE -**http://arduino.cc/en/Main/OldSoftwareReleases**

ConnectingSensorstoArduinoBoard:

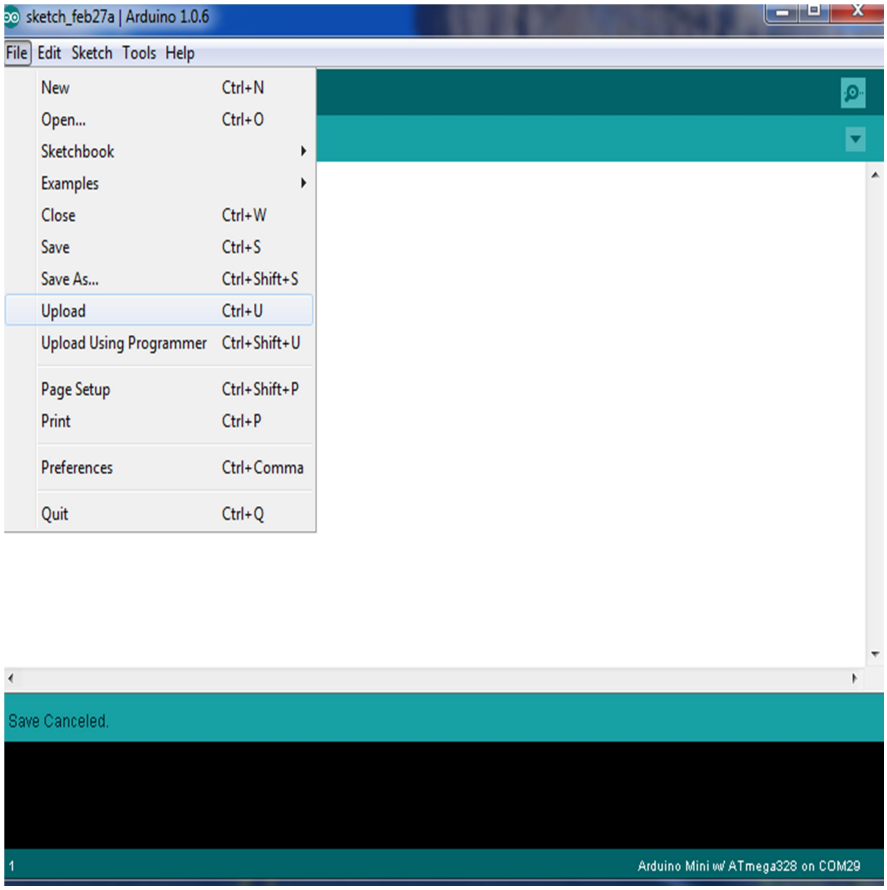
SensorBoard	LCDShield
T	A2
A0	A0
Vcc	5V
GND	GND

- ConnectTemperaturesensorOutputpintoA2pinonArduinoUNOBoardusingasinglepinwire. Also,connectlightsensoroutputpintoA0pinonArduinoUNOBoardusingsinglepinwire,
- ConnectUSBcablefromArduinoUNOboardtoPC.

Note: Beforegettingstarted,checkthattheArduinosoftwareisinstalled.Whenconnectedforthe firsttime,checktheArduinoUNOboardUSBdriversoftwareisupdatedaspertheInstallation document.

Programming:

3. StartArduinoIDEbyselectingtheprograminthewindowsstartmenu.YouwillgettheWindowas below.



4. Stepsforconstructingasimple‘C’programinArduino.

Step1: Create a new sketch [program]

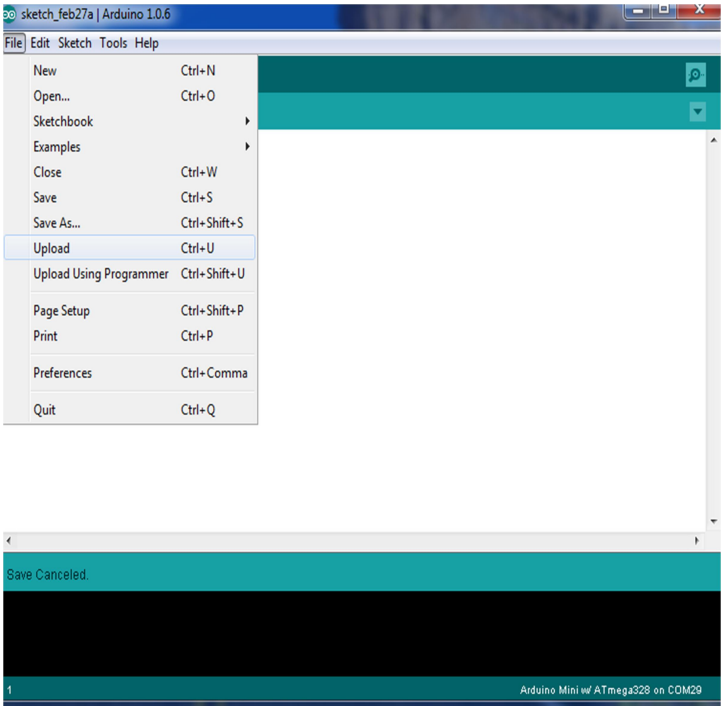
```
int T = A2;      // Power Switch
// A2 has already been defined as Analog Pin2 in Arduino Library
int L = A0;
void setup()
{
/********** Analog Read function*********/
```

```
void Temperature()
{
        int value_temp = analogRead(T);       // reading from A2 pin
        float millivolts_temp = ((value_temp/1023.0)*5000);
        float Celsius = millivolts_temp/10;
        Serial.println(Celsius);
}
void LIG()
{
        int value_lig = analogRead(T);          //reading from A2 pin
        delay(10);
        value_lig = analogRead(L);
        float volts_lig = (value_lig/1023.0)*5;
        // Calculate the Lux = 500/[R1*((Vin - Vsense)/Vsense)]
        int LIGHT = 500/(4*((5-volts_lig)/volts_lig));
        lcd.setCursor(0,1);
        lcd.print("L:");
        lcd.print(LIGHT);
        lcd.print("Lx");
        Serial.print("Light: ");
        Serial.print(LIGHT);
        Serial.println(" Lux");
        delay(2000);
        lcd.clear();
}
        void loop()
{
        Temperature();
        LIG();
        delay(2000);
}
```

Step10:

Now save theproject:Goto**File**menu,youcanfind**save**optionasthewindowsshownbelow Clickon**save**,giveaFilenameandclick**ok**.

NowgototheFilemenu,clickon**Upload**option.Thenthecodeiscompiledandloadedinto Arduino UNOboard.



IfyougetanyissueswiththeuploadingtheremightbeaprobleminttheselectionofCOMportor withthedrivers.RefertotheArdunioenvironmentsetupmanualtomakesurethattherearenо issues.

3.b) Accelerometer & Gyroscope

Accelerometer & Gyroscope Sensor Data Displaying on Output screen Using Raspberry-pi

Practical's Objective:

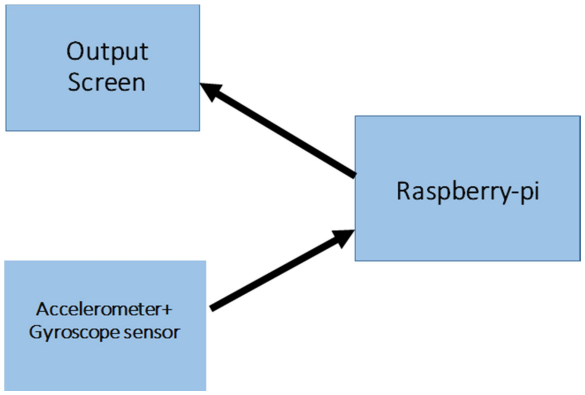
To capture data from sensor and display the same on output screen using raspberry-pi.

Description of Sensor Board Elements:

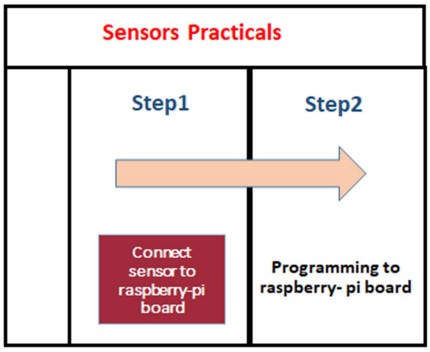
A Accelerometer & Gyroscope sensor (Sensor name: MPU6050) is used to measures force in each of its axis and the angular velocity with respect to the body axis.

Note: As per its names, the technical datasheets of the sensors can be found in the internet for more information.

7. **Sensor Practical Hardware flow Diagram:**



8. **Sensor Practical To Do:**



9. **H/w & S/w Requirements:**

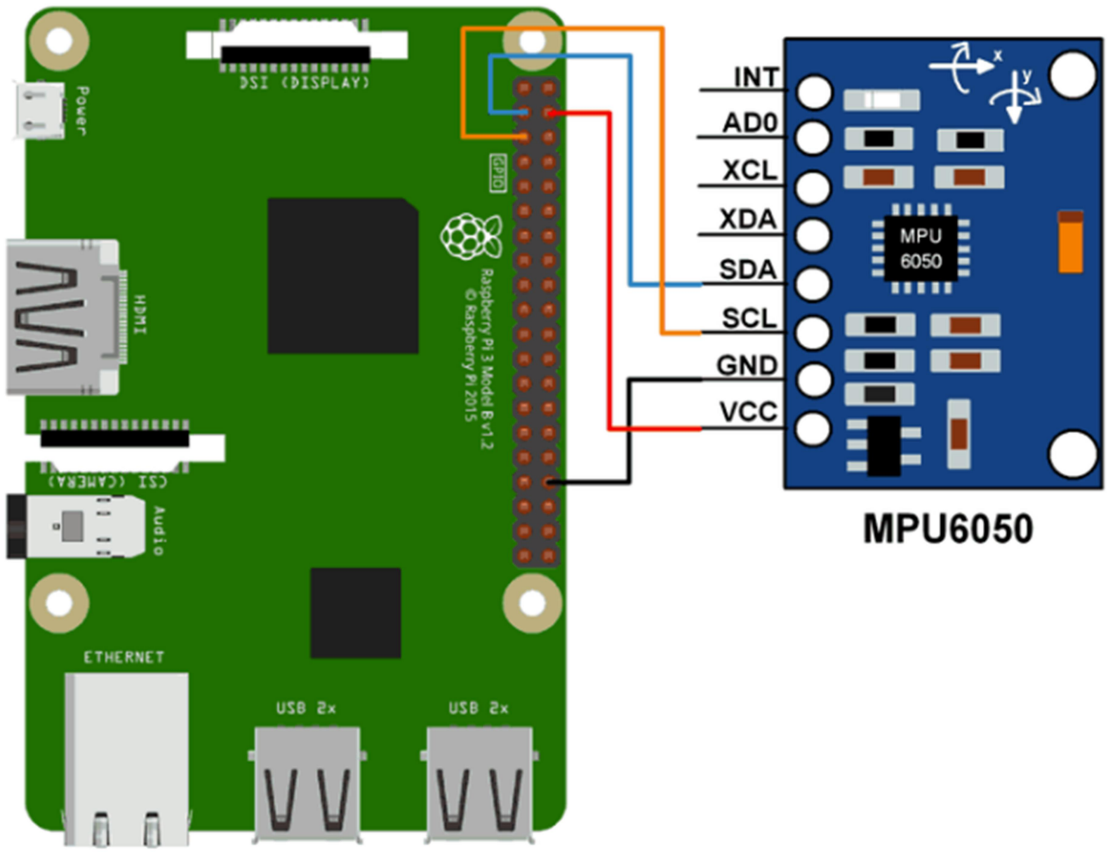
Hardware required:

- Raspberry-pi Board
- 2A charger
- Connectors
- LAN cable
- MPU6050 sensor

10. **Software Requirement:**

Python IDE 3 –

11. **Connecting Sensors to Raspberry-Pi Board:**



MPU6050 Interfacing with Raspberry Pi

12. Programming:

```
import smbus                    #import SMBus module of I2C
from time import sleep          #import


#some MPU6050 Registers and their Address
PWR_MGMT_1   = 0x6B
SMPLRT_DIV   = 0x19
CONFIG       = 0x1A
GYRO_CONFIG  = 0x1B
INT_ENABLE   = 0x38
ACCEL_XOUT_H = 0x3B
ACCEL_YOUT_H = 0x3D
ACCEL_ZOUT_H = 0x3F
GYRO_XOUT_H  = 0x43
GYRO_YOUT_H  = 0x45
```

```
    GYRO_ZOUT_H  = 0x47
def MPU_Init():
        #write to sample rate register
        bus.write_byte_data(Device_Address, SMPLRT_DIV, 7)

        #Write to power management register
        bus.write_byte_data(Device_Address, PWR_MGMT_1, 1)

        #Write to Configuration register
        bus.write_byte_data(Device_Address, CONFIG, 0)

        #Write to Gyro configuration register
        bus.write_byte_data(Device_Address, GYRO_CONFIG, 24)

        #Write to interrupt enable register
        bus.write_byte_data(Device_Address, INT_ENABLE, 1)

    def read_raw_data(addr):
        #Accelero and Gyro value are 16-bit
    high = bus.read_byte_data(Device_Address, addr)
    low = bus.read_byte_data(Device_Address, addr+1)

         #concatenate higher and lower value
    value = ((high << 8) | low)

         #to get signed value from mpu6050
    if(value > 32768):
    value = value - 65536
    return value
    bus = smbus.SMBus(1)    # or bus = smbus.SMBus(0) for older version boards
    Device_Address = 0x68   # MPU6050 device address
    MPU_Init()
```

```
print (" Reading Data of Gyroscope and Accelerometer")
while True:
    #Read Accelerometer raw value
    acc_x = read_raw_data(ACCEL_XOUT_H)
    acc_y = read_raw_data(ACCEL_YOUT_H)
    acc_z = read_raw_data(ACCEL_ZOUT_H)

    #Read Gyroscope raw value
    gyro_x = read_raw_data(GYRO_XOUT_H)
    gyro_y = read_raw_data(GYRO_YOUT_H)
    gyro_z = read_raw_data(GYRO_ZOUT_H)

    #Full scale range +/- 250 degree/C as per sensitivity scale factor
    Ax = acc_x/16384.0
    Ay = acc_y/16384.0
    Az = acc_z/16384.0

    Gx = gyro_x/131.0
    Gy = gyro_y/131.0
    Gz = gyro_z/131.0

    print ("Gx=%.2f" %Gx, u'\u00b0'+ "/s", "\tGy=%.2f" %Gy, u'\u00b0'+ "/s", "\tGz=%.2f" %Gz, u'\u00b0'+ "/s", "\tAx=%.2f g" %Ax, "\tAy=%.2f g" %Ay, "\tAz=%.2f g" %Az)
    sleep(1)
```

3.c) Actuators Ex: Motors

Stepper motor using Raspberry-Pi

Practical's Objective:

To show output of stepper motor using raspberry-pi.

Description of Sensor Board Elements:

Stepper motors are DC **motors** that move in discrete steps.

13. Sensor Practical Hardware flow Diagram:

14. H/w & S/w Requirements:

Hardware required:

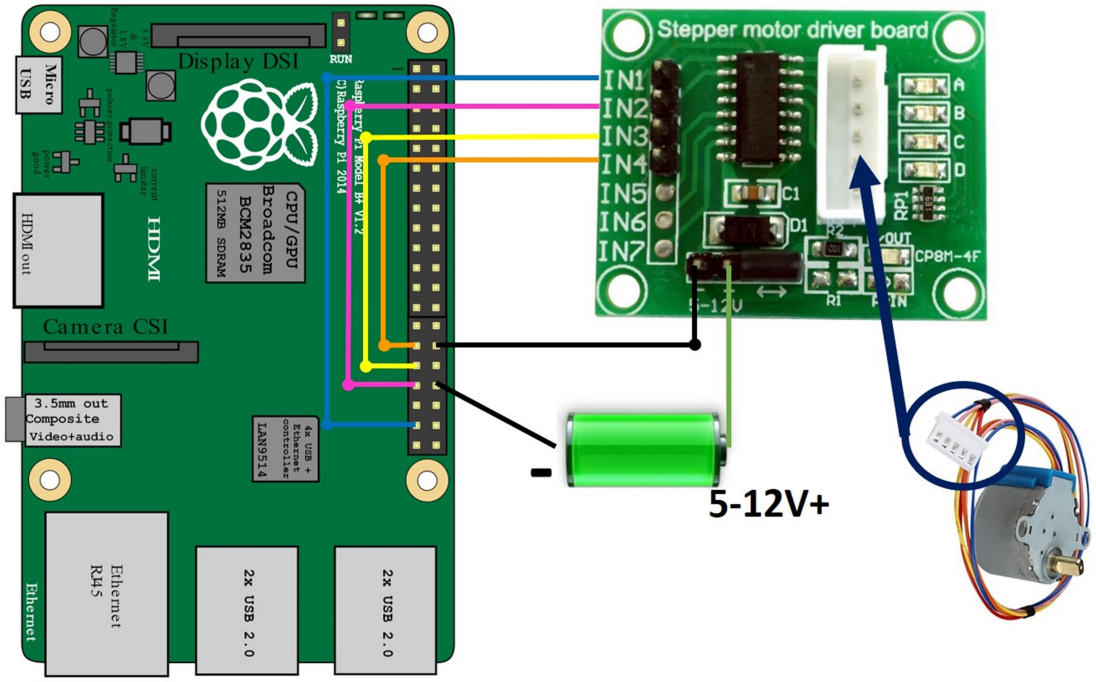
- Raspberry-pi Board
- 2A charger
- Connectors
- LAN cable
- Stepper motor with Driver ic

15. Software Requirement:

Python IDE 3 –

16. Connecting Sensors to Raspbery-pi Board:

Stepper Motor	I2c Shield
5V	5v(pin 2)
GND	GND(pin 6)
IN1	GPIO27(PIN 13)
IN2	GPIO22(PIN 15)
IN3	GPIO9(PIN 19)
IN4	GPIO10(PIN 21)



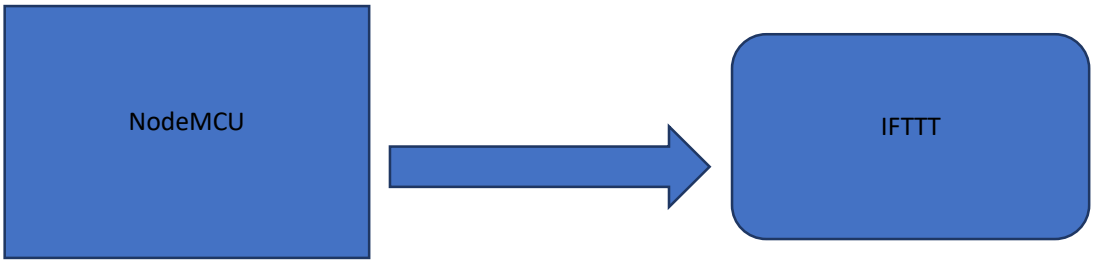
17. Programming:

```
import time
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
StepPins = [27,22,10,9]
for pin in StepPins:
GPIO.setup( pin, GPIO.OUT )
GPIO.output( pin, False )
      Seq = [[1,0,0,0],
            [1,1,0,0],
            [0,1,0,0],
            [0,1,1,0],
            [0,0,1,0],
            [0,0,1,1],
            [0,0,0,1],
            [1,0,0,1]]
     for j in range(512):
     for half in range(8):
     for pin in range(4):
     GPIO.output(StepPins[pin],Seq[half][pin])
     time.sleep(0.001)
```

Week 3:

Introduction to loT with The Thing Box & IFTTT

NodeMCU with IFTTT:



Hardware & Software Required:

Hardware required:

- NodeMCU

Software required:

- Arduino IDE

Programming:

```
#include<ESP8266WiFi.h>
                                    // Include the ESP8266 WiFiSecure library:
                                    #include <WiFiClientSecure.h>


                                    ///////////////////////////////
                                    // WiFi Network Definitions //
                                    ///////////////////////////////
                                    // Replace these two character strings with the name and
                                    // password of your WiFi network.
                                    const char mySSID[] = "YourWiFiName";
                                    const char myPSK[] = "YourWifiPassword";


                                    //////////////////////
                                    // IFTTT Constants //
                                    //////////////////////
                                    // IFTTT destination server:
                                    const char* IFTTTServer = "maker.ifttt.com";
```

```
// IFTTT https por:
const int httpsPort = 443;
// IFTTT Event:
const String MakerIFTTT_Event = "button";
// IFTTT private key:
const String MakerIFTTT_Key = "YourPrivateKeyIFTTT";


String httpHeader = "POST
/trigger/"+MakerIFTTT_Event+"/with/key/"+MakerIFTTT_Key +"
HTTP/1.1\r\n" +
                    "Host: " + IFTTTServer + "\r\n" +
                    "Content-Type: application/x-www-form-urlencoded\r\n\r\n";


void setup()
{
 int status;
Serial.begin(9600);
Serial.println();
Serial.print(F("connecting to "));
Serial.println(mySSID);
WiFi.begin(mySSID, myPSK);
 while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
 }
Serial.println("");
Serial.println(F("WiFi connected"));
Serial.println(F("IP address is: "));
Serial.println(WiFi.localIP());
Serial.println(F("Press any key to post to IFTTT!"));
}


void loop()
{
 // If a character has been received over serial:
 if (Serial.available())
 {
  // !!! Make sure we haven't posted recently
  // Post to IFTTT!
postToIFTTT();
  // Then clear the serial buffer:
  while (Serial.available())
Serial.read();
 }
}
```

```
void postToIFTTT()
{
  // Create a client, and initiate a connection
WiFiClientSecure client;
  if (client.connect(IFTTTServer, httpsPort) <= 0)
  {
Serial.println(F("Failed to connect to server."));
    return;
  }
Serial.println(F("Connected."));


Serial.println(F("Posting to IFTT Event!"));
client.print(httpHeader);


  // available() will return the number of characters
  // currently in the receive buffer.
  while (client.available())
Serial.write(client.read()); // read() gets the FIFO char
  // connected() is a boolean return value - 1 if the
  // connection is active, 0 if it's closed.
  if (client.connected())
client.stop(); // stop() closes a TCP connection.
}
```

Week 4 & 5:

Build your own Raspberry Pi Web Server

Setting Webserver On Raspberry Pi

Practical Steps to be followed:

- Install Apache :

```
sudo apt-get update
```

- Then install the apache2 package with this command

```
sudo apt-get install apache2 -y
```

- Test the web server:

By default, Apache puts a test HTML file in the web folder. This default web page is served when you browse to `http://localhost/` on the Pi itself, or `http://192.168.1.10` (whatever the Pi's IP address is) from another computer on the network.

- To find the Pi's IP address, type `hostname -I` at the command line (or read more about finding your IP address).
- The default web page is just an HTML file on the filesystem. It is located at `/var/www/html/index.html`
- Navigate to this directory in a terminal window and have a look at what's inside:

```
cd /var/www/html
ls -al
```

- This shows that by default there is one file in `/var/www/html/` called `index.html` and it is owned by the `root` user (as is the enclosing folder). In order to edit the file, you need to change its ownership to your own username. Change the owner of the file (the default `pi` user is assumed here) using `sudochown pi: index.html` .

Your Own Website:

If you know html you can put your own html files in this directory and serve them as a website on your local network.

Week 6:

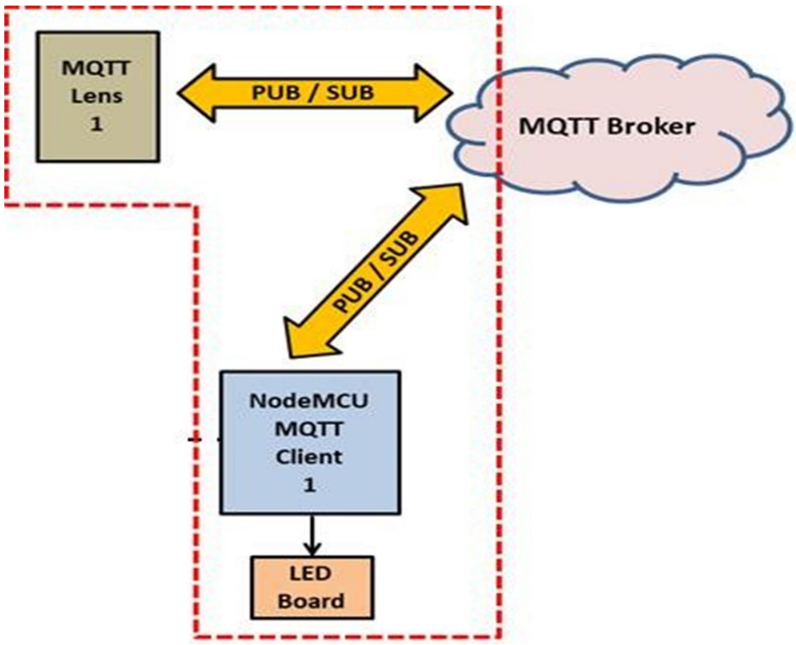
Build a Web-App: Blinking an LED over Internet

Subscribe to actuateLED–On & Off

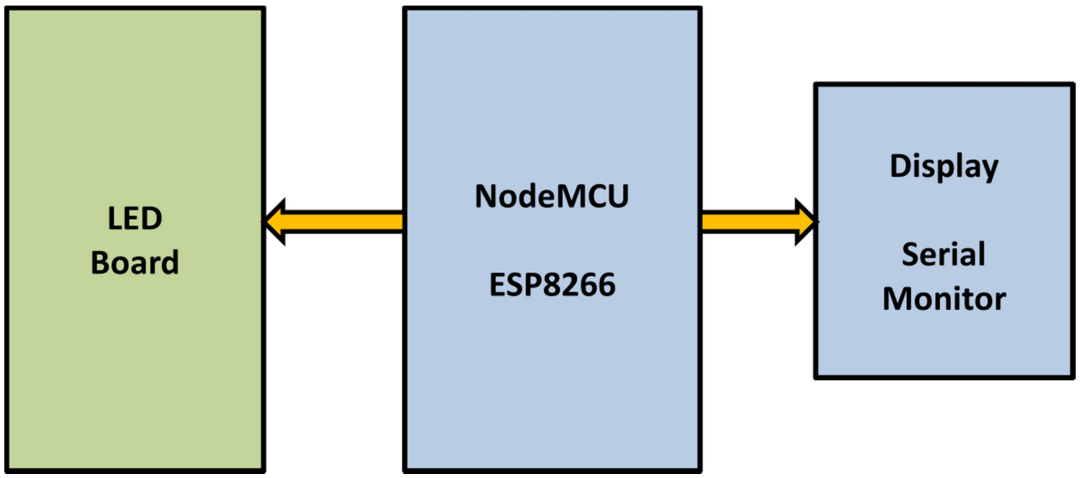
Practical's Objective:

SubscribeforthedatasentbyMQTTLensanddependingondatareceived,actuatetheLEDs

End-End MQTT Flow diagram:



Practical Hardware flow Diagram



H/WandS/Wrequirements:

- **HardwareRequirement:**
  - NodeMCU (with base)Board
  - Type-D USBcable
  - 5V DCAdaptor
  - µ USBcable
  - LEDs
  - 1-pin F-F connectingwire

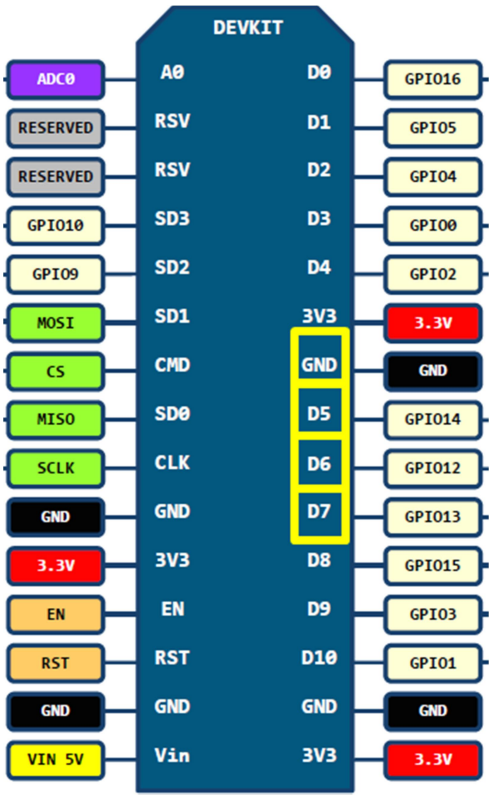
- **SoftwareRequirement:**
  - Arduino(IDE)1.6.9onwards.Clickbelowforlatestsoftwarerelease.https://

www.arduino.cc/en/Main/OldSoftwareReleases

2. HardwareConnectivity:

➢ ConnectingNodeMCUtothebase

- ImagebelowrepresentsthepinnumbersoftheNodeMCU



- BelowaretheimagesofNodeMCUs.

➢ **ConnectingLEDboardtoNodeMCUboard:**

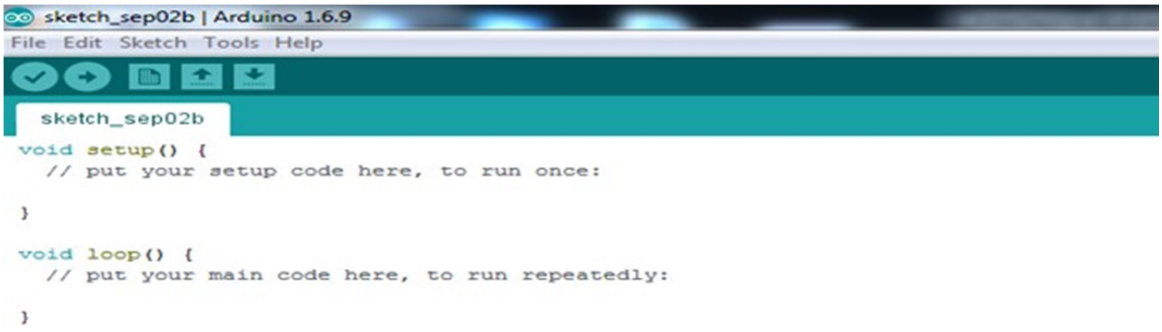
LED	GPIO	NodeMCU
1 - Red	13	D7
2 - White	12	D6
3 - Blue	14	D5
GN D		GND

- ConnectRed,White,BlueLEDpinstoD5,D6,D7respectivelyonNodeMCUboardas mentionedintheabovetable.
- ConnectGNDpinsonLEDtoGNDonNodeMCU
- EnsurethepowersupplyisgiventobothNodeMCU.

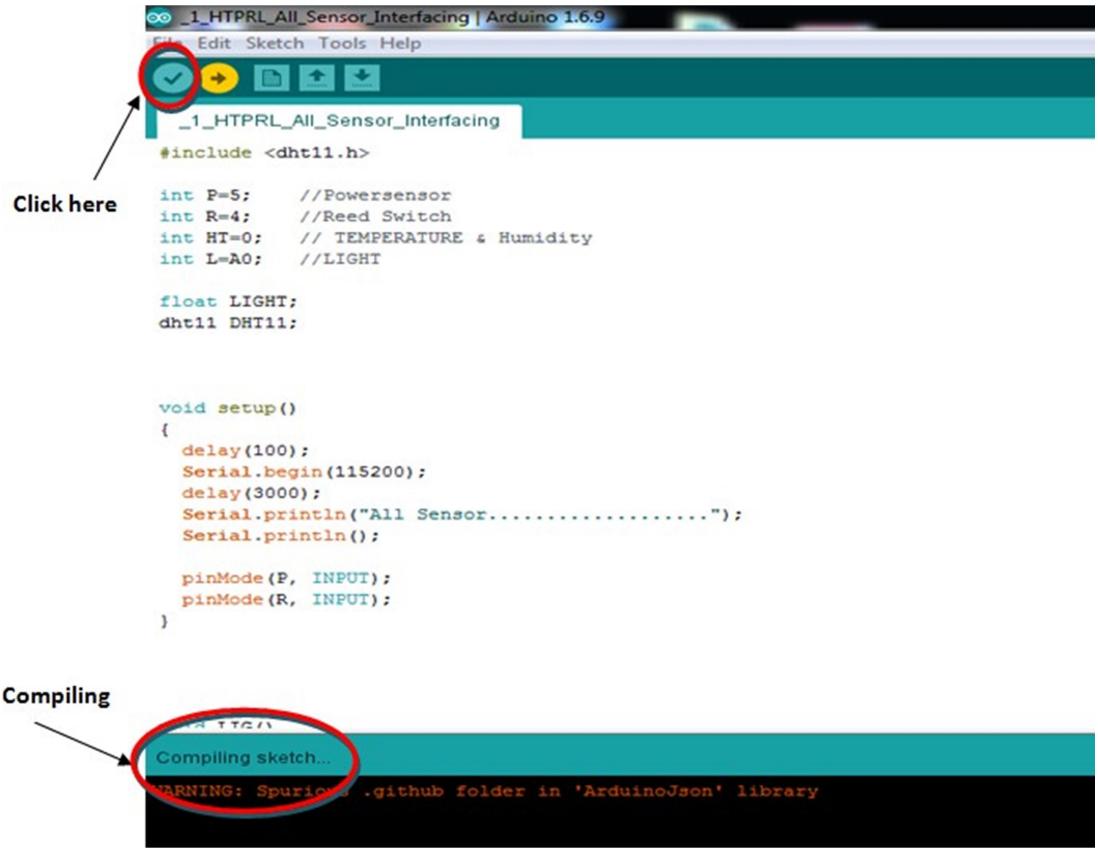
Programming the NodeMCU:

- **Compile/Save/UploadtheProjectcode:**
  - Youcanuseeitherofthetwooptions(option1**or**option2)belowfortheproject code.
  - Intheарduinocode,edityourownwifiSSIDandpasswordfields.

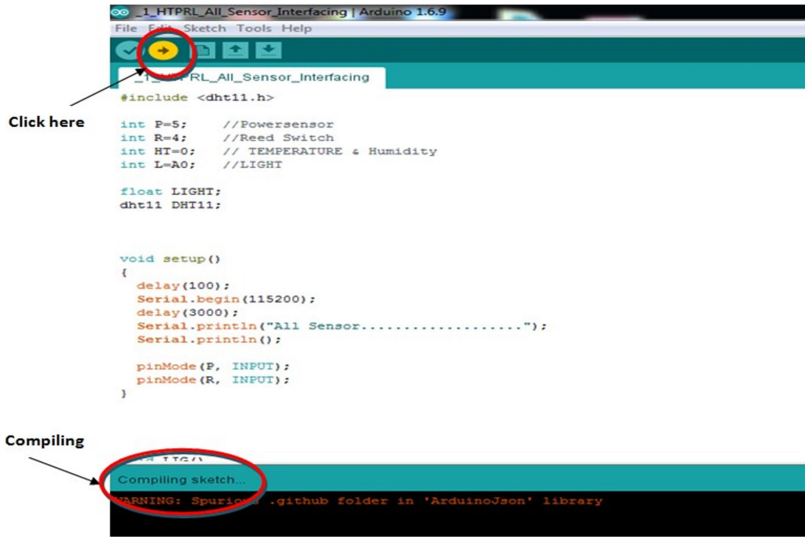
  - StarttheArduinoIDEeitherfromstartmenuordesktopicon.Youwillbeseena windowasshownbeloworanyotherpreviouslyopenedsketch.Copythecodefromthe and paste in the Arduino sketch (overwrite the existing text or code in the sketch ifany).
  - Nowsavetheproject:Goto**File**menu,youcanfind**save**option.Clickon**save**, giveaFilenameandclick**ok**.



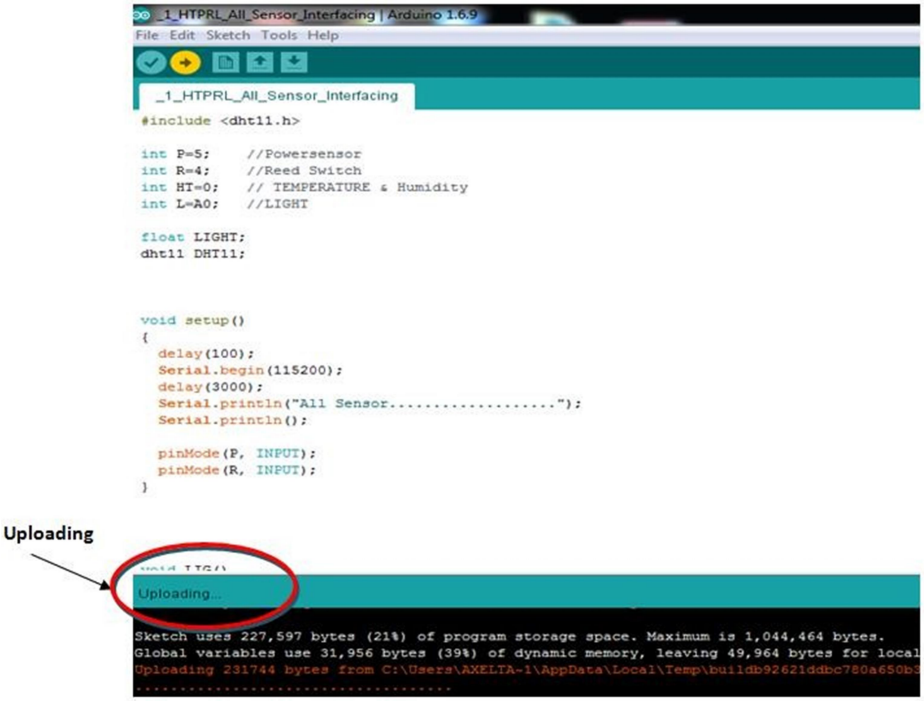
- Now,clickonthe‘Verify’buttonorgoto**sketch->Verify**.Thiswillcompilethecode as shownbelow:



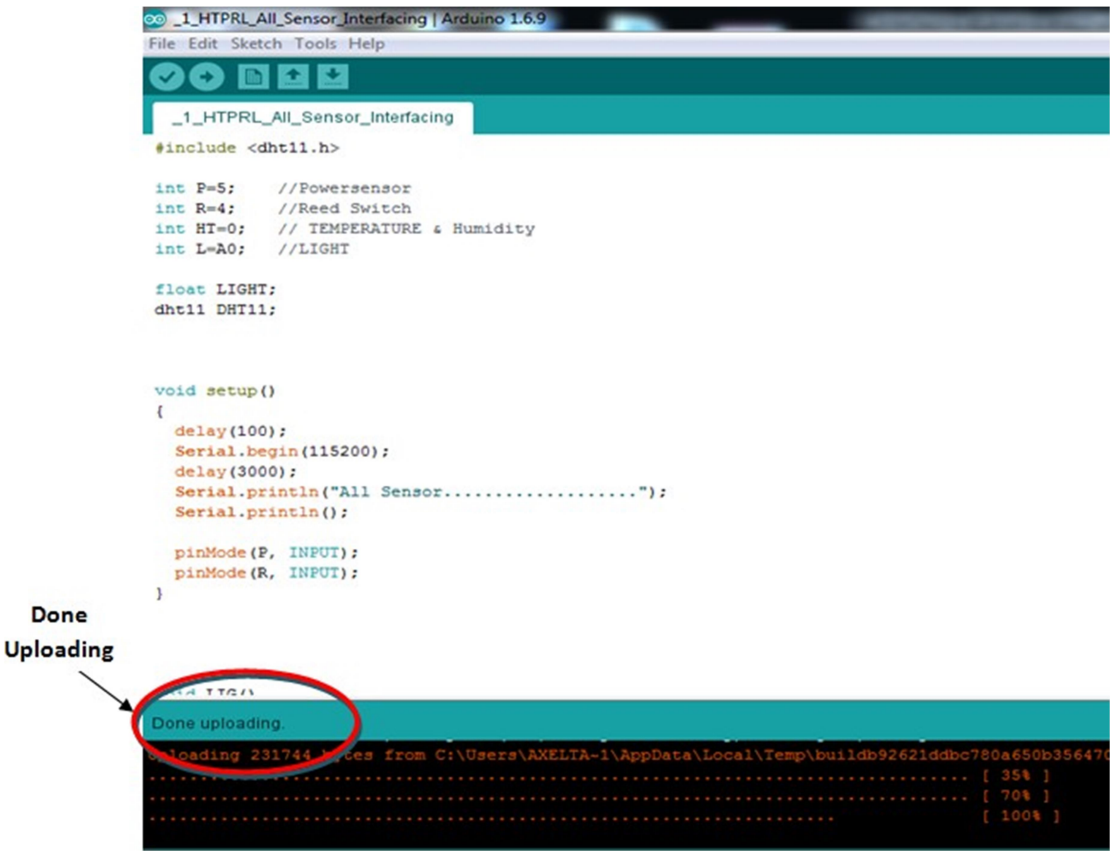
- NowgotoTools>Port>COM#SelectProperCOMPort,sameaswhichisdetectedunder devicemanager.
- Now,clickonthe‘Upload’buttonorgoto**sketch->Upload**.Thiswillcompilethe codeandthenuploadsittotheNodeMCUasshowninthebelowimages.


Click here
Compiling
_1_HTPRL_All_Sensor_Interfacing | Arduino 1.6.9
File Edit Sketch Tools Help
_1_HTPRL_All_Sensor_Interfacing
#include <dht11.h>
int P=5; //Powersensor
int R=4; //Reed Switch
int HT=0; // TEMPERATURE & Humidity
int L=A0; //LIGHT
float LIGHT;
dht11 DHT11;
void setup()
{
delay(100);
Serial.begin(115200);
delay(3000);
Serial.println("All Sensor.....");
Serial.println();
pinMode(P, INPUT);
pinMode(R, INPUT);
}
Compiling sketch...
.github folder in 'ArduinoJson' library

- Youcanobservethatthecodeisuploadingasshownintheimagebelow.

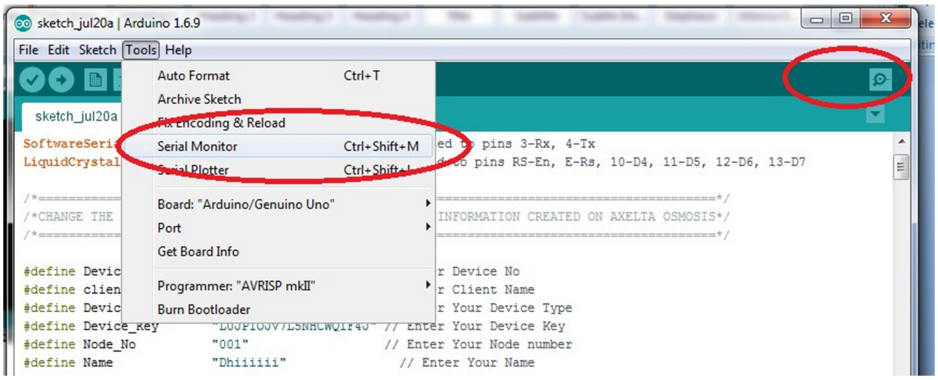


- Finallyyoucanobservethatthecodeisuploaded.

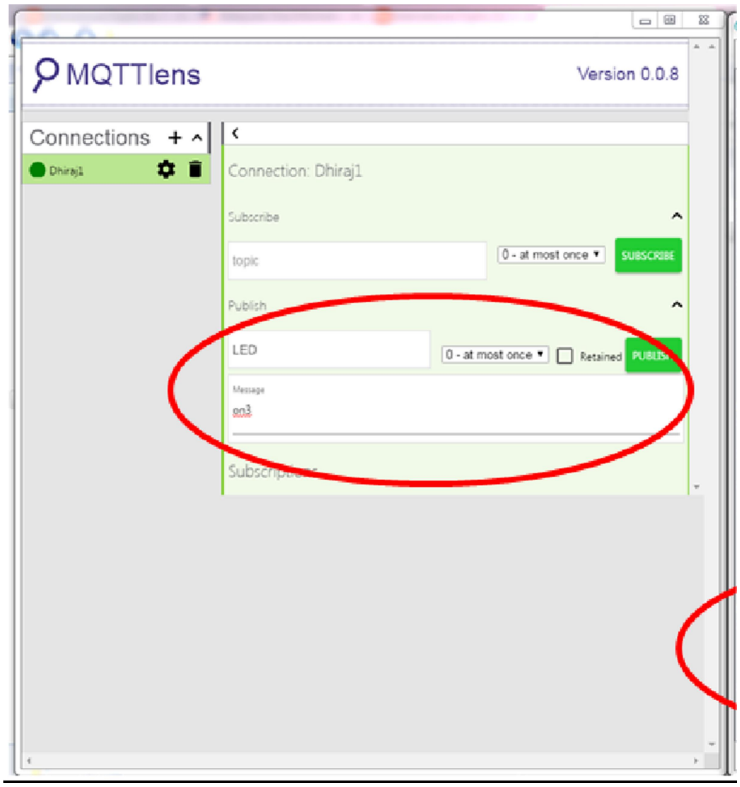

_1_HTPRL_All_Sensor_Interfacing | Arduino 1.6.9
File Edit Sketch Tools Help
_1_HTPRL_All_Sensor_Interfacing
#include <dht11.h>
int P=5; //Powersensor
int R=4; //Reed Switch
int HT=0; // TEMPERATURE & Humidity
int L=A0; //LIGHT
float LIGHT;
dht11 DHT11;
void setup()
{
delay(100);
Serial.begin(115200);
delay(3000);
Serial.println("All Sensor.....");
Serial.println();
pinMode(P, INPUT);
pinMode(R, INPUT);
}
Done
Uploading
Done uploading.
[35%]
[70%]
[100%]

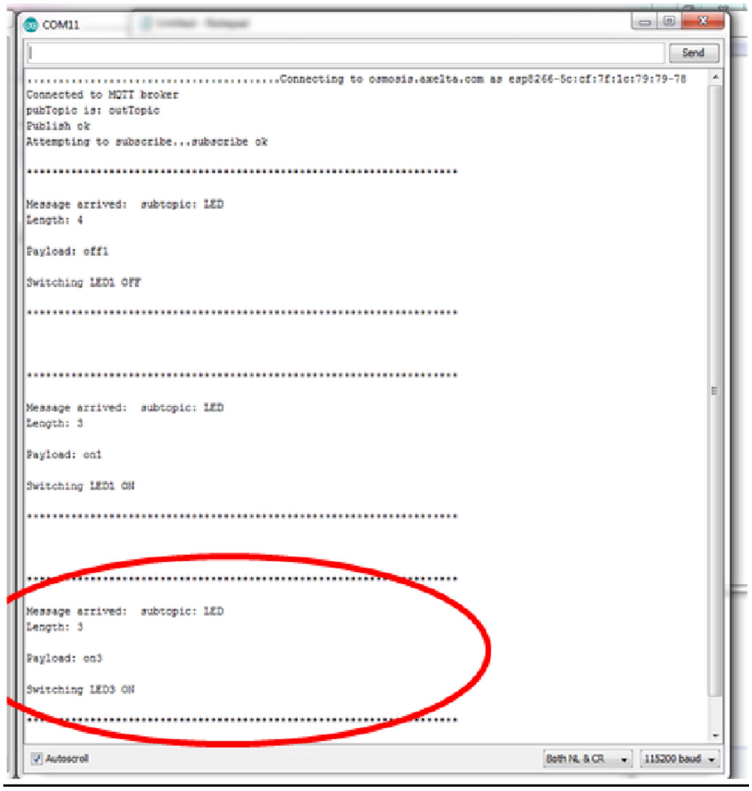
Output:

- Toseetheoutput,goto**Tools->SerialMonitor**orClickMagnifyingglasssymbolattop rightcorner.
- Itwillopenanewwindowcalledserialmonitor.Selecttheshowcasedoptionandbaud rateto**115200**atrightbottomsideasshownbelow.

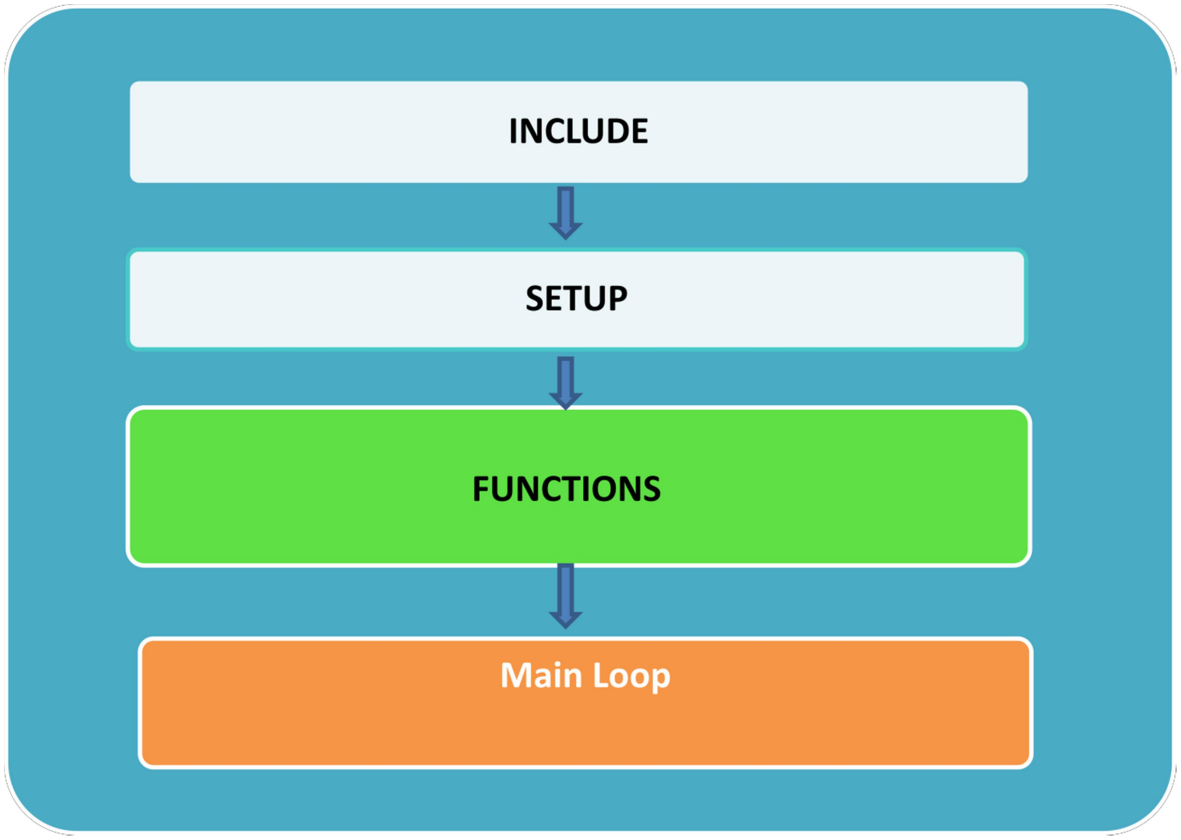


- OpenMQTTLensandenterPublishTopic‘LED’withmessageandclickon‘Publish’.
- DependingonmessagereceivedNodeMCUwillactuateLEDeitherOnorOffandthe statuscanbeviewedontheserialmonitorasshown.
- MessagestobesenttoactuateLEDsare:“ON1”,“ON2”,“ON3”,“OFF1”,“OFF2”, “OFF3”


MQTTlens
Version 0.0.8
Connections
Dhiraj1
Connection: Dhiraj1
Subscribe
topic
0 - at most once
SUBSCRIBE
Publish
LED
0 - at most once
Retained
PUBLISH
Message
on3


COM11
Send
Connecting to osmosis.axelta.com as esp8266-5c:cf:7f:1c:79:79-78
Connected to MQTT broker
pubTopic is: outTopic
Publish ok
Attempting to subscribe...subscribe ok
Message arrived: subtopic: LED
Length: 4
Payload: off1
Switching LED1 OFF
Message arrived: subtopic: LED
Length: 3
Payload: on1
Switching LED1 ON
Message arrived: subtopic: LED
Length: 3
Payload: on3
Switching LED3 ON
Autoscroll
Both NL & CR
115200 baud

3. **Understandingthecodeflow:**



- Theentirecodecanbedividedintodifferentsectionsasshownintheaboveimage. Briefexplanationisprovidedbelow:
  a. **INCLUDE**section**:**Containsallthenecessary*headerfiles,globalvariables,objects, functionprototypedeclarations*beingusedinthecode.
  b. **SETUP**section**:**Systemfunctionthatcontainsallthenecessarysettingsrequiredlike-*baudrate*(rateatwhichserialcommunicationtakesplaceintermsofbits/sec), *input & output pin configurations, wifi connectivity, generate client name and connectingtoMQTTbroker*.
  c. **FUNCTIONS**section**:**Containsthedefinitionsofvarioususerdefinedfunctionslike *callback(),macToStr(),reconnect(),find_string(),find_char_loc()* used in the code.
  d. **MAINLOOP**section**:**Systemfunctionthatisexecutedfirstandcallsthe*reconnect()* and*client.loop()*functionstocheckttheclientconnectionandprocessincoming messagesandmaintainitsconnectiontotheserverrespectively.
- Furtherexplanationofthecodeisprovidedascommentswhereevernecessry:

a) **INCLUDE**section:

```
/*wifi client library from esp8266 - contains all necessary APIs related to wifi*/
#include <ESP8266WiFi.h>
/*MQTT client library from knolleary.net - contains all necessary APIs related to MQTT*/
#include <PubSubClient.h>

//**********************************************//

const char*ssid="Axelta";/*DefneyourownwifiSSID*/
const char* password = "Axelta140218"; /*Defne your own wifi password*/

char* server = "osmosis.axelta.com"; /*Axelta MQTT broker-server host name*/
//IPAddressserver(54, 201, 150, 33); /*Axelta MQTT broker-server IP address*/
//IPAddressserver(192,168,1,246);/*ifyouhadanyloaclMQTTbrokerenterIPaddressofthatbrokermachine*/

char* pubtopic = "outTopic"; /*MQTT Publish topic*/
char* subtopic = "LED"; /*MQTT Subscribe topic*/

//**********************************************//

constintredLED=13;/*definedigitalpinforLEDboard*/
constintwhiteLED=14;/*definedigitalpinforLEDboard*/
constintblueLED=12;/*definedigitalpinforLEDboard*/

charmessage_buff[100];/*charecterarrayforstoringincomingmsgduringsubscription*/ String clientName1; /*Define string variable for clientname*/

WiFiClientwifiClient; /*declaring WiFiClient object variable*/
voidcallback(char*subtopic,byte*payload,unsignedintlength);/*Thisfunctioniscalledwhenmessagearrivedforsubscribedtopic*/
PubSubClientclient(server, 1883, callback, wifiClient); /*Necessary parameters for MQTT clients*/
```

b) SETUPsection:

```
/*Systemfunctionthatcontainsallthenecessarysettingsrequiredlike-
baudrate,input&outputpinconfigurations, wificonnectivity*/

voidsetup()

{

 delay(1000);
 Serial.begin(11520
 0); delay(5000);

 Serial.println("SubscribeLEDono
 ff"); Serial.println();


 /*DefineGPIOasaninputoroutpu
 t*/ pinMode(redLED,
 OUTPUT);
 digitalWrite(redLED, 0);
 pinMode(whiteLED,
 OUTPUT);
 digitalWrite(whiteLED, 0);
 pinMode(blueLED, OUTPUT);
 digitalWrite(blueLED,0);


 /*ConnecttolocalwifinetworkusingSSID&ps
 wd*/ Serial.println("Wifi disconnecting... ");
 WiFi.disconnect();

 delay(1000);
 Serial.print("Connecting to
 ");Serial.println(ssid);
 WiFi.begin(ssid, password);
 int dot=0;


/*Waittillwifinetworkgetsconnected*/

 while(WiFi.status()  !=WL_CONNECTED)

 {

  delay(500);
  Serial.print(".");
  Serial.print(dot);
  dot++;
```

```
 if(dot==20||dot==40||dot==60||dot==80||dot==100){Serial.println();}
 if(dot>120){Serial.println();Serial.println("Error Connecting Wifi Network");break;}
}
Serial.println(
);
Serial.println(
);
delay(1000)
;
/*GenerateclientnamebasedonMACaddressandlast8bitsofmicrosecondcounter*/
StringclientName;
clientName += "esp8266-
"; uint8_t mac[6];
WiFi.macAddress(mac);
clientName +=
macToStr(mac);
clientName +="-";
clientName+=String(micros()&0xff,
16); clientName1 =clientName;
delay(1000);


Serial.print("Connecting
to"); Serial.print(server);
Serial.print(" as ");
Serial.println(clientName
1); delay(1000);


/*Connect to MQTT broker */
if(client.connect((char*) clientName1.c_str()))
{
 Serial.println("Connected to MQTT
 broker"); Serial.print("pubTopic is: ");
 Serial.println(pubtopic);
 delay(1000);
 if(client.publish(pubtopic, "hello from ESP8266")) /*publish hello:test message*/
 {
  Serial.println("Publish ok");
 }
else
```

```
{
 Serial.println("Publish failed");
}
delay(1000);

Serial.print("Attempting to subscribe...");
if(client.subscribe(subtopic)) /*subscribe to a subtopic */
{
 Serial.println("subscribe ok");
}
else
{
 Serial.println("subscribe failed");
}
}
else
{
 Serial.println("MQTT connect
 failed"); Serial.println("Will reset
 and try again...");
 abort(); /*abort the program execution and comes out directly from the place of the call.*/
}
delay(1000);
}
```

c) FUNCTIONSsection:

```
/*To notify the upper application of certain activities occurring in the MQTT-client core.*/
voidcallback(char* subtopic, byte* payload, unsigned int length)
{
inti = 0;
Serial.printl
n();
Serial.println("**********************************
**********************************");
Serial.println();
```

```
Serial.println("Messagearrived:subtop
ic:"+String(subtopic));
Serial.println("Length:"+String(len
gth,DEC));

Serial.println();

for(i=0; i<length; i++)

{

 message_buff[i] = payload[i];

}

message_buff[i] = '\0';

String msgString = String(message_buff);

Serial.println("Payload:"+msgString);/*Printallreceivedmsgcontents
forsubscribedtopics*/ Serial.println();


/*DependingonreceivedmsgactuaterespectiveLEDeitheron/of
f*/ msgString.toLowerCase();

 if(find_string(msgString,"on1"))

 {

  Serial.println("Switch
  ing LED1 ON");
  digitalWrite(redLED,
  HIGH); delay(10);

 }

 else if (find_string(msgString,"on2"))

 {

  Serial.println("Switch
  ing LED2 ON");
  digitalWrite(whiteLE
  D, HIGH); delay(10);

 }

 else if (find_string(msgString,"on3"))

 {

  Serial.println("Switch
  ing LED3 ON");
  digitalWrite(blueLED
  , HIGH); delay(10);

 }

 else if (find_string(msgString,"off1"))
```

```
  {
   Serial.println("Switchin
   g LED1 OFF");
   digitalWrite(redLED,
   LOW); delay(10);
  }
  else if (find_string(msgString,"off2"))
  {
   Serial.println("Switchin
   g LED2 OFF");
   digitalWrite(whiteLED,
   LOW); delay(10);
  }
  else if (find_string(msgString,"off3"))
  {
   Serial.println("Switchin
   g LED3 OFF");
   digitalWrite(blueLED,
   LOW); delay(10);
  }
else
  {
   Serial.println("Command NOT
   Supported"); delay(10);
  }
  Serial.println();
  Serial.println("**********************************
  **********************************");
  Serial.println();
  Serial.println();
 }

 /*CreateuniqueclientnameusingMACaddressofNode
 MCUboard*/ StringmacToStr(constuint8_t*mac)
```

```
{
 String result;
 for(int i = 0; i< 6; ++i)
 {
  result += String(mac[i], 16);
  if(i<
   5)
   result
   +=':';
 }
 returnresult;
}
/*Reconnect function definition with defined pub & sub topics*/
voidreconnect()
{
 while(!client.connected())
 {
  Serial.print("AttemptingMQTTconnection...");

  if  (client.connect((char*)clientName1.c_str()))
  {
   Serial.println("connected");
   if(client.publish(pubtopic,"h
   ellofromESP8266"))
   Serial.println("Publishok");
   else
   Serial.println("Publi
   sh failed");
   delay(1000);
   Serial.print("Attempting to subscribe...");
   if(client.subscribe(subtopic))

   Serial.println("subscribe ok");
```

```
  else
   Serial.println("subscribe failed");
  }
else
  {
   Serial.print("failed,
   rc=");
   Serial.print(client.st
   ate());
   Serial.println("tryagai
   nin5seconds");
   delay(5000);
  }
 }
}
/*function definition for
find_string*/
booleanfind_string(Stringbase
,Stringsearch)
{
intlen = search.length(); // find the length of the base string
for(intm=0;m<((base.length()-
len)+1);m++)//Iteratefromthebeginningofthebasestringtilltheendminuslen
gth of thesubstring
{
if(base.substring(m,(m+len))==search) // Check if the extracted
Substring Matches the Search String
{
returntrue; // if it matches exit the function with a true value
}
}
returnfalse; // if the above loop did not find any matches, control
would come here and return a false value
}
```

```
/*function definition for find_char*/
intfind_char_loc(String base, char search)
{
 for(intm=0;m<base.length();m++)//Iteratefromthebeginningofthebasestringtilltheendminuslengthofthe substring
 {
  if(base[m]==search) // Check if the character Matches the Search character
  {
   returnm; // if it matches exit the function with the current location value
  }
 }
}
```

d) MAIN LOOPsection:

```
/*Main function from where the execution begins */
voidloop()
{
 if(!client.connected()) /*if client not connected, call reconnect function*/
 {
  reconnect();
 }
 client.loop();/*Thisfunctionshouldbecalledregularlytoallowtheclienttoprocessincomingmessagesandmaintainitsconnectiontotheserver.*/
}
```

Week 7&8:

Live Temperature and Humidity Monitoring over Internet

Monitor Temperature and Humidity Sensor Data through MQTT Lens

Practical's Objective:

Tocapturedatafromdifferentsensorsanddisplaythesameon SerialMonitorusingNodeMCU andpublishallsensordatatoMQTTBrokerandsubscribethroughLens.

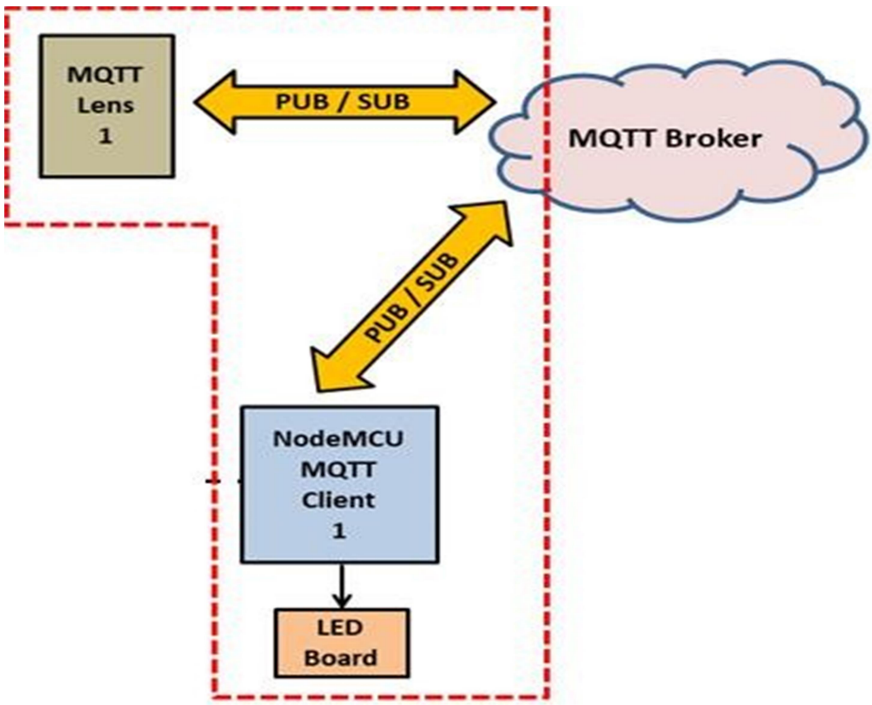
Description of Sensor Board Elements:

A**Temperaturesensor**(Sensorname:DHT11)isusedtomonitorthetemperatureofthe environment.

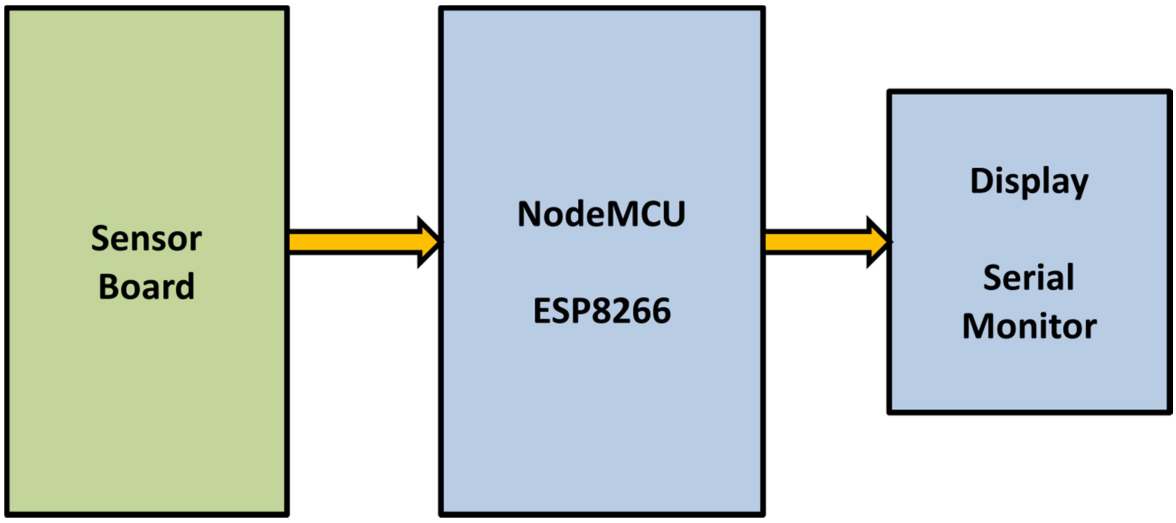
A**Humiditysensor**(Sensorname:DHT11)isusedtomonitorthehumiditylevelsofenvironment.

Note:Asperitsnames,thetechnicaldatasheetsofthesensorscanbefoundintheinternetfor moreinformation.

End-End IoT Flow diagram:



Practical Hardware flow Diagram



H/W and S/W requirements:

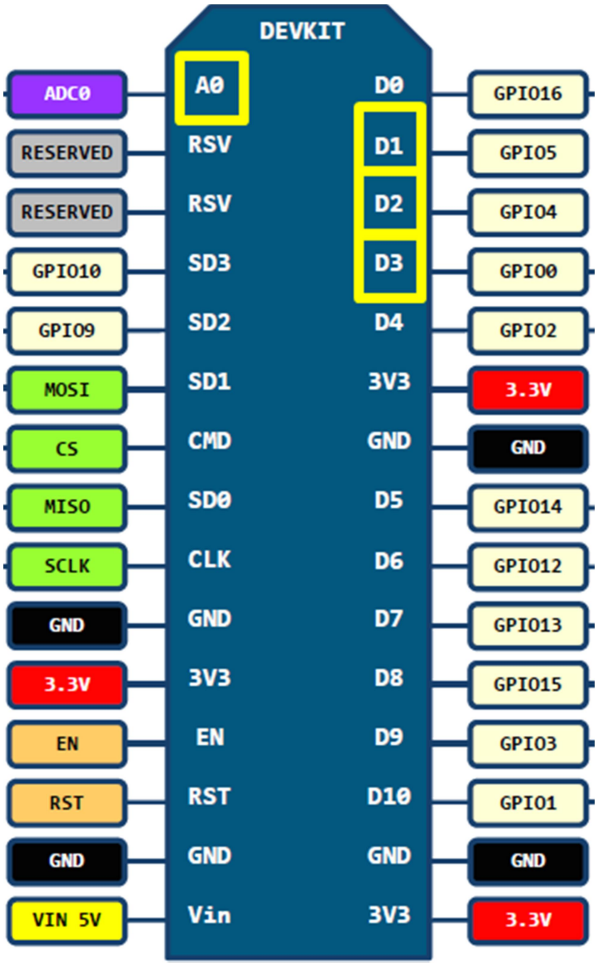
- **HardwareRequirement:**
  - NodeMCU (with base)Board
  - Temperaturesensor
  - Humiditysensor
  - 2-pin F-F connectingwire
  - 5V DCAdaptor
  - µ USBcable
  - LEDboard
  - 1-pin F-F connectingwire

- **SoftwareRequirement:**
  - Arduino(IDE)1.6.9onwards.Clickbelowforlatestsoftwarerelease.https://

www.arduino.cc/en/Main/OldSoftwareReleases

1. **HardwareConnectivity:**

- ImagebelowrepresentsthepinnumbersoftheNodeMCU



- BelowaretheimagesofNodeMCUs.


CP2102
USB to
Serial


CH340G
USB to
Serial


NODEMCU
Amica

➢ **ConnectingSensorboardtoNodeMCUboard:**

- **Sensornotationsonthesensorboard:**

  H – Temperature and Humidity Sensor : Digital sensor

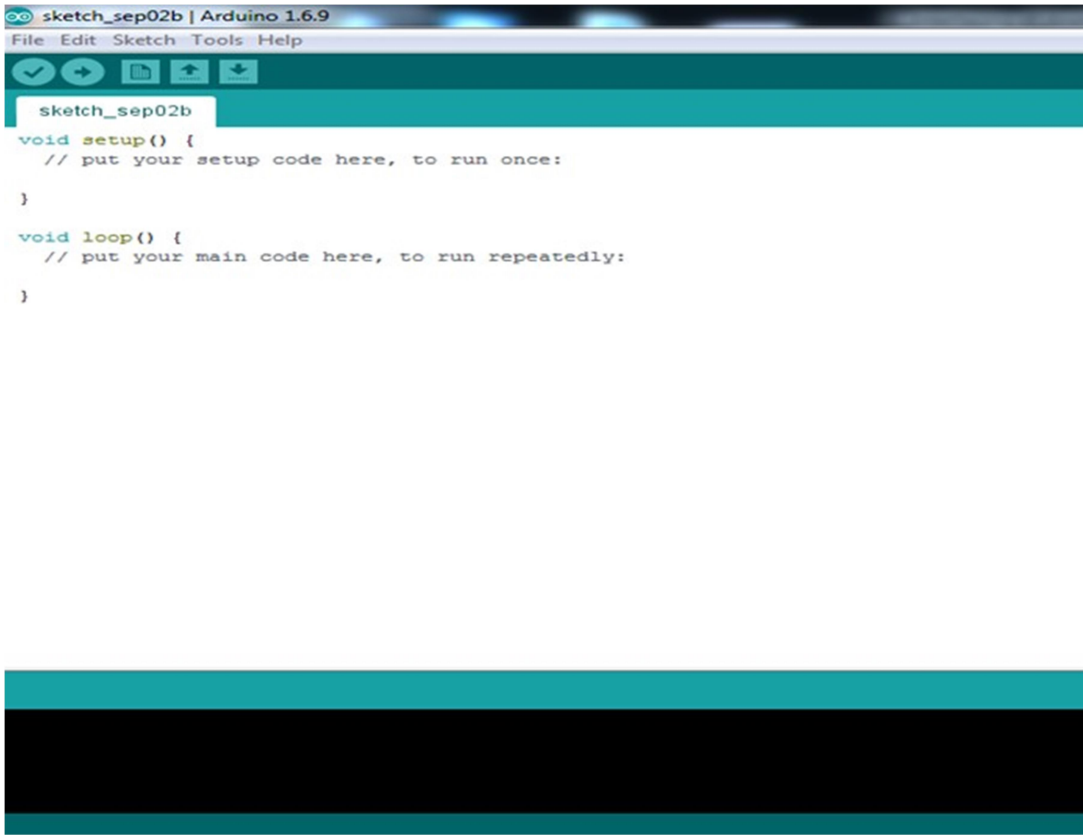
Sensor	Sensor Pins	GPIO	NodeMCU
TemperatureandHumidity	**H**	**0**	**D3**

- ConnecttheDHT11outputpin(namedas**L**onsensorboard)toD3onNodeMCU board as mentioned in the above table. Connect the Vcc of sensor to Vcc of NodeMCUandGNDtoGNDofNodeMCUand(Colorsoftheconnectingwires doesn't matterhere)
- EnsurethepowersupplyisgiventobothNodeMCUandsensorboardsasdiscussed in previousstep.

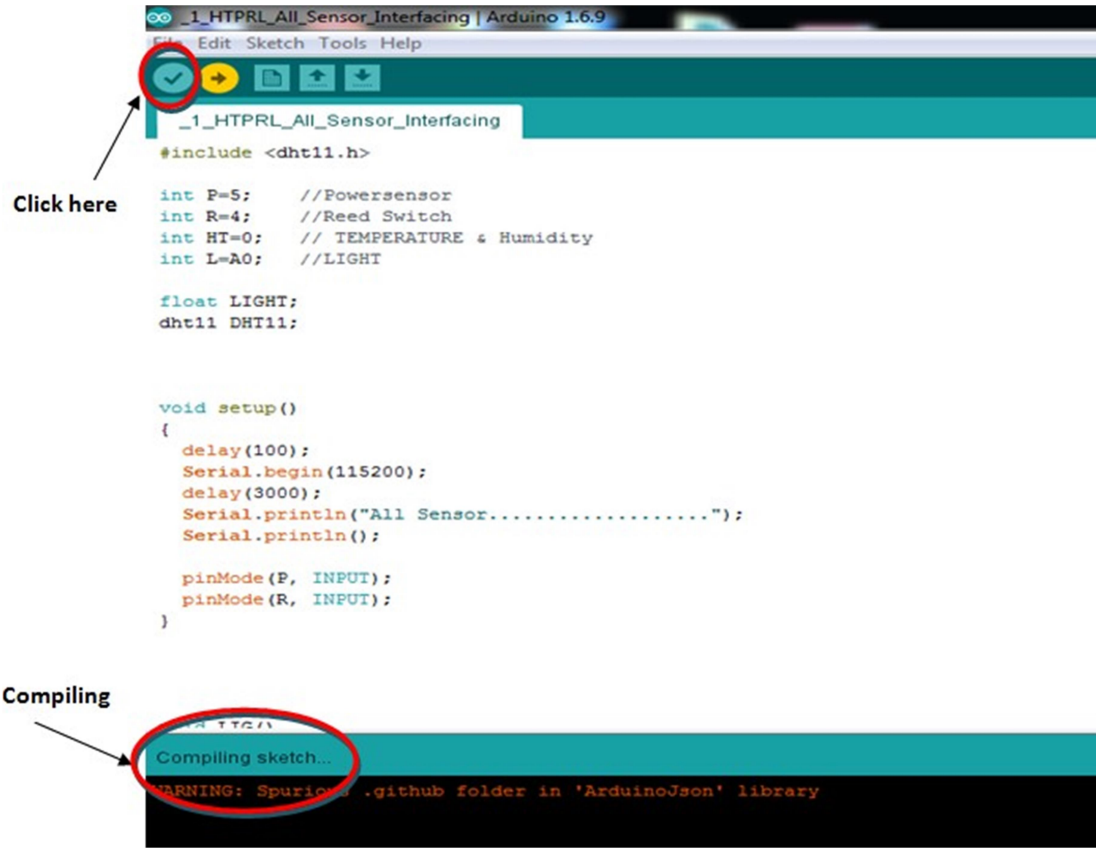
2. **Programming theNodeMCU:**

➢ **Compile/Save/UploadtheProjectcode:**

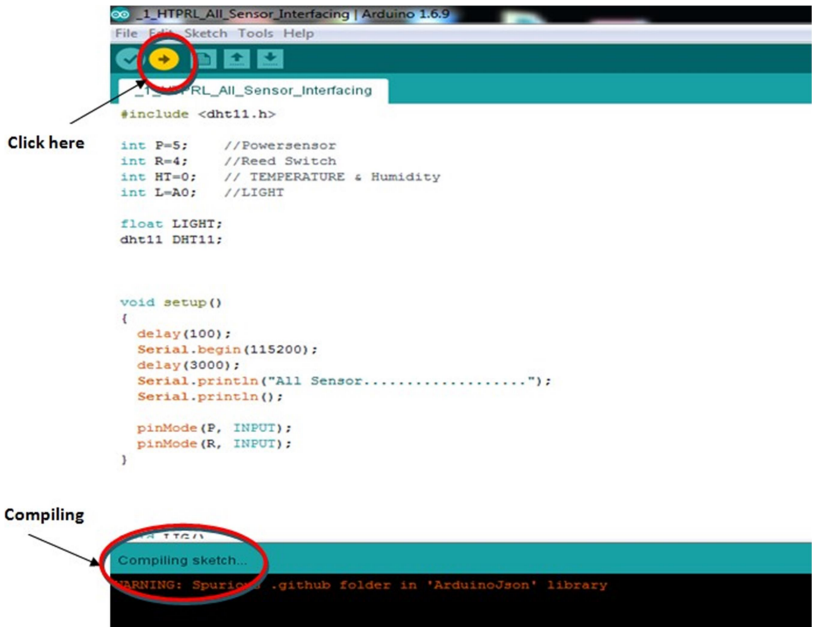
- Youcanuseeitherofthetwooptions(option1**or**option2)belowfortheproject code.
- Inthearduinocode,edityourownwifiSSIDandpasswordfields.
- **Option1:**StarttheArduinoIDEeitherfromstartmenuordesktopicon.Youwillbeseenawindo wasshownbeloworanyotherpreviouslyopenedsketch.Copythe codeandpasteintheArduinosketch(overwritetheexistingtextorcodeinthe sketch ifany).
- Nowsavetheproject:Goto**File**menu,youcanfind**save**option.Clickon**save**, giveaFilenameandclick**ok**.



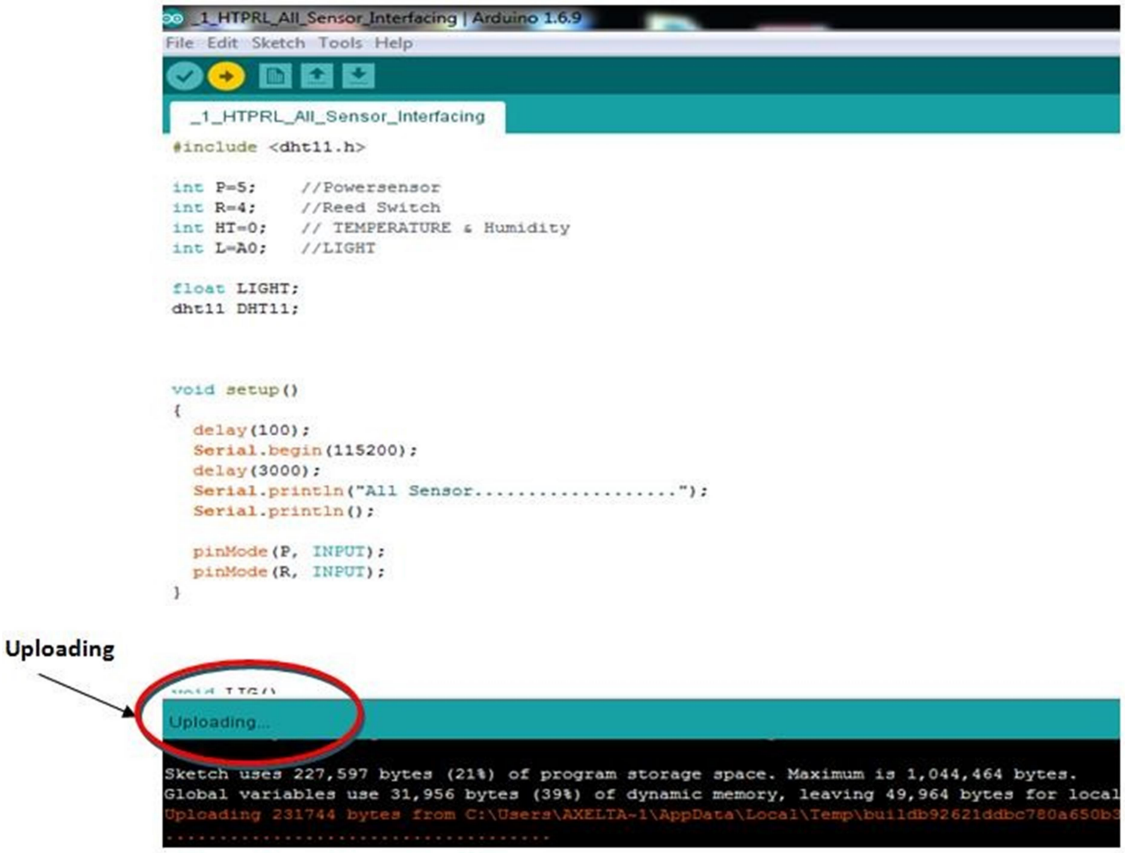
- Now,clickonthe‘Verify’buttonorgoto**sketch->Verify**.Thiswillcompilethecode as shownbelow:



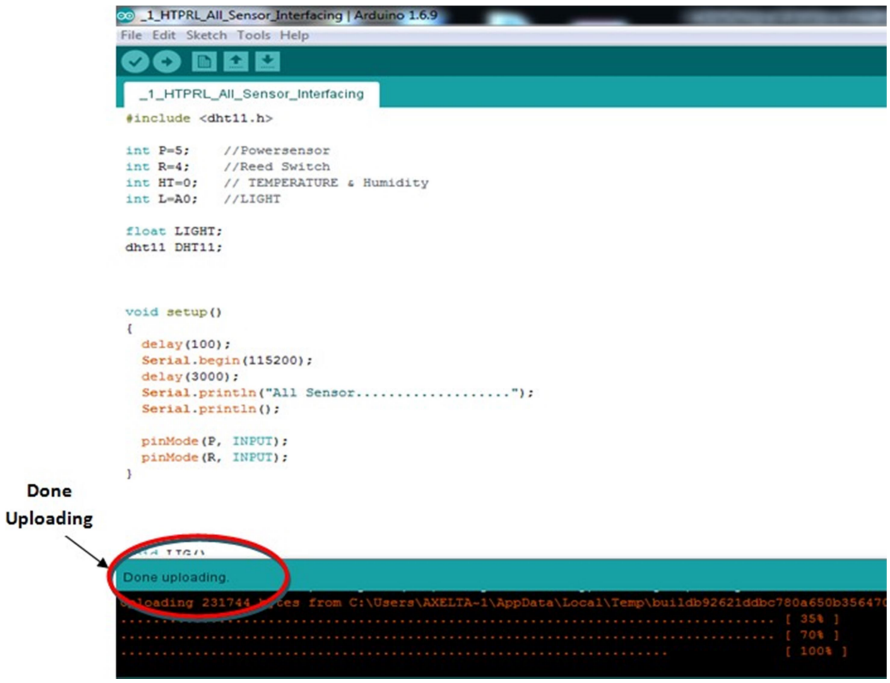
- Nowgoto**Tools>Port>COM#**SelectProperCOMPort,sameaswhichisdetectedunder devicemanager.
- Now,clickonthe‘Upload’buttonorgoto**sketch->Upload**.Thiswillcompilethe codeandthenuploadsittotheNodeMCUasshowninthebelowimages.


Click here
_1_HTPRL_All_Sensor_Interfacing | Arduino 1.6.9
File Edit Sketch Tools Help
_1_HTPRL_All_Sensor_Interfacing
#include <dht11.h>
int P=5; //Powersensor
int R=4; //Reed Switch
int HT=0; // TEMPERATURE & Humidity
int L=A0; //LIGHT
float LIGHT;
dht11 DHT11;
void setup()
{
delay(100);
Serial.begin(115200);
delay(3000);
Serial.println("All Sensor.....");
Serial.println();
pinMode(P, INPUT);
pinMode(R, INPUT);
}
Compiling
Compiling sketch...
.github folder in 'ArduinoJson' library

- Youcanobservethatthecodeisuploadingasshownintheimagebelow.



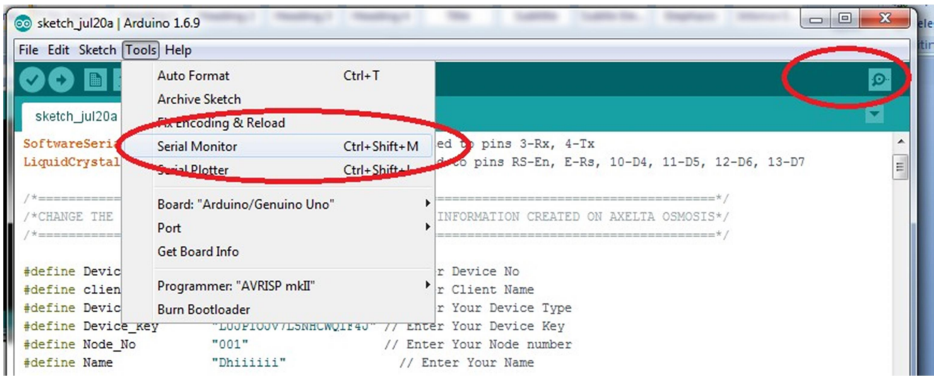
- Finallyyoucanobservethatthecodeisuploaded.

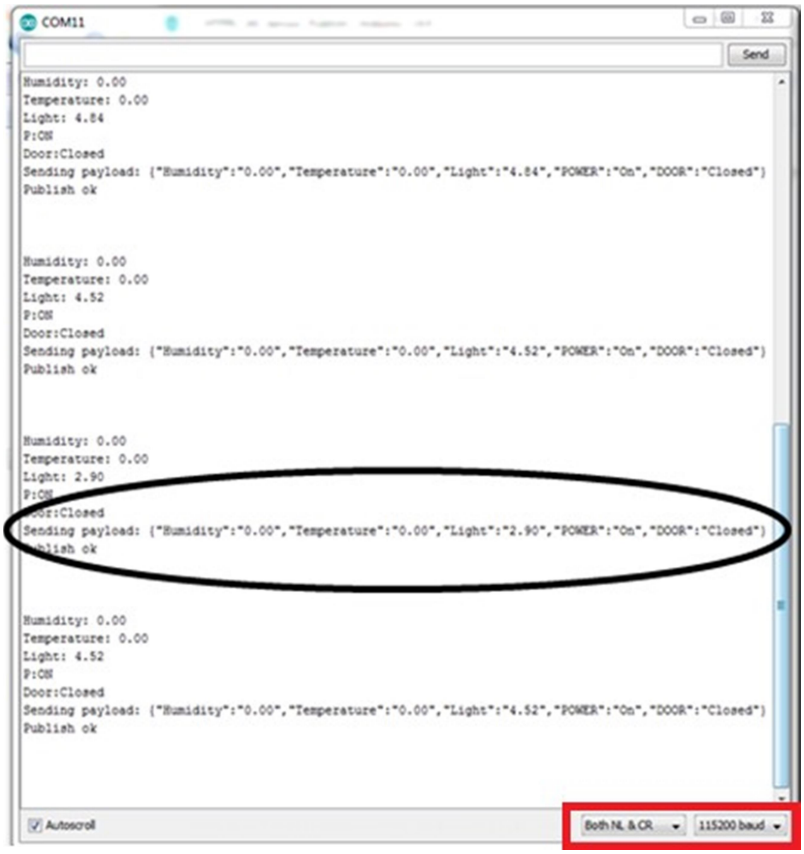

_1_HTPRL_All_Sensor_Interfacing | Arduino 1.6.9
File Edit Sketch Tools Help
_1_HTPRL_All_Sensor_Interfacing
#include <dht11.h>
int P=5; //Powersensor
int R=4; //Reed Switch
int HT=0; // TEMPERATURE & Humidity
int L=A0; //LIGHT
float LIGHT;
dht11 DHT11;
void setup()
{
delay(100);
Serial.begin(115200);
delay(3000);
Serial.println("All Sensor.....");
Serial.println();
pinMode(P, INPUT);
pinMode(R, INPUT);
}
Done
Uploading
Done uploading.
[35%]
[70%]
[100%]

Note: Ifyoufaceanyissueswiththeuploadingtheremightbeproblemintheselection ofCOMportorwiththedrivers.

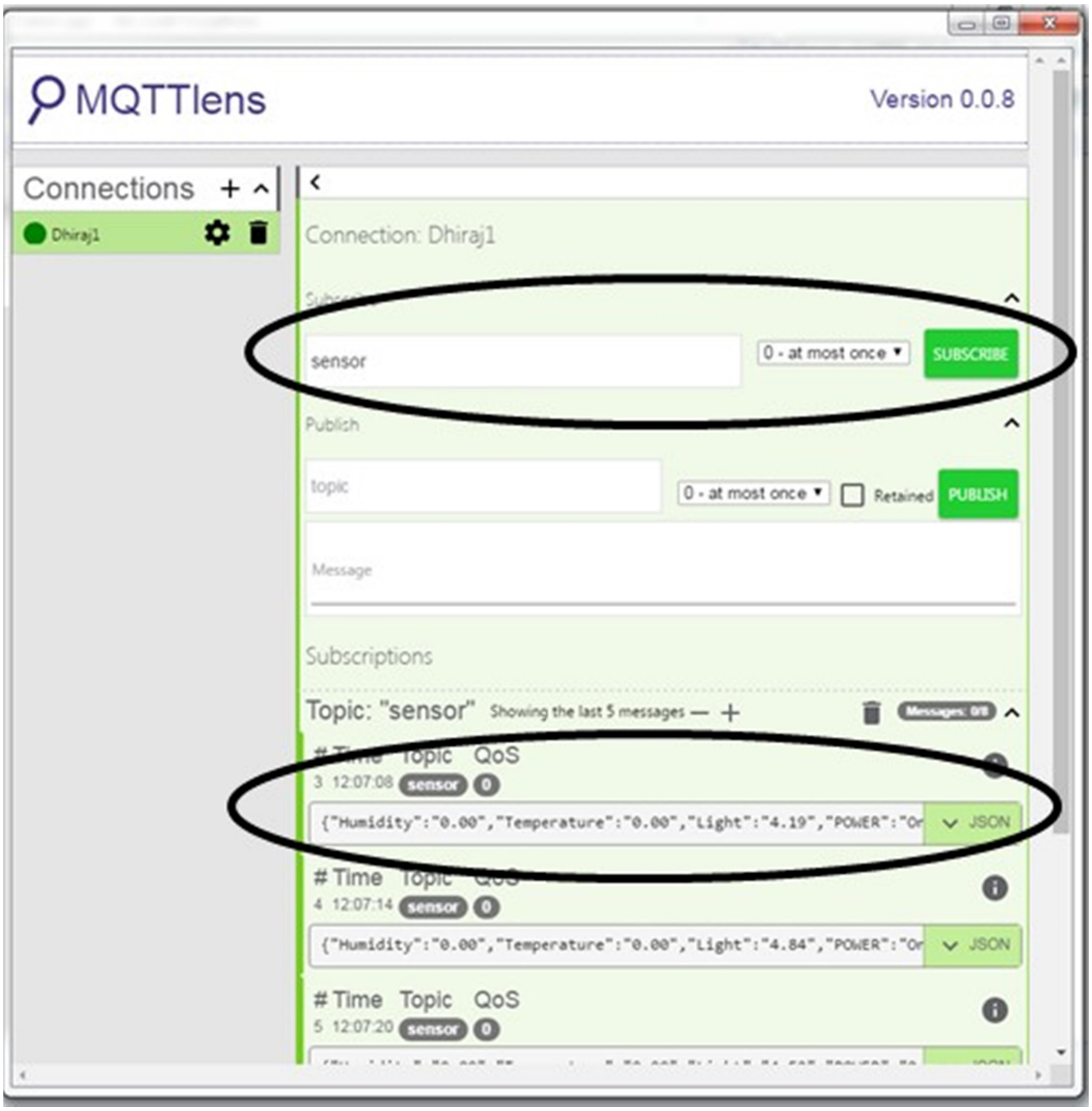
3. **Output:**

- Toseetheoutput,goto**Tools->SerialMonitor**orClickMagnifyingglasssymbolattop rightcorner.
- Itwillopenanewwindowcalledserialmonitor.Selecttheshowcasedoptionandbaud rateto**115200**atrightbottomsideasshownbelow.

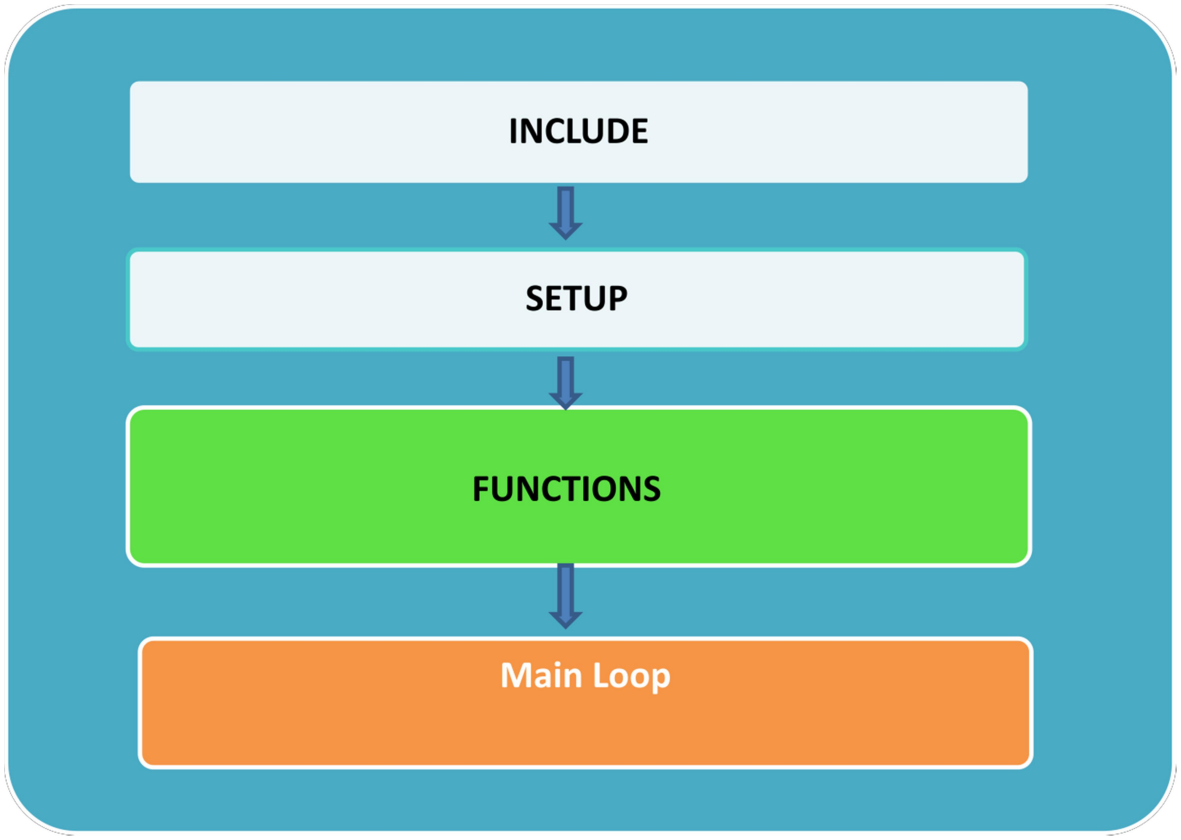



COM11
Send
Humidity: 0.00
Temperature: 0.00
Light: 4.84
P:ON
Door:Closed
Sending payload: {"Humidity":"0.00","Temperature":"0.00","Light":"4.84","POWER":"On","DOOR":"Closed"}
Publish ok
Humidity: 0.00
Temperature: 0.00
Light: 4.52
P:ON
Door:Closed
Sending payload: {"Humidity":"0.00","Temperature":"0.00","Light":"4.52","POWER":"On","DOOR":"Closed"}
Publish ok
Humidity: 0.00
Temperature: 0.00
Light: 2.90
P:ON
Door:Closed
Sending payload: {"Humidity":"0.00","Temperature":"0.00","Light":"2.90","POWER":"On","DOOR":"Closed"}
Publish ok
Humidity: 0.00
Temperature: 0.00
Light: 4.52
P:ON
Door:Closed
Sending payload: {"Humidity":"0.00","Temperature":"0.00","Light":"4.52","POWER":"On","DOOR":"Closed"}
Publish ok
Autoscroll
Both NL & CR
115200 baud

- OpenMQTTLensandenterTopicofsubscription(**char***pubtopic="sensor")andclicksubscribe.
- DatapostedbyNodeMCUwillstartappearingunderSubscriptionsasshownbelow:



4. **Understandingthecodeflow:**



- Theentirecodecanbedividedintodifferentsectionsasshownintheaboveimage. Briefexplanationisprovidedbelow:
  a) **INCLUDE**section**:**Containsallthenecessary*headerfiles,globalvariables,objects, functionprototypedeclarations*beingusedinthecode.
  b) **SETUP**section**:**Systemfunctionthatcontainsallthenecessarysettingsrequiredlike-*baudrate*(rateatwhichserialcommunicationtakesplaceintermsofbits/sec), *input & output pin configurations, wifi connectivity, generate client name and connectingtoMQTTbroker*.
  c) **FUNCTIONS**section**:**Containsthedefinitionsofvarioususerdefinedfunctionslike*LIG(),HUMTEM(),REED(),POWER(),callback(),macToStr(),reconnect()* used in the code.
  d) **MAINLOOP**section**:**Systemfunctionthatisexecutedfirstfromwhereallthe user definedfunctionsofthecodearecalled,payloadiscreatedandpublished.
- Furtherexplanationofthecodeisprovidedascommentswhereevernecessry:

a) **INCLUDE**section:

```
/*MQTT client library from knolleary.net - contains all necessary
APIs related to MQTT*/
#include <PubSubClient.h>
/*wifi client library from esp8266 - contains all necessary APIs
related to wifi*/
#include <ESP8266WiFi.h>
/*necessary header file for accessing Arduino DHT11 Temperature &
Humidity Sensor library*/
#include <dht11.h>

/**********************************************************
**************************/

const char* ssid = "Axelta"; /*Defne your own wifi SSID*/
const char* password = "Axelta140218"; /*Defne your own wifi
password*/

char* server = "osmosis.axelta.com"; /*Axelta MQTT broker-server
host name*/
//IPAddressserver(54, 201, 150, 33); /*Axelta MQTT broker-server IP
address*/
//IPAddressserver(192,168,1,246);/*ifyouhadanyloaclMQTTbrokerenterI
Paddressofthatbrokermachine*/

char* pubtopic = "sensor"; /*MQTT Publish topic*/

/*****************************************
*********************************/
StringclientName1;/*Definestringvariableforcli
entname*/
charmessage_buff[500]; /*character array for storing incoming msg
during subscription */

/*Initialize the variables as integers indexing the GPIOs to which
the sensors are connected to*/
```

```
intHT=0; // TEMPERATURE & Humidity sensor to GPIO0



dht11 DHT11; /*declaring dht11
object variable*/
WiFiClientwifiClient;


voidcallback(char*subtopic,byte*payload,unsignedintlength);/*Thisfunctioniscalledwhe
nmessagearrivedfor subscribedtopic*/

PubSubClientclient(server, 1883, callback, wifiClient); /*Necessary
parameters for MQTT clients*/
```

b) SETUPsection:

```
/*systemfunctionthatcontainsallthenecessarysettingsrequiredlike-
baudrate,input&outputpinconfigurations*/

voidsetup()

{
 delay(1000);
 Serial.begin(
 115200);
 delay(5000);

 Serial.println("AllSensorP
 ublishtoBroker");
 Serial.println();


 /*connecttolocalwifinetworkusing

 SSID&password*/

 Serial.println("Wifi

 disconnecting...");

 WiFi.disconnect();

 delay(1000);
 Serial.print("Conn
 ecting to ");
 Serial.println(ss
 id);

 WiFi.begin(ssid, password);
```

```
intdot=0;
/*wait till wifi network gets connected*/
while(WiFi.status() != WL_CONNECTED)
{
 delay(500);
 Serial.print
 (".");
 Serial.print
 (dot);
 dot++;
 if(dot==20||dot==40||dot==60| |dot==80| |dot==100){Serial.println(
 );} if(dot>120){Serial.println();Serial.println("Error Connecting Wifi
 Network");break;}
}
Serial.pr
intln();
Serial.pr
intln();
delay(10
00);
/*GenerateclientnamebasedonMACaddressandlast8bitsofmi
crosecondcounter*/ StringclientName;
clientName +=
"esp8266-";
uint8_t mac[6];
WiFi.macAddress(m
ac); clientName +=
macToStr(mac);
clientName +="-";
clientName+=String(micro
s()&0xff,16);
clientName1
=clientName;
delay(1000);
Serial.print("Con
necting to");
Serial.print(ser
ver);
Serial.print("
as ");
Serial.println(cl
ientName1);
delay(1000);
/*Connect to MQTT broker */
```

```
if(client.connect((char*) clientName1.c_str()))
{
Serial.println("Connected
to MQTT broker");
Serial.print("pubTopic
is: ");
Serial.println(pubtopic);
delay(1000);
if(client.publish(pubtopic, "hello from ESP8266")) /*publish
hello:test message*/
{
 Serial.println("Publish ok");
}
else
{
 Serial.println("Publish failed");
}
}
else
{
Serial.println("MQTT
connect failed");
Serial.println("Will
reset and try again...");
abort();
}
delay(1000);
}
```

c) **FUNCTIONS**section:

```
voidcallback(char* subtopic, byte* payload, unsigned int length)
{
 inti = 0;
 Serial.println("Messagearrived:topi
 c:"+String(subtopic));
 Serial.println("Length:"+String(l
 ength,DEC));
 for(i=0; i<length; i++)
 {
  message_buff[i] = payload[i];
 }
 message_buff[i] = '\0';
 String msgString = String(message_buff);
 Serial.println("Payload: " + msgString); /*Print all received msg
 contents for subscribed topics*/
}

/*Create unique client name using MAC address of NodeMCU board*/

String macToStr(constuint8_t* mac)
{
 String result;
 for(int i = 0; i< 6; ++i)
 {
  result += String(mac[i], 16);
  if(i<
   5)
   result
   +=':';
 }
 returnresult;
}
```

voidreconnect()

```
{
```

```
while(!client.connected())
{
 Serial.print("AttemptingMQTTconnection...");

 if  (client.connect((char*)clientName1.c_str()))
 {
  Serial.println("connected");
  if(client.publish(pubtopic,"h
  ellofromESP8266"))
  Serial.println("Publishok");
 else
  Serial.println("Publish
  failed"); delay(1000);
 }
else
 {
  Serial.print("failed,
  rc=");
  Serial.print(client.st
  ate());
  Serial.println("tryagai
  nin5seconds");
  delay(5000);
 }
}
```

```
}
/*userfunctiondefinitionforreadinganddisplayingHumidityandTemperature(analogvaluethroughdigitalpin)
serially*/

voidHUMTEM()
{
intchk = DHT11.read(HT); /*read humidity &temp value from 0-GPIO0*/
floatHUM=DHT11.humidity
; gHUM=HUM;
delay(10);
floatTEM=DHT11.temperature;
gTEM=TEM;
delay(10);
Serial.print("Humidit
y: ");
Serial.println(HUM);
Serial.print("Temperat
ure: ");
Serial.println(TEM);
}
```

d) MAIN LOOP section:

```
/*Main function from where the execution begins */
voidloop()
{
Serial.print("Axetla
Systems");
delay(1000);
while(1)
{
if(!client.connected()) /*if client not connected, call reconnect
function*/
{
reconnect();
```

```
  }


  else

  {
  /*calling the
  functions*/
  HUMTEM();

  delay(500);

/*Create string payload for publish*/

 String payload =
"{\"Humidity\":\"";
 payload += String(gHUM);

 payload +=
"\",\"Temperature\":\"";
 payload += String(gTEM);

  payload += "\"}";

/*publish payload string */

 if(client.connected())

 {

  Serial.print("Sending payload:
 "); Serial.println(payload);

  if(client.publish(pubtopic, (char*) payload.c_str()))

  {

  Serial.println("Publish ok"); Serial.println(); Serial.println();
  Serial.println();

  }else

  {

   Serial.println("Publish failed");

  }}

 delay(5000);

  }

  }

}
```

Week 9&10:

Introduction to Open Source Cloud Platforms for loT: OpenloT, ThingSpeak, thinger.io, Google Cloud Platform.

Sending Multiple Values ToThingSpeak:

ThingSpeak requires a user account and a channel. A channel is where you send data and where ThingSpeak stores data. Each channel has up to 8 data fields, location fields, and a status field. You can send data every 15 seconds to ThingSpeak, but most applications work well every minute.

1. Flow Diagram:

2.Practical to do:

- Sign up for new *User Account* – https://thingspeak.com/users/sign_up
- Create a new *Channel* by selecting *Channels*, *My Channels*, and then *New Channel*
- Note the *Write API Key* and *Channel ID*
- Download the Thingspeak library in Arduino.
- configure the unsigned long myChannelNumber = 31461 and const char * myWriteAPIKey = "LD79EOAAWRVYF04Y";

3.Hardware and Software Requirement:

Hardware required:

Arduino Uno

Software required:

Arduino Ide

Programming:

```
// This example selects the correct library to use based on the board selected under the Tools menu in the IDE.
// Yun, Ethernet shield, WiFi101 shield and MKR1000 are supported.
// EPS8266 and ESP32 are not compatible with this example.
// With Yun, the default is that you're using the Ethernet connection.
// If you're using a wi-fi 101 or ethernet shield (http://www.arduino.cc/en/Main/ArduinoWiFiShield),
```

uncomment the corresponding line below
// *

```
//#define USE_WIFI101_SHIELD
//#define USE_ETHERNET_SHIELD
#include<Thingspeak.h>

#if defined(ARDUINO_ARCH_ESP8266) || defined(ARDUINO_ARCH_ESP32)
      #error "EPS8266 and ESP32 are not compatible with this example."
#endif


#if !defined(USE_WIFI101_SHIELD) && !defined(USE_ETHERNET_SHIELD) &&
!defined(ARDUINO_SAMD_MKR1000) && !defined(ARDUINO_AVR_YUN)
  #error "Uncomment the #define for either USE_WIFI101_SHIELD or
USE_ETHERNET_SHIELD"
#endif


#if defined(ARDUINO_AVR_YUN)
    #include "YunClient.h"
YunClient client;
#else
  #if defined(USE_WIFI101_SHIELD) || defined(ARDUINO_SAMD_MKR1000)
    // Use WiFi
    #include <SPI.h>
    #include <WiFi101.h>
    char ssid[] = "<YOURNETWORK>";     //  your network SSID (name)
    char pass[] = "<YOURPASSWORD>";    // your network password
    int status = WL_IDLE_STATUS;
WiFiClient  client;
  #elif defined(USE_ETHERNET_SHIELD)
    // Use wired ethernet shield
    #include <SPI.h>
    #include <Ethernet.h>
    byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
EthernetClient client;
  #endif
#endif


#ifdef ARDUINO_ARCH_AVR
  // On Arduino:  0 - 1023 maps to 0 - 5 volts
  #define VOLTAGE_MAX 5.0
  #define VOLTAGE_MAXCOUNTS 1023.0
#elif ARDUINO_SAMD_MKR1000
```

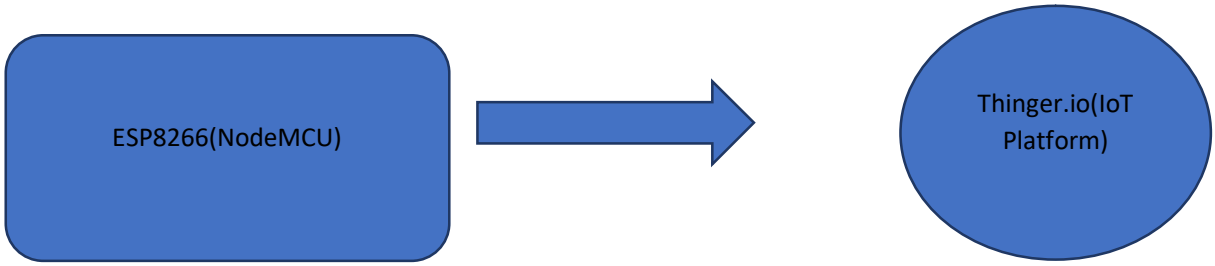
```
  // On MKR1000:  0 - 1023 maps to 0 - 3.3 volts
  #define VOLTAGE_MAX 3.3
  #define VOLTAGE_MAXCOUNTS 1023.0
#elif ARDUINO_SAM_DUE
  // On Due:  0 - 1023 maps to 0 - 3.3 volts
  #define VOLTAGE_MAX 3.3
  #define VOLTAGE_MAXCOUNTS 1023.0
#endif


  **** Visit https://www.thingspeak.com to sign up for a free account and
create
  **** a channel.  The video tutorial
http://community.thingspeak.com/tutorials/thingspeak-channels/
  **** has more information. You need to change this to your channel, and
your write API key
  **** IF YOU SHARE YOUR CODE WITH OTHERS, MAKE SURE YOU REMOVE YOUR
WRITE API KEY!!
unsigned long myChannelNumber = 31461;
const char * myWriteAPIKey = "LD79EOAAWRVYF04Y";
void setup() {
  #ifdef ARDUINO_AVR_YUN
Bridge.begin();
  #else
    #if defined(USE_WIFI101_SHIELD) || defined(ARDUINO_SAMD_MKR1000)
     WiFi.begin(ssid, pass);
    #else
Ethernet.begin(mac);
    #endif
  #endif
ThingSpeak.begin(client);
}


void loop() {
  // Read the input on each pin, convert the reading, and set each field
to be sent to ThingSpeak.
  // On Uno,Mega,Yun:  0 - 1023 maps to 0 - 5 volts
  // On MKR1000,Due: 0 - 4095 maps to 0 - 3.3 volts
  float pinVoltage = analogRead(A0) * (VOLTAGE_MAX / VOLTAGE_MAXCOUNTS);
ThingSpeak.setField(1,pinVoltage);


  // Write the fields that you've set all at once.
ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
```

```
delay(20000); // ThingSpeak will only accept updates every 15 seconds.
}
```

ESP8266 With Arduino IDE and Thinger.io:

This post will show how to connect the ESP8266 to the Thinger.io.

1.Flow Diagram:



2. Practical to do:

- Install the ESP8266 board package in Arduino IDE.
- Install Thinger.io package.
- Upload the code and select the appropriate ESP8266 board in IDE.

3. Hardware & Software Required:

Hardware Required:

- ESP8266
- LED's

Software Required:

- Arduino IDE

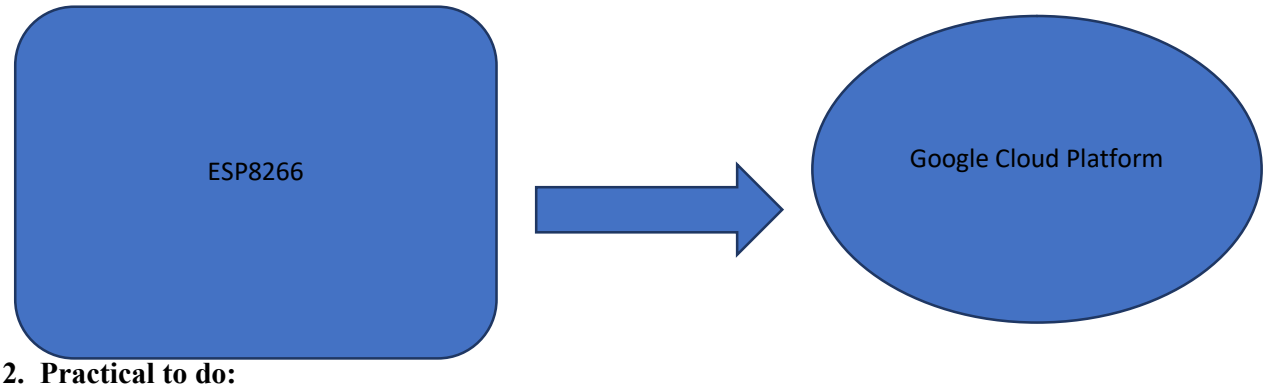
4. Programming:

```
#include <SPI.h>
#include <ESP8266WiFi.h>
#include <ThingerWifi.h>

#define USERNAME "your_user_name"
#define DEVICE_ID "your_device_id"
#define DEVICE_CREDENTIAL "your_device_credential"

#define SSID "your_wifi_ssid"
#define SSID_PASSWORD "your_wifi_ssid_password"

ThingerWifithing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

void setup() {
pinMode(BUILTIN_LED, OUTPUT);
```

```
thing.add_wifi(SSID, SSID_PASSWORD);

  // resource input example (i.e. turning on/off a light, a relay, configuring
a parameter, etc)
thing["led"] <<[](pson& in){
digitalWrite(BUILTIN_LED, in ? HIGH : LOW);
};

  // resource output example (i.e. reading a sensor value)
thing["millis"] >>[](pson& out){
out = millis();
};

  // resource input/output example (i.e. passing input values and do some
calculations)
thing["in_out"] = [](pson& in, pson& out){
out["sum"] = (long)in["value1"] + (long)in["value2"];
out["mult"] = (long)in["value1"] * (long)in["value2"];
  };
}

void loop() {
thing.handle();
  }
```

ESP8266 with Arduino and Google Cloud Platform:

1.Flow Diagram:



2. Practical to do:

- Import the code of “backoff.h”,”esp8266_wifi.h”,”cli.h” and dump them in Arduino IDE library.
- Use those dumped library in the Arduino code.

3.Hardware& Software Required:

Hardware Required:

- ESP8266
- LED’s

Software Required:

- Arduino IDE

Programming:

“backoff.h”

```
Ifndef__BACKOFF_H_
_
                    #define __BACKOFF_H__


                    int backOffCount = 0;
                    long minBackoff = 5000; // 1000 if you don't mind sending
                    lots of data
                    long maxBackoff = 60000;
                    long minJitter = 50;
                    long maxJitter = 1000;


                    long currDelay = minBackoff;
                    long lastRequestTime = millis();


                    void resetBackoff() {
                    backOffCount = 0;
                    }
                    // Returns true if the backoff duration has passed
                    bool backoff() {
                      if ((millis() - lastRequestTime) >currDelay) {
                    backOffCount++;
                    currDelay = (backOffCount * backOffCount * minBackoff) +
                    random(minJitter,maxJitter);
                        if (currDelay>maxBackoff) {
                    currDelay = maxBackoff;
                        }
                    Serial.printf("Waiting: %ld\n", currDelay);
                    delay(500); // FIXME remove
                    lastRequestTime = millis();
                        return true;
                      }
                      return false;
                    }
                    #endif
```

“cli.h”

```
                     #ifndef__CLI_H__
                             #define __CLI_H__

                                       boolean stopped = false;
```

```
 void cliLoop() {
String msg = "";
while (Serial.available() || stopped)
{
if (Serial.available()) {
msg = Serial.readStringUntil('\n');
}
if (msg == "stop") {
Serial.println("STOPPING!!!");
stopped = true;
}


if (msg == "go") {
Serial.println("Resume");
stopped = false;
}


if (msg == "sensor") {
String data = "Wifi: " +
String(WiFi.RSSI()) + " db";
Serial.println(data);
}


if (stopped) {
delay(10000);
Serial.print(".");
}
}
}
#endif
“esp8266_wifi.h”
    #define __ESP8266_WIFI_H__
    #include <ESP8266WiFi.h>
    #include <WiFiClientSecure.h>
    #include <time.h>
    #include <rBase64.h>
    #include <CloudIoTCore.h>


    #include "ciotc_config.h" //
    Wifi configuration here
    // Clout IoT configuration that
    you don't need to change
    const char* host =
```

```
CLOUD_IOT_CORE_HTTP_HOST;
const int httpsPort =
CLOUD_IOT_CORE_HTTP_PORT;
CloudIoTCoreDevice* device;
unsigned int priv_key[8];
unsigned long iss = 0;
String jwt;
booleanwasErr;
// Helpers for this board
String getDefaultSensor() {
return  "Wifi: " +
String(WiFi.RSSI()) + "db";
}
String getJwt() {
if (iss == 0 || time(nullptr) -
iss> 3600) {  // TODO: exp in
device
// Disable software watchdog as
these operations can take a
while.
ESP.wdtDisable();
Serial.println("Refreshing
JWT");
iss = time(nullptr);
jwt = device->createJWT(iss);
ESP.wdtEnable(0);
}
return jwt;
}
void setupWifi() {
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
Serial.println("Connecting to
WiFi");
while (WiFi.status() !=
WL_CONNECTED) {
delay(100);
}
configTime(0, 0, "pool.ntp.org",
"time.nist.gov");
Serial.println("Waiting on time
sync...");
while (time(nullptr) <
1510644967) {
delay(10);
}
device = new CloudIoTCoreDevice(
```

```
project_id, location,
registry_id, device_id,
private_key_str);
// Device/Time OK, refresh JWT
Serial.println(getJwt());
}
void doRequest(WiFiClientSecure*
client, booleanisGet, String
postData) {
String header =
String("POST  ") +
device-
>getSendTelemetryPath().c_str()
+
String(" HTTP/1.1");
String authstring =
"authorization: Bearer " +
String(getJwt().c_str());
if (isGet) {
header = String("GET ") +
device->getLastConfigPath() + "
HTTP/1.1";
authstring = "authorization:
Bearer " + getJwt();
}
String request = header + "\n" +
"host:
cloudiotdevice.googleapis.com\n"
+
"cache-control: no-cache\n" +
authstring + "\n";
if (isGet) {
request = request +
String("\n");
} else {
request = request +
"method: post\n" +
"content-type:
application/json\n" +
"content-length:" +
String(postData.length()) +
"\n\n" + postData + "\n\n";
}
Serial.println("Connecting to "
+ String(host));
client->connect(host,
httpsPort);
```

```
Serial.println("Verifying
certificate");
if (!client->verify(fingerprint,
host)) {
Serial.println(
"Error: Certificate not
verified! "
"Perhaps the fingerprint is
outdated.");
// return;
}


// Connect via https.
Serial.println(request);
client->print(request);
unsigned long timeout =
millis();
while (client->available() == 0)
{
if (millis() - timeout > 5000) {
Serial.println(">>> Client
Timeout !");
delay(10000);
}
}
}
void sendTelemetry(String data)
{
String postdata =
String("{\"binary_data\": \"") +
rbase64.encode(data) +
String("\"}");
WiFiClientSecure client;
doRequest(&client, false,
postdata);
while (!client.available()) {
delay(100);
Serial.print('.');
}
Serial.println();
while (client.connected()) {
String line =
client.readStringUntil('\n');
if (line.startsWith("HTTP/1.1
200 OK")) {
// reset backoff
```

```
resetBackoff();
}
Serial.println(line);
if (line == "\r") {
break;
}
}
while (client.available()) {
String line =
client.readStringUntil('\n');
Serial.println(line);
}
Serial.println("Complete.");
}
// Helper that just sends
default sensor
void sendTelemetry() {
sendTelemetry(getDefaultSensor()
);
}
void getConfig() {
WiFiClientSecure client;
doRequest(&client, true, "");
// Handle headers here
while (client.connected()) {
String line =
client.readStringUntil('\n');
Serial.println(line);
if (line == "\r") {
Serial.println("--");
break;
}
}
// Handle respone body here
while (client.available()) {
String line =
client.readStringUntil('\n');
Serial.println(line);
if (line.indexOf("binaryData") >
0) {
// Reset backoff
resetBackoff();
String val =
line.substring(line.indexOf(":
") + 3,line.indexOf("\","));
if (val == "MQ==") {
Serial.println("LED ON");
```

```
digitalWrite(LED_BUILTIN, HIGH);
} else {
Serial.println("LED OFF");
digitalWrite(LED_BUILTIN, LOW);
}
}
}
client.stop();
}
#endif //__ESP8266_WIFI_
```

Main Program:

```
#include "esp8266_wifi.h"
#include "cli.h"
#include "backoff.h"


void setup() {
  // put your setup code here, to run once:
Serial.begin(115200);
setupWifi();
pinMode(LED_BUILTIN, OUTPUT);
}


void loop() {
  if (backoff()) {
    // Log signal strength
sendTelemetry();
delay(1000);
getConfig();
  }


delay(10); // too fast
cliLoop();
}
```

Setting Webserver On Raspberry Pi

Practical Steps to be followed:

- Install Apache :

```
sudo apt-get update
```

- Then install the apache2 package with this command

```
sudo apt-get install apache2 -y
```

- Test the web server:

By default, Apache puts a test HTML file in the web folder. This default web page is served when you browse to `http://localhost/` on the Pi itself, or `http://192.168.1.10` (whatever the Pi's IP address is) from another computer on the network.

- To find the Pi's IP address, type `hostname -I` at the command line (or read more about finding your IP address).
- The default web page is just an HTML file on the filesystem. It is located at `/var/www/html/index.html`
- Navigate to this directory in a terminal window and have a look at what's inside:

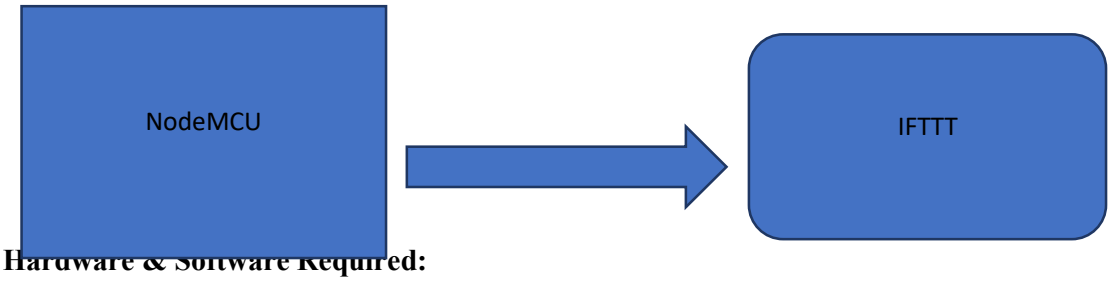
```
cd /var/www/html
ls -al
```

- This shows that by default there is one file in `/var/www/html/` called `index.html` and it is owned by the `root` user (as is the enclosing folder). In order to edit the file, you need to change its ownership to your own username. Change the owner of the file (the default `pi` user is assumed here) using `sudochown pi: index.html`

Your Own Website:

If you know html you can put your own html files in this directory and serve them as a website on your local network.

NodeMCU with IFTTT:



Hardware & Software Required:

Hardware required:

NodeMCU

Software required:

Arduino IDE

Programming:

```
#include<ESP8266WiFi.h
>
                          // Include the ESP8266 WiFiSecure library:
```

```
#include <WiFiClientSecure.h>


// WiFi Network Definitions //

// Replace these two character strings with the name
and
// password of your WiFi network.
const char mySSID[] = "YourWiFiName";
const char myPSK[] = "YourWifiPassword";


// IFTTT Constants //

// IFTTT destination server:
const char* IFTTTServer = "maker.ifttt.com";
// IFTTT https por:
const int httpsPort = 443;
// IFTTT Event:
const String MakerIFTTT_Event = "button";
// IFTTT private key:
const String MakerIFTTT_Key = "YourPrivateKeyIFTTT";
String httpHeader = "POST
/trigger/"+MakerIFTTT_Event+"/with/key/"+MakerIFTTT_K
ey +" HTTP/1.1\r\n" +
                    "Host: " + IFTTTServer + "\r\n" +
                    "Content-Type: application/x-www-
form-urlencoded\r\n\r\n";
void setup()
{
  int status;
Serial.begin(9600);
Serial.println();
Serial.print(F("connecting to "));
Serial.println(mySSID);
WiFi.begin(mySSID, myPSK);
  while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
  }
Serial.println("");
Serial.println(F("WiFi connected"));
Serial.println(F("IP address is: "));
Serial.println(WiFi.localIP());
Serial.println(F("Press any key to post to IFTTT!"));
}
void loop()
```

```
{
  // If a character has been received over serial:
  if (Serial.available())
  {
    // !!! Make sure we haven't posted recently
    // Post to IFTTT!
postToIFTTT();
    // Then clear the serial buffer:
    while (Serial.available())
Serial.read();
  }
}
void postToIFTTT()
{
  // Create a client, and initiate a connection
WiFiClientSecure client;
  if (client.connect(IFTTTServer, httpsPort) <= 0)
  {
Serial.println(F("Failed to connect to server."));
    return;
  }
Serial.println(F("Connected."));
Serial.println(F("Posting to IFTT Event!"));
client.print(httpHeader);
  // available() will return the number of characters
  // currently in the receive buffer.
  while (client.available())
Serial.write(client.read()); // read() gets the FIFO
char
  // connected() is a boolean return value - 1 if the
  // connection is active, 0 if it's closed.
  if (client.connected())
client.stop(); // stop() closes a TCP connection.
}
```

Week 11& 12:

IoT based Home Security System with Email

IntruderdetectionandsendinganemailusingNodeMCU

Practical's Objective:

TodetecttheifthereisanyintruderthroughPIRsensorandsendinganemailusingNodeMcu.

Description of Sensor Elements:

A **PIR** (Passive Infrared Sensor) is used as a motion detector.

Note:Asperitsnames,thetechnicaldatasheetsofthesensorscanbefoundintheinternetfor moreinformation.

1. **SensorPracticalHardwareflowDiagram:**

2. **H/w&S/wRequirements:**

Hardware required:

- NodeMCUBoard
- PIRSensors
- Connectors
- USBcable

Software Requirement:

- Aurdino IDE -**http://arduino.cc/en/Main/OldSoftwareReleases**

ConnectingSensorstoArduinoBoard:

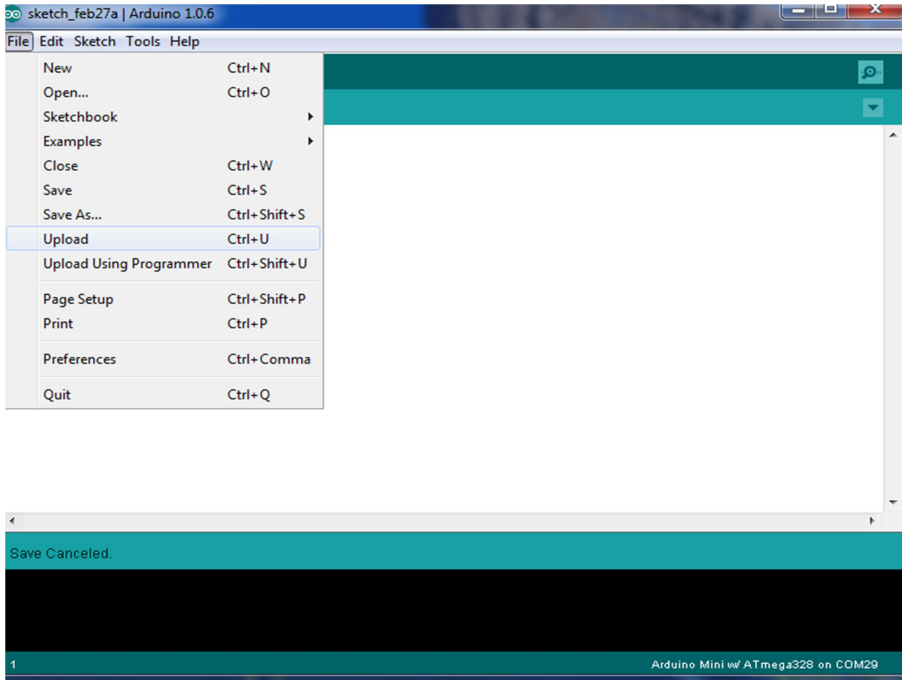
Sensor	LCDShield
PIR	D5
Vcc	5V
GND	GND

- ConnectPIRsensorOutputpinto**pinD5**onNodeMCUBoardusingasinglepinwire.
- ConnectUSBcablefromNodeMCUboardtoPC.

Note: Beforegettingstarted,checkthattheArduinosoftwareisinstalled.Whenconnectedforthe firsttime,checktheArduinoUNOboardUSBdriversoftwareisupdatedaspertheInstallation document.

Programming:

1. StartArduinoIDEbyselectingtheprograminthewindowsstartmenu.YouwillgettheWindowas below.



Steps for constructing a simple 'C' program in Arduino.

Createanewsketch[program] **#include**

<ESP8266WiFi.h>#includeGsender.h"

```
#pragma region Globals
constchar*ssid="";       //WIFInetworkname
constchar*password="";      // WIFI
network password uint8_t
connection_state=0;

//ConnectedtoWIFIornot

uint16_treconnect_interval=10000;
                                  //Ifnotconnectedwai
ttimetotryagain#pragma endregionGlobals

int P=5;


uint8_t WiFiConnect(const char* nSSID = nullptr, const
char* nPassword = nullptr)
{
  static uint16_t attempt = 0; Serial.print("Connecting to");
  if(nSSID) {

    WiFi.begin(nSSID, nPassword); Serial.println(nSSID);

  } else {
    WiFi.begin(ssid, password); Serial.println(ssid);

  }
  uint8_t i = 0;

  while(WiFi.status()!= WL_CONNECTED &&i++ < 50)
```

```
  {
    delay(200); Serial.print(".");
  }
  ++attempt; Serial.println(""); if(i == 51) {
    Serial.print("Connection: TIMEOUT on
    attempt: ");
    Serial.println(attempt);
    if(attempt % 2 == 0)
      Serial.println("Check if access point available
    or SSID and Password\r\n"); return false;
  }
  Serial.println("Connection:E
  STABLISHED");
  Serial.print("Got IP
  address: ");
  Serial.println(WiFi.loca
  lIP());
  return true;
}
void Awaits()
{
  uint32_t ts = millis(); while(!connection_state)
  {
```

```
    delay(50);

    if(millis()>(ts+reconnect_interval)&&!
      connection_state){
      connection_state =WiFiConnect();

      ts = millis();
    }
  }
}

void setup()
{
  Serial.begin(115200); connection_state=WiFiConnect();

  if(!connection_state) //
    if not connected
    toWIFIAwaits();
              //constantlyt
    ryingtoconnect

  Gsender *gsender=Gsender::Instance();
                                  //Gettingpointe
  rtoclassinstanceStringsubject="Subjectisoptio
  nal!";

  if((p==1))
  {
  if(gsender->Subject(subject)-
    >Send("boris.on@live.com", "Setup test")) {
    Serial.println("Message send.");
```

```
  } else {

    Serial.print("Error

    sending message: ");

    Serial.println(gsend

    er->getError());

  }
  }
}
```

```
void loop(){}
```

- NowgotoyourGoogleaccountsettingsandenable"Allowlesssecureapps"atthebottom of thepage.
- Setyourwifiaccesspointname(SSID)andpassword.
- InSetup()functionfindandchangetheexample@gmail.comtomailIDtowhichyouwant to send amail.

```
if(gsender->Subject(subject)->Send("example@gmail.com","MotionDetected"))
```

- Now open Gsender.htab
- WeneedBase64encodedemailaddressandpasswordofgmailaccountwhichwillbeused to send emails use https://www.base64encode.org/ for encoding, result must be somethinglike.

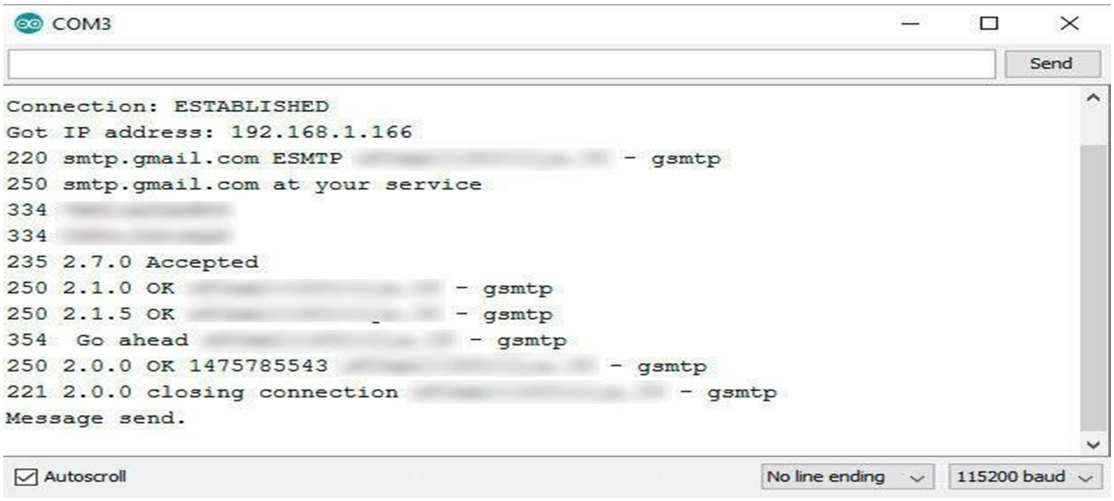
```
const char* EMAILBASE64_LOGIN = "Y29zbWkxMTExMUBnbWFpbC5jb20=";
const char* EMAILBASE64_PASSWORD = "TGFzZGFzZDEyMzI=";
```

- Now set fromfield

```
constchar*FROM="your_email@gmail.com";
```

Nowsavetheproject:Goto**File**menu,youcanfind**save**optionast hewindowsshownbelow
Clickon**save**,giveaFilenameandclick**ok**.

Nowgotothe**File**menu,clickon**Upload**option.Thenthecodeiscompile dandloadedintonodemcuboard.



OUTPUT: