

IOT BASED WEATHER STATION

A PROJECT REPORT

in partial fulfillment for the award of the degree

of

***BACHELOR OF TECHNOLOGY IN ELECTRONICS AND
COMMUNICATION ENGINEERING***

AND

BACHELOR OF COMPUTER APPLICATIONS

Under the Guidance of

SHOUVIK SARKAR

Project Carried Out At



Ardent Computech Pvt Ltd (An ISO 9001:2015 Certified)

CF-137, Sector - 1, Salt Lake City, Kolkata - 700 064

Submitted By

RISHI TRIPATHI

SUKANYA GUPTA



Future Institute of Engineering and Management Sonarpur, West

Bengal, INDIA

JUNE – JULY 2018

In association with



(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

1. Title of the Project: **IOT BASED WEATHER STATION**

2. Project Members: 1. RISHI TRIPATHI
2. SUKANYA GUPTA

3. Name and Address of the Guide: **MR. SHOUVIK SARKAR**

Senior Faculty

Ardent Computech Pvt Ltd (An ISO 9001:2015 Certified)

Module 132, SDF Building, Sector 5, Salt Lake, Kolkata - 64

Ph.D* M.Tech.* B.E*/B.Tech.* MCA* M.Sc.*

4. Educational Qualification of the Guide: ☐ ☒ Y ☐ ☐ ☐

5. Working / Training experience of the Guide: **7 Years**

6. Project Version Control History

Version	Primary Authors	Description of Version	Date Completed
Final	1. RISHI TRIPATHI 2. SUKANYA GUPTA	Project Report	30 th July, 2018

1.

2.

Signatures of Team Members
Approval

Signature of

Date:

Date:

For Office Use Only

Mr. SHOUVIK SARKAR
Project Proposal Evaluator

☐ **Approved**

☐ **Not Approved**

PROJECT RESPONSIBILITY FORM

WEATHER STATION

GROUP NO.	SL.NO.	NAME OF MEMBER	RESPONSIBILITY
1	1	RISHI TRIPATHI	SOFTWARE,CODING,HARDWARE
	2	SUKANYA GUPTA	CIRCUITRY,HARDWARE

Each group member must participate in project development and developing the ideas for the required elements. Individual group members will be responsible for completing tasks which help to finalize the project and the performance. All group members must be assigned a task.

Date:

Name of the Students

1. RISHI TRIPATHI
2. SUKANYA GUPTA

Signatures of the students

a.

b.

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled “**IOT BASED WEATHER STATION**” in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY IN ELECTRONICS AND COMMUNICATION ENGINEERING AND BACHELOR OF COMPUTER APPLICATIONS** at **ARDENT COMPUTECH PVT. LTD, SALT LAKE, KOLKATA, WEST BENGAL**, is an authentic work carried out under the guidance of **MR. SHOUVIK SARKAR**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date:

Name of the Students

1. RISHI TRIPATHI
2. SUKANYA GUPTA

Signature of the students

- a.
- b.



Ardent Computech Pvt Ltd (An ISO 9001:2008 Certified)
CF-137, Sector - 1, Salt Lake City, Kolkata - 700 064

CERTIFICATE

This is to certify that this proposal of minor project entitled “**IOT BASED WEATHER STATION**” is a record of bona fide work, carried out by **1. RISHI TRIPATHI** and **2.SUKANYA GUPTA** under my guidance at **ARDENT COMPUTECH PVT LTD.** In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF COMPUTER SCIENCE** and as per regulations of the **ARDENT®**. To the best of my knowledge, the results embodied in this report, are original innature and worthy of incorporation in the present version of the report.

Guide / Supervisor

Mr. SHOUVIK SARKAR

FACULTY

Ardent Computech Pvt Ltd (An ISO 9001:2015 Certified)
Module 132, SDF Building, Sector 5, Salt Lake, Kolkata – 64

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. We take this sincere opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project work.

We would like to show our greatest appreciation to Mr. Shouvik Sarkar sir, Project Manager at Ardent, Kolkata. We always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

INDEX

TABLE OF CONTENTS

Name of the Topic	Page No.
IOT Based Weather Station	1
In Association With	2
Project Responsibility Form	3
Declaration	4
Certificate	5
Acknowledgement	6
Table of Content	7
Introduction	9
History	10
Application	10
Advantage and Disadvantage	11
Criticism and Controversies	13
Piracy and Autonomy	14
Data storage	15
Conclusion	16
Introduction	18
Objective	20
Components required (hardware/software)	20
Description of components	21

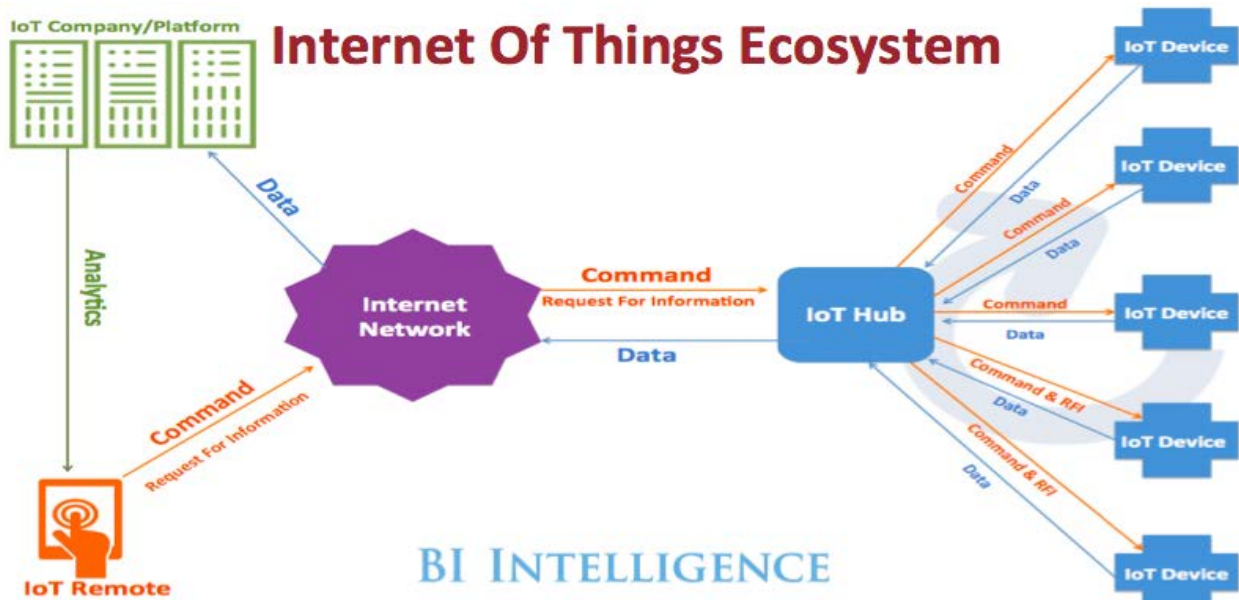
Circuit connection	33
Working code(lm 35)	36
Pictures (uploading/circuitry/output)	41
Pictures (uploading/circuitry/output)	43
Working code(dht 11)	44
Implementation with Thinkspeak	48
Conclusion and future scope	50
Reference links	51

INTRODUCTION

The **Internet of Things (IoT)** is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect and exchange data creating opportunities for more direct integration of the physical world into computer-based systems, resulting in efficiency improvements, economic benefits, and reduced human exertions.

The number of IoT devices increased 31% year-over-year to 8.4 billion in 201 and it is estimated that there will be 30 billion devices by 2020 The global market value of IoT is projected to reach \$7.1 trillion by 2020.

IoT involves extending internet connectivity beyond standard devices, such as desktops, laptops, smartphones and tablets, to any range of traditionally *dumb* or non-internet-enabled physical devices and everyday objects. Embedded with technology, these devices can communicate and interact over the internet, and they can be remotely monitored and controlled.



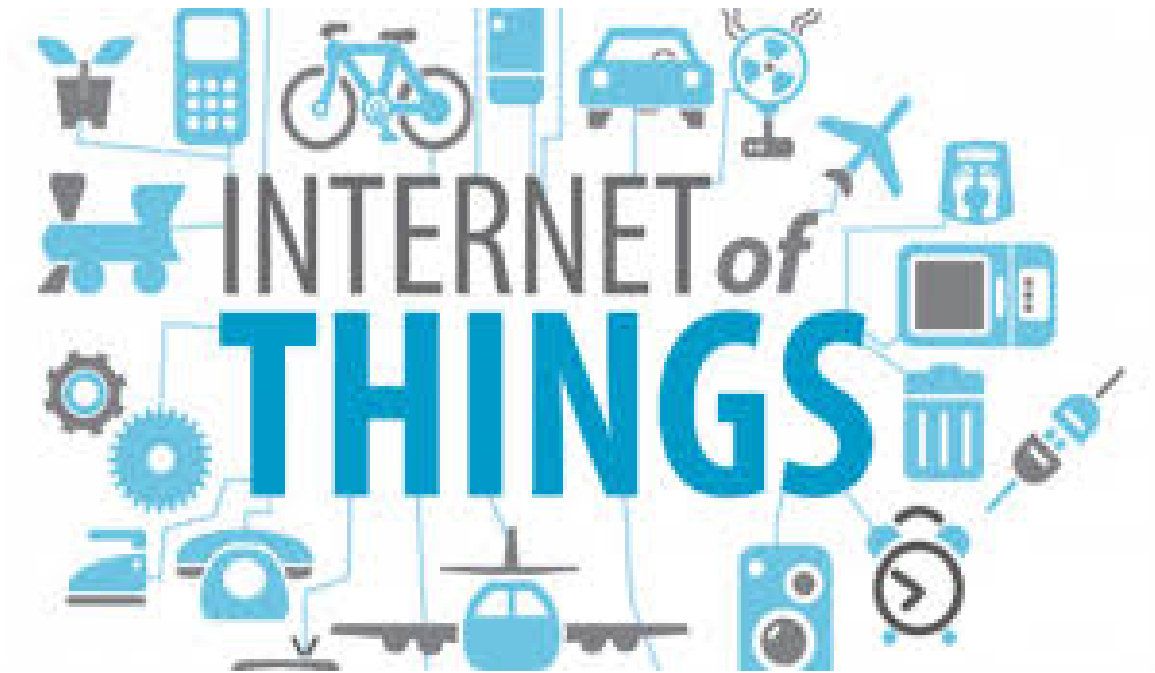
HISTORY

The definition of the Internet of things has evolved due to convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems^lTraditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), and others all contribute to enabling the Internet of things.

Applications

A growing portion of IoT devices are created for consumer use, including connected vehicles, home automation/smart home, wearable technology, connected health, and appliances with remote monitoring capabilities

Costs of operation in all infrastructure related areas. Even areas such as waste management can benefit from automation and optimization that could be brought in by the IoT.



ADVANTAGES AND DISADVANTAGES OF IOT

Advantages

Here are some advantages of IoT:

1. **Data:** It gets easier for us to take the right decision when we have more information. Knowing the weather conditions beforehand we can plan out day to day routine in an easier and proper way.
2. **Tracking:** The system can keep a track both on the quality and the viability of things at home. If we know the expiration date of products before one consumes them improves safety and quality of life. Also, you will never run out of anything when you need it at the last moment.

3. **Time:** The amount of time saved in monitoring and the number of trips done otherwise would be tremendous

. 4. **Money:** The financial aspect is the best advantage. This technology could replace humans who are in charge of monitoring and maintaining supplies.

5.**Emergency:** IOT based devices can be used to monitor the vitals of human beings such as old people who might be in danger ,the device could sense and send the information to the nearest hospital so that immediate help could be provided to the respective person.

Disadvantages

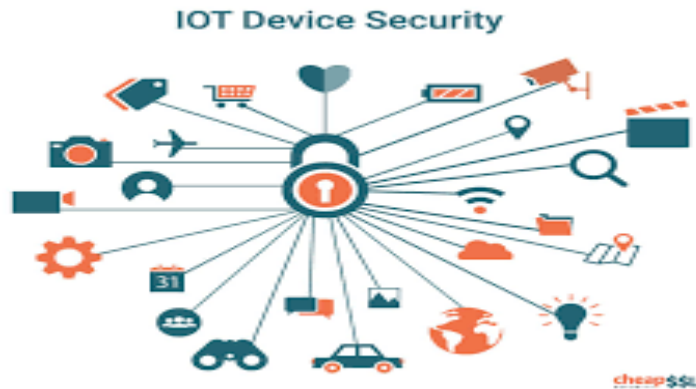
Here are some disadvantages of IoT:

1. **Compatibility:** As of now, there is no standard for tagging and monitoring with sensors. A uniform concept like the USB or Bluetooth is required which should not be that difficult to do.

2. **Complexity:** There are several opportunities for failure with complex systems providing us with incorrect/false data which would make us unnecessarily spend time or energy on unimportant things.Or if there is a software bug causing the printer to order ink multiple times when it requires a single cartridge.

3. **Privacy/Security:** Privacy is a big issue with IoT. All the data must be encrypted so that data about your financial status or how much milk you consume isn't common knowledge at the work place or with your friends. It could share our personal details with others and cause disruption in our privacy.

4. **Safety:** There is a chance that the software can be hacked and your personal information misused. There could be many possibilities. Your prescription being changed or your account details being hacked could put you at risk. Hence, all the safety risks become the consumer's responsibility



CRITICISM AND CONTROVERSIES

Platform fragmentation

IoT suffers from platform fragmentation and lack of technical standards a situation where the variety of IoT devices, in terms of both hardware variations and differences in the software running on them, makes the task of developing applications that work consistently between different inconsistent technology ecosystems hard. Customers may be hesitant to bet their IoT future on a proprietary software or hardware devices that uses proprietary protocols that may fade or become difficult to customize and interconnect. IoT's amorphous computing nature is also a problem for security, since patches to bugs found in the core operating system often do not reach users of older and lower-price devices. One set of researchers say that the failure of vendors to support older devices with patches and updates leaves more than 87% of active Android devices vulnerable.

PRIVACY and AUTONOMY

The Internet of things offers immense potential for

- 1. Empowering citizens**
- 2. Making government transparent**
- 3. Broadening information access.**
- 4. Howard cautions, however, that privacy threats are enormous, as is the**

potential for social control and political manipulation.

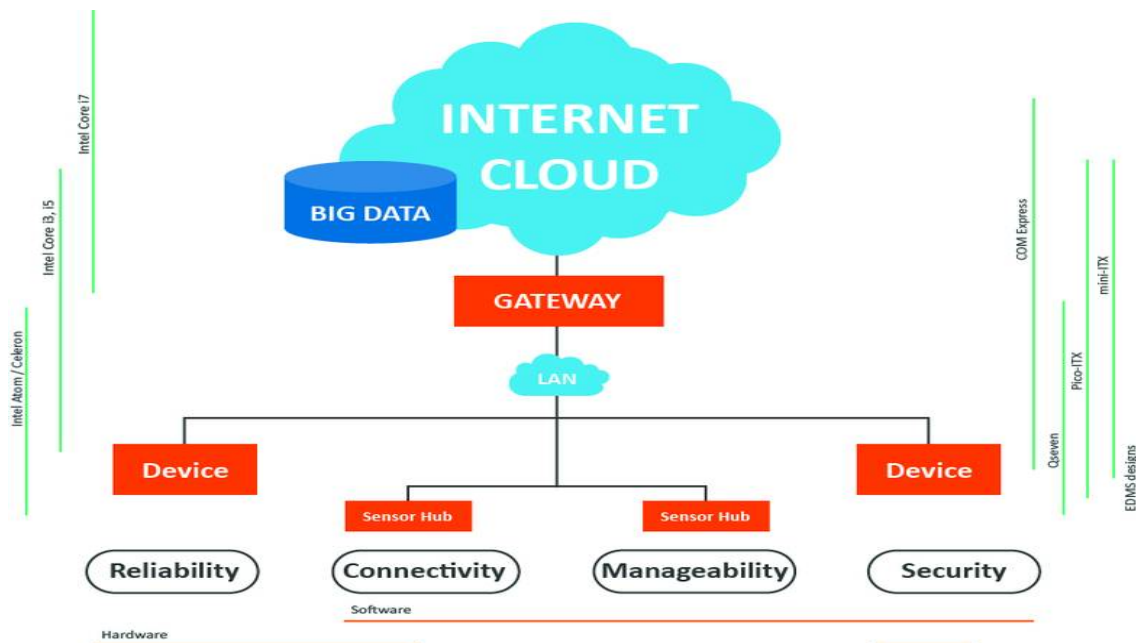


Concerns about privacy have led many to consider the possibility that big data infrastructures such as the Internet of things and data

mining are inherently incompatible with privacy. Writer Adam Greenfield claims that these technologies are not only an invasion of public space but are also being used to perpetuate normative behavior, citing an instance of billboards with hidden cameras that tracked the demographics of passersby who stopped to read the advertisement.

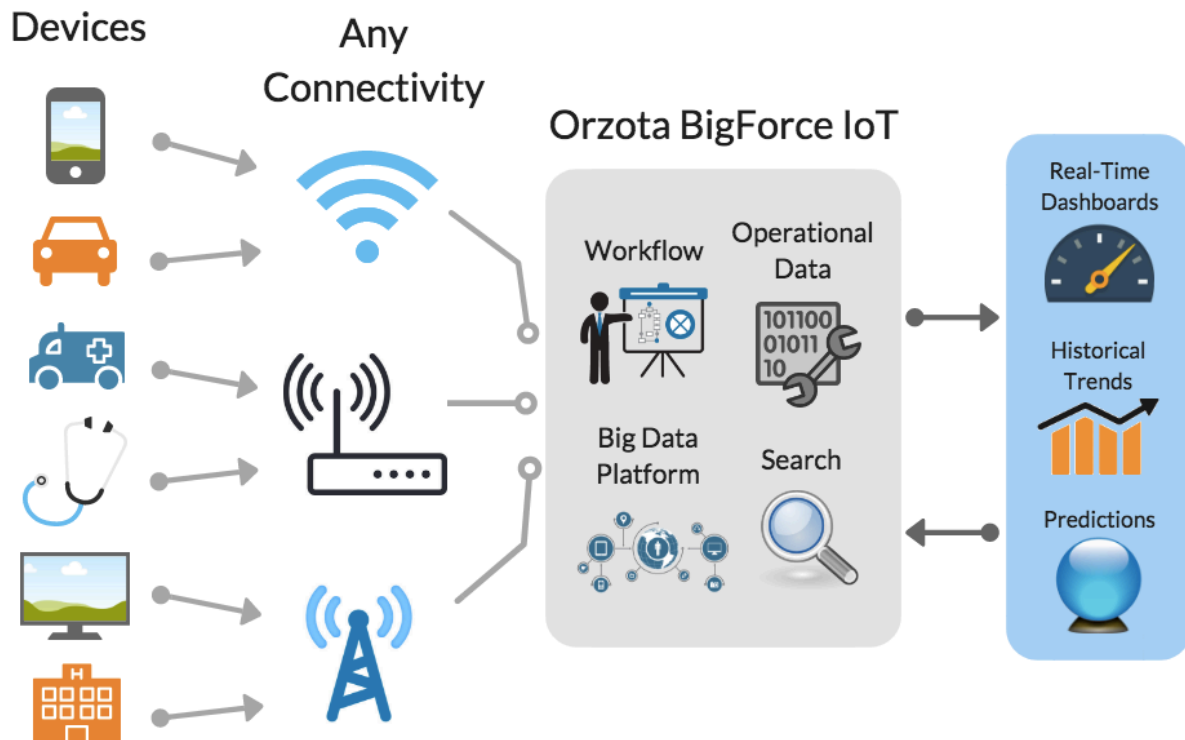
Data storage

A challenge for producers of IoT applications is to clean, process and interpret the vast amount of data which is gathered by the sensors. There is a solution proposed for the analytics of the information referred to as Wireless Sensor Networks. These networks share data among sensor nodes that are sent to a distributed system for the analytics of the sensory data. Another challenge is the storage of this bulk data. Depending on the application there could be high data acquisition requirements which in turn lead to high storage requirements. Currently the internet is already responsible for 5% of the total energy generated and this consumption will increase significantly when we start utilizing applications



CONCLUSION

Although IoT has quite a few disadvantages, its advantages of saving the consumer time and money can't be ignored. So the time isn't far when the Internet Of Things will be commonly seen in both households and companies. Helping in emergency situation and providing necessary data for conditions is the best advantage iot based device could provide. Efforts will have to be made to find ways to combat its disadvantages.



IOT

BASED

WEATHER

STATION

INTRODUCTION

Weather station is a facility, either on land or sea, with instruments and equipment for measuring atmospheric conditions to provide information for weather forecasts and to study the weather and climate. The measurements taken include temperature, atmospheric pressure, humidity, wind speed, wind direction, and precipitation amounts. Wind measurements are taken with as few other obstructions as possible, while temperature and humidity measurements are kept free from direct solar radiation, or insolation. Manual observations are taken at least once daily, while automated measurements are taken at least once an hour. Weather conditions out at sea are taken by ships and buoys, which measure slightly different meteorological quantities such as sea surface temperature (SST), wave height, and wave period. Drifting weather buoys outnumber their moored versions

Weather Station :

A Weather station able to collect data related to the environment and weather using many different sensors.



Using weather station , we can measure

- Temperature
- Humidity
- Rain
- gases percentage in air
- UV index
- Wind
- Pressure – Barometric

Objective:

The primary goals of this project to measure the temperature inside the room and then display this information on a server. We have used Things Speak as a server.

COMPONENTS REQUIRED

Hardware Components

- NodeMCU
- LM35 Temperature Sensor
- Bread Board
- Jumper Wires
- Micro USB Cable
- DHT11 Temperature and Humidity sensor
- laptop

Software Components

- Arduino IDE
- Windows 10

DESCRIPTION OF HARDWARE COMPONENTS

NODE MCU



NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware, it is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266 pin single row pin package. Convenient connection and special packages can be provided according to users need.

History

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core widely used in IoT applications (see related projects). NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R

committed the gerber file of an ESP8266 board, named devkit v0.9 Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform, and committed to NodeMCU project, then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to NodeMCU project enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays.

In summer 2015 the creators abandoned the firmware project and a group of independent but dedicated contributors took over. By summer 2016 the NodeMCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs.

Pins of NodeMCU

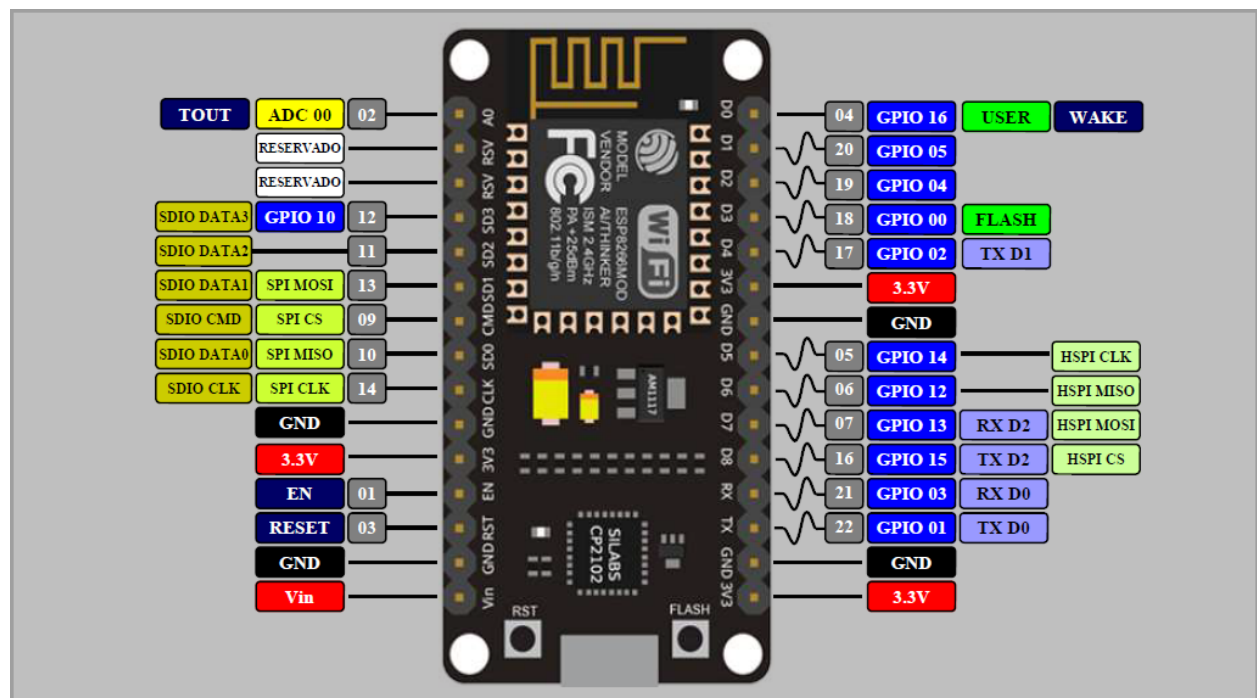
NodeMCU provides access to the GPIO (General Purpose Input/Output) and for developing purposes below pin mapping table from the API documentation should be referenced.

IO index	ESP8266 pin	IO index	ESP8266 pin
0 [*]	GPIO16	7	GPIO13
1	GPIO5	8	GPIO15
2	GPIO4	9	GPIO3

3	GPIO0	10	GPIO1
4	GPIO2	11	GPIO9
5	GPIO14	12	GPIO10
6	GPIO12		

[*] D0 (GPIO16) can only be used for GPIO read/write

PICTORIAL REPRESENTATION OF THE PINS



CODE

Code examples

The NodeMCU repository contains its own collection of elaborate code examples. Besides that the NodeMCU documentation provides small examples for most functions and modules.

Connect to an AP

```
print(wifi.sta.getip())
--nil
wifi.setmode(wifi.STATION)
wifi.sta.config{ssid="SSID",pwd="password"}
-- for older versions of the firmware wifi.sta.config("SSID","password")
-- wifi.sta.connect() not necessary because wifi.sta.config sets auto-connect = true
tmr.create():alarm(1000, 1, function(cb_timer)
  if wifi.sta.getip() == nil then
    print("Connecting...")
  else
    cb_timer.unregister()
    print("Connected, IP is "..wifi.sta.getip())
  end
end)
```

Control GPIO

```
ledPin = 1
swPin = 2
```



```
gpio.mode(ledPin,gpio.OUTPUT)
gpio.write(ledPin,gpio.HIGH)
gpio.mode(swPin,gpio.INPUT)
print(gpio.read(swPin))
```

HTTP request

```
-- A simple HTTP client
conn = net.createConnection(net.TCP, 0)
conn:on("receive", function(sck, payload) print(payload) end)
conn:on("connection", function(sck)
    sck:send("GET / HTTP/1.1\r\nHost: nodemcu.com\r\n"
        .. "Connection: keep-alive\r\nAccept: */*\r\n\r\n")
end)
conn:connect(80, "nodemcu.com")
```

Doing something similar using the HTTP module:

```
http.get("http://nodemcu.com", nil, function(code, data)
    if (code < 0) then
        print("HTTP request failed")
    else
        print(code, data)
    end
end)
```

HTTP server

```
-- a simple HTTP server
srv = net.createServer(net.TCP)
```

```

srv:listen(80, function(conn)
  conn:on("receive", function(sck, payload)
    print(payload)
    sck:send("HTTP/1.0 200 OK\r\nContent-Type: text/html\r\n\r\n<h1> Hello,
NodeMCU.</h1>")
  end)
  conn:on("sent", function(sck) sck:close() end)
end)

```

Connect to MQTT Broke

```

-- init mqtt client with keepalive timer 120sec
m = mqtt.Client("clientid", 120, "user", "password")

-- setup Last Will and Testament (optional)
-- Broker will publish a message with qos = 0, retain = 0, data = "offline"
-- to topic "/lwt" if client don't send keepalive packet
m:lwt("/lwt", "offline", 0, 0)

m:on("connect", function(con) print ("connected") end)
m:on("offline", function(con) print ("offline") end)

-- on publish message receive event
m:on("message", function(conn, topic, data)
  print(topic .. ":" )
  if data ~= nil then
    print(data)
  end

```

```

end
end)

-- for secure: m:connect("192.168.11.118", 1880, 1)
m:connect("192.168.11.118", 1880, 0, function(conn) print("connected") end)

-- subscribe topic with qos = 0
m:subscribe("/topic",0, function(conn) print("subscribe success") end)
-- or subscribe multiple topic (topic/0, qos = 0; topic/1, qos = 1; topic2 , qos = 2)
-- m:subscribe({["topic/0"]=0,["topic/1"]=1,topic2=2}, function(conn)
print("subscribe success") end)
-- publish a message with data = hello, QoS = 0, retain = 0
m:publish("/topic","hello",0,0, function(conn) print("sent") end)

m:close();
-- you can call m:connect again

```

UDP client and server

```

-- a udp server
s=net.createServer(net.UDP)
s:on("receive",function(s,c) print(c) end)
s:listen(5683)

-- a udp client
cu=net.createConnection(net.UDP)

```

```
cu:on("receive",function(cu,c) print(c) end)
cu:connect(5683,"192.168.18.101")
cu:send("hello")
```

ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your **WiFi** network.

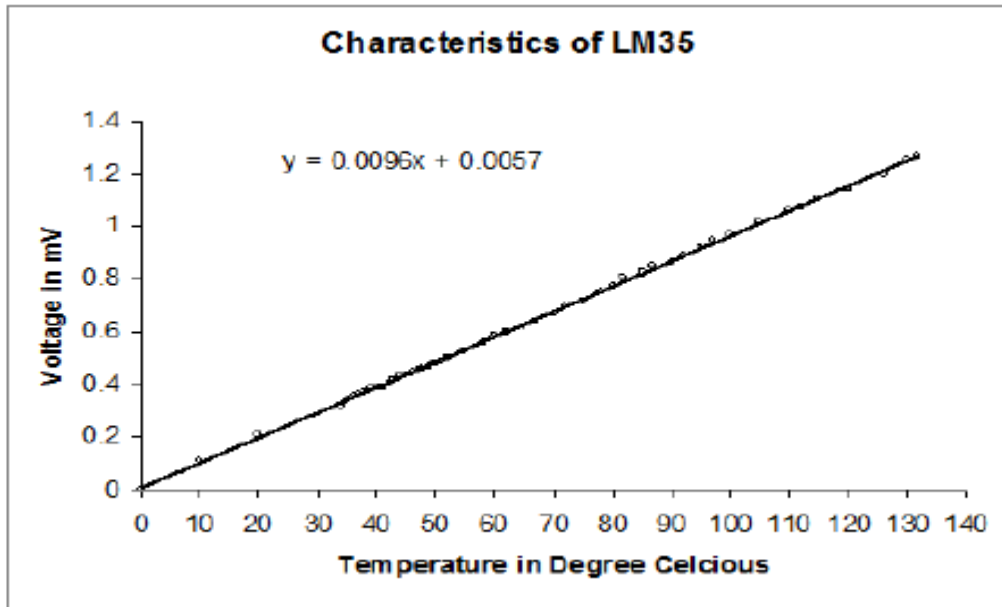
The **ESP8266** is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

LM35

The LM35 is an integrated circuit sensor that can be used to measure temperature with an electrical output proportional to the temperature (in °C). It can measure temperature more accurately than using a thermistor. The sensor circuitry is sealed and not subject to oxidation. The LM35 generates a higher output voltage than thermocouples and may not require that the output voltage be amplified. The LM35 has an output voltage that is proportional to the Celsius temperature. The scale factor is .01V/°C.

The LM35 does not require any external calibration or trimming and maintains an accuracy of $\pm 0.4^{\circ}\text{C}$ at room temperature and $\pm 0.8^{\circ}\text{C}$ over a range of 0°C to $+100^{\circ}\text{C}$. Another important characteristic of the LM35 is that it draws only 60 micro amps from its supply and possesses a low self-heating capability. The LM35 comes in many different packages such as TO-92 plastic transistor-like package, TO-46 metal can transistor-like package, 8-lead surface mount SO-8 small outline package.

- ## Features of LM35 Temperature Sensor



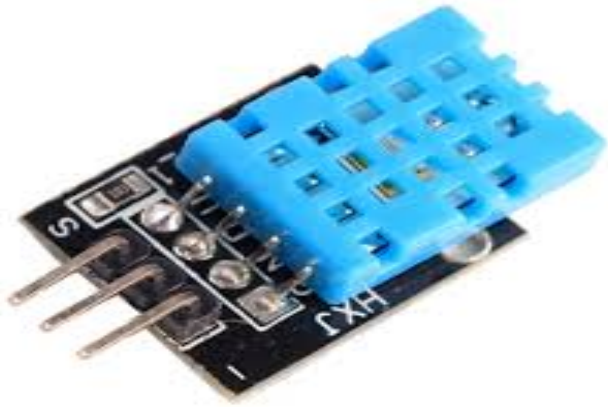
The output voltage of LM35 is proportional to its temperature.

DHT11 TEMPERATURE AND HUMIDITY SENSOR

Humidity sensors are the ones which have the ability to detect the relative humidity of the immediate environments in which they are placed. They have the capability both the moisture and temperature in the air. It can express relative humidity as a percentage of the ratio of moisture in the air to the maximum amount that can be held in the air at the current temperature. As air becomes hotter, it holds more moisture, so the relative humidity changes with the temperature.

Most humidity sensors use capacitive measurement to determine the amount of moisture in the air. This type of measurement relies on two electrical conductors with a non-conductive polymer film laying between them to create an electrical

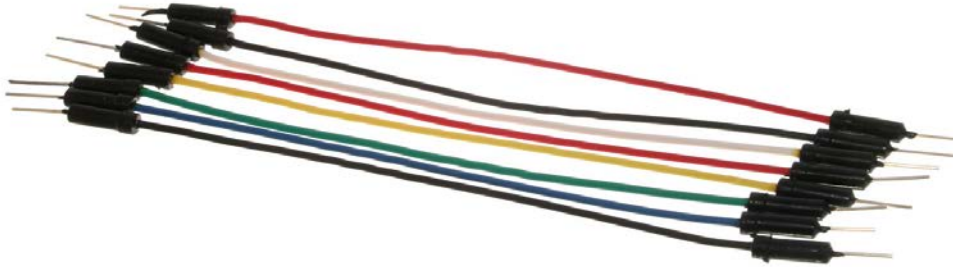
field between them. Moisture from the air collects on the film and causes changes in the voltage levels between the two plates. This change is then converted into a digital measurement of the air's relative humidity after taking the air temperature into account.



The above image shows the 3 pin dht11 sensor which we have used in our project.

JUMPER WIRES

A **jumper wire** (also known as jumper, jumper wire, jumper cable, [DuPont](#) wire, or DuPont cable – named for one manufacturer of them) is an [electrical wire](#), or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a [breadboard](#) or other prototype or test circuit, internally or with other equipment or components, without soldering.



We have used both male to male and male to female jumper wires in our circuit connection between dsp11 and esp8266 wifi module. Jumper wires are easier to work with than the single strand wires so it made our work easier.

ARDUINO IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux). It is written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It includes a code editor with features such as text cutting and pasting,

searching and replacing text,

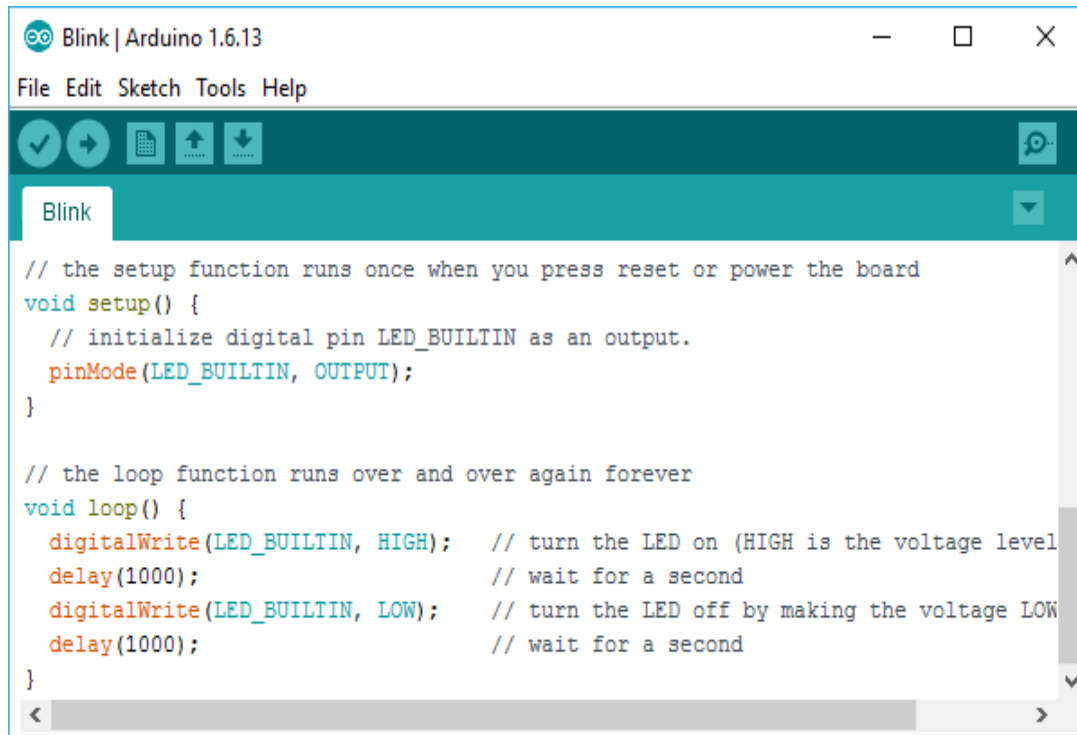
automatic indenting,

brace matching,

syntax highlighting

It also provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

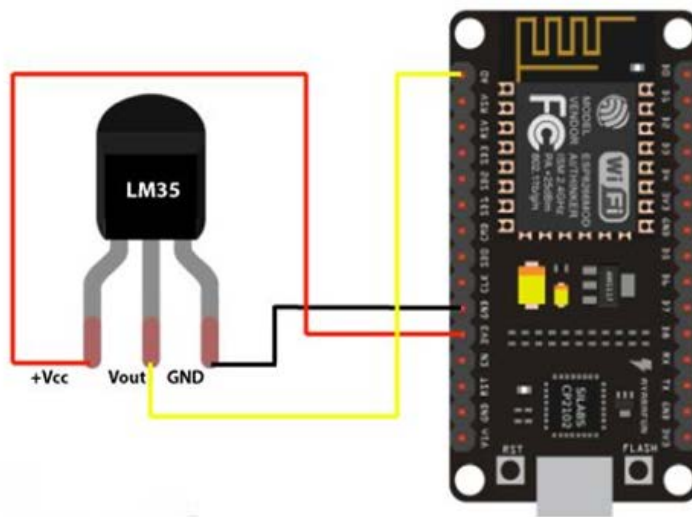
A screenshot of the Arduino IDE window titled "Blink | Arduino 1.6.13". The window has a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening, saving, compiling, and uploading. The main area shows the "Blink" sketch with the following code:

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

CIRCUIT CONNECTIONS

CIRCUIT DIAGRAM(WITH LM35)



The **circuit connections** are made as follows:

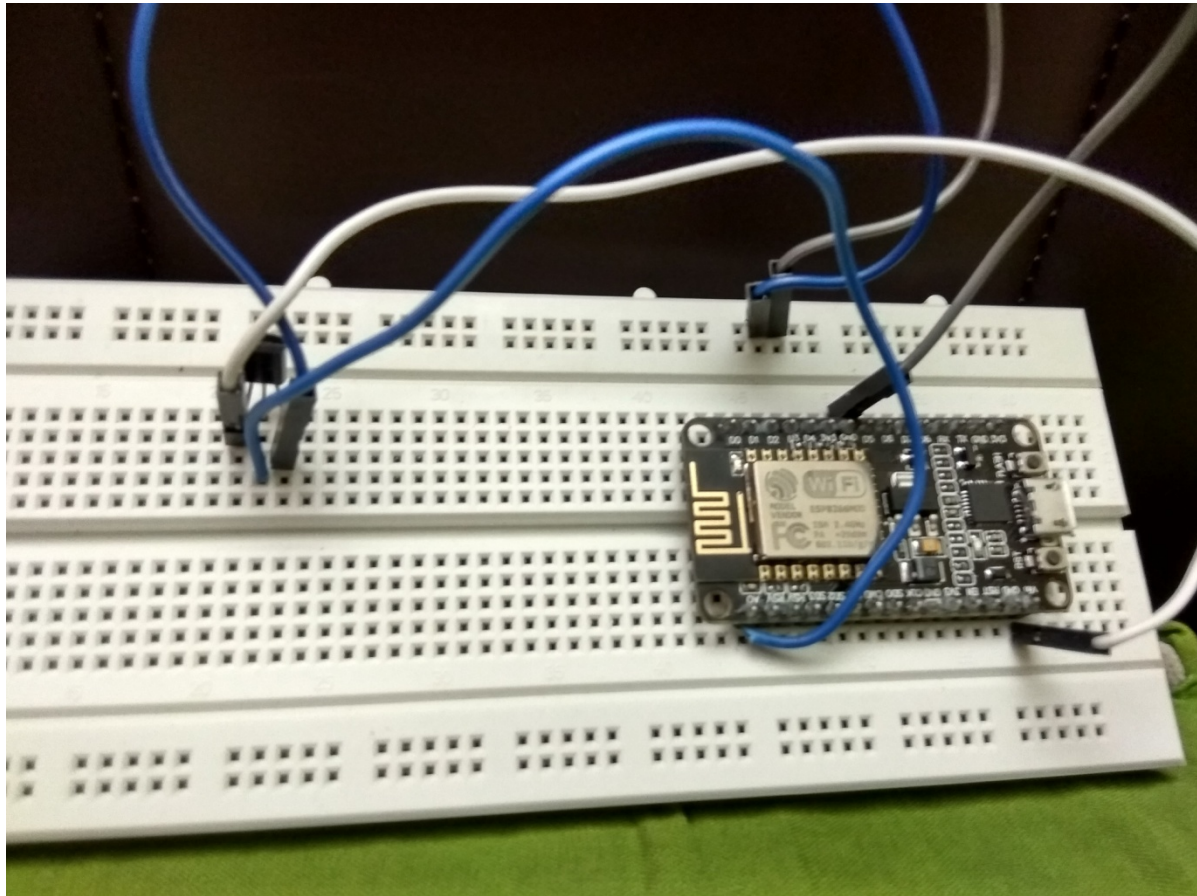
Pin 1 of the LM35 goes into **+3v** of the NodeMCU.

Pin 2 of the LM35 goes into Analog Pin **A0** of the NodeMCU.

Pin 3 of the LM35 goes into Ground Pin (**GND**) of the NodeMCU.

Before we get started with coding we need Arduino IDE.

CONNECTION IN BREADBOARD



THIS IS HOW WE HAVE CONNECTED THE LM35 TEMPERATURE SENSOR AND THE NODE MCU.WE HAVE FOLLOWED THE PIN CONFIGURATIONS STATED ABOVE.

WORKING CODE

(with LM35 for temperature)

```
#include <Adafruit_Sensor.h>
```

```
#include <DHTesp.h>
```

```
#include <DHT.h>
```

```
#include <DHT_U.h>
```

```
#include <ThingSpeak.h>
```

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

```
// replace with your channel's thingspeak API key,
```

```
String apiKey = "QJIAZMSDK4CUUOHO";
```

```
const char* ssid = "Redmi";
```

```
const char* password = "redmi123";
```

```
const char* server = "api.thingspeak.com";
```

```
#define DHTPIN 4// D2 pin on Nodemcu
```

```
DHT dht(DHTPIN, DHT11, 11);
```

```
WiFiClient client;
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
    delay(10);
```

```
    dht.begin();
```

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

```
    Serial.println();
```

```
    Serial.println();
```

```
    Serial.print("Connecting to ");
```

```
    Serial.println(ssid);
```

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

```
    while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);

    Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

}


void loop() {

    float h = dht.readHumidity();

    float t = dht.readTemperature(true);

    if (isnan(h) || isnan(t)) {

        Serial.println("Failed to read from DHT sensor!");

        return;

    }

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

        String postStr = apiKey;
```

```
postStr += "&field1=";  
  
postStr += String((int)t);  
  
postStr += "&field2=";  
  
postStr += String((int)h);
```

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

```
Serial.print("Temperature: ");  
  
Serial.print(t);  
  
Serial.print(" degrees Celcius Humidity: ");
```

```
    Serial.print(h);

    Serial.println("% send to Thingspeak");

}

client.stop();

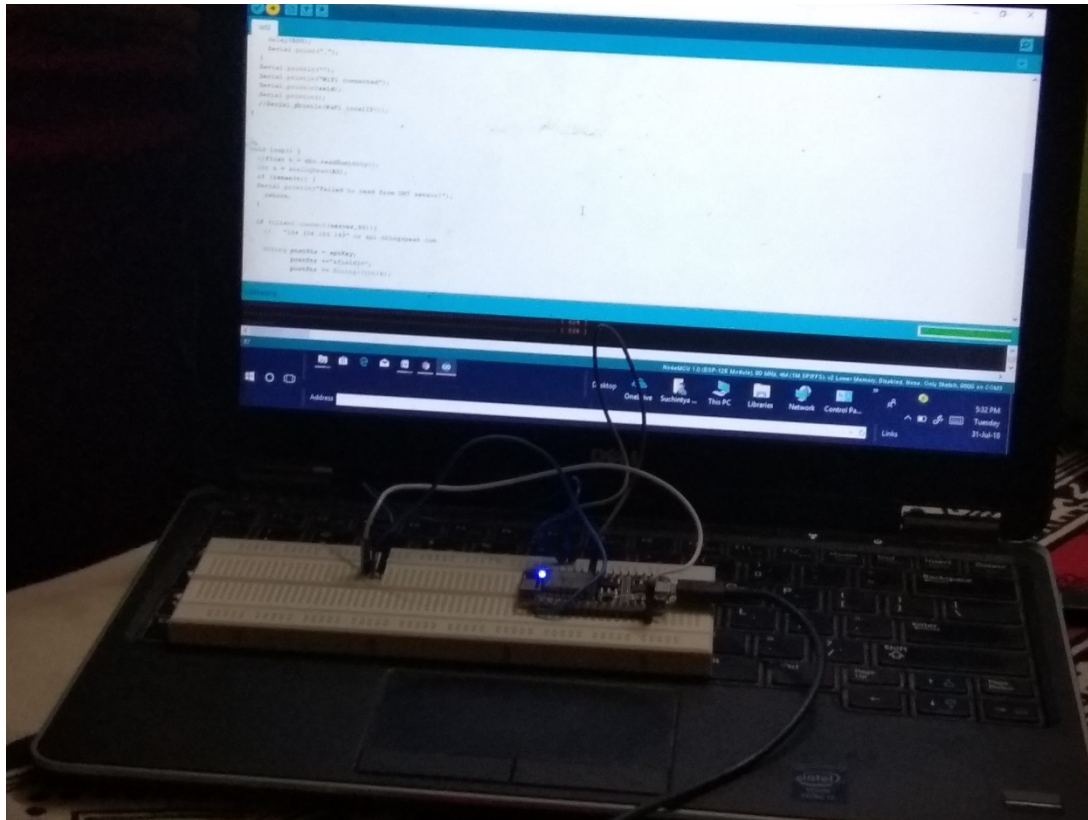

Serial.println("Waiting...");

// time between updates

delay(120000); // 2 mins

}
```

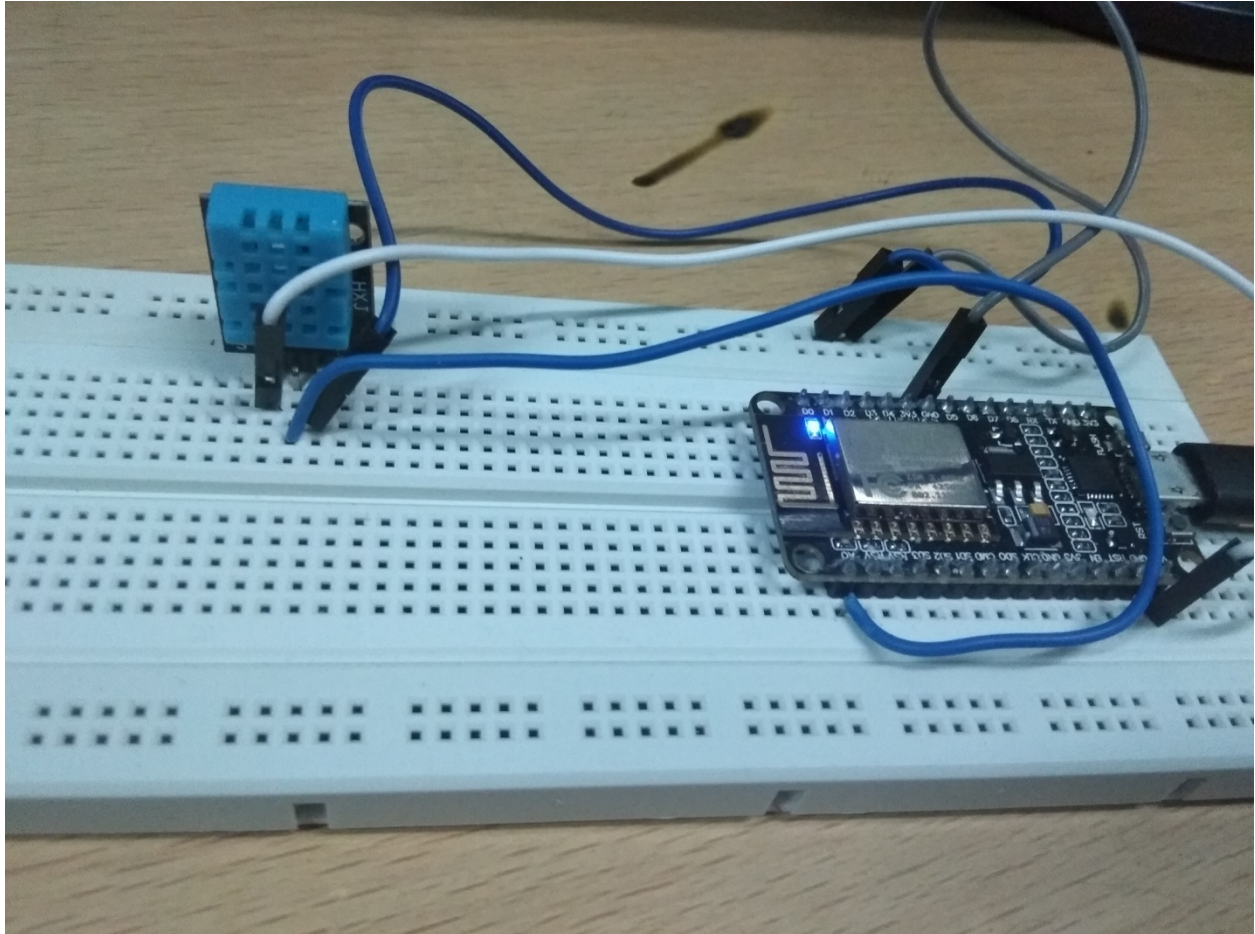

UPLOADING THE CODE TO ESP8266 NODE MCU



After successful compilation of the code we upload it to our wifi module .there is a blue flash which blinks while we upload it.after the uploading process gets completed the blue flash turns off and we go to serial monitor to see our output.



CIRCUIT DIAGRAM WITH DHT11 SENSOR



The **wiring connections** are made as follows :

Pin 1 of the DHT11 goes into **+3v** of the NodeMCU.

Pin 2 of the DHT11 goes into ANALOG PIN A0 of the NodeMCU.

Pin 3 of the DHT11 goes into Ground Pin (**GND**) of the NodeMCU.

AFTER SUCCESSFUL COMPLETION OF THE ABOVE CIRCUIT DIAGRAM WE COPILE THE CODE IN ARDUINO IDE AND UPLOAD IT AND THEN FOLLOW THE NECESSARY STEPS.

WORKING CODE WITH DHT11 FOR TEMPERATURE AND HUMIDITY

```
#include <Adafruit_Sensor.h>
```

```
#include <DHT.h>
```

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

```
// replace with your channel's thingspeak API key,
```

```
String apiKey = "QJIAZMSDK4CUUOHO";
```

```
const char* ssid = "Redmi"
```

```
const char* password = "redmi123";
```

```
const char* server = "api.thingspeak.com";
```

```
#define DHTPIN 2
```

```
DHT dht(DHTPIN, DHT11,15);
```

```
WiFiClient client;
```

```
void setup() {
```

```
  Serial.begin(9600);
```

```
  delay(10);
```

```
  dht.begin();
```

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

```
  Serial.println();
```

```
  Serial.println();
```

```
  Serial.print("Connecting to ");
```

```
  Serial.println(ssid);
```

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

```
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
```

```
    Serial.print(".");
```

```
  }
```

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

```
  Serial.println("WiFi connected");
```

```
}
```

```
void loop() {
```

```
float h = dht.readHumidity();  
float t = dht.readTemperature();  
if (isnan(h) || isnan(t)) {  
  Serial.println("Failed to read from DHT sensor!");  
  return;  
}
```

```
if (client.connect(server,80)) {  
  String postStr = apiKey;  
  postStr += "&field1=";  
  postStr += String(t);  
  postStr += "&field2=";  
  postStr += String(h);  
  postStr += "\r\n\r\n";
```

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

```
  Serial.print("Temperature: ");
```

```
Serial.print(t);  
Serial.print(" degrees Celcius Humidity: ");  
Serial.print(h);  
Serial.println("% send to Thingspeak");  
}  
client.stop();  
  
Serial.println("Waiting...");  
delay(20000);  
}
```

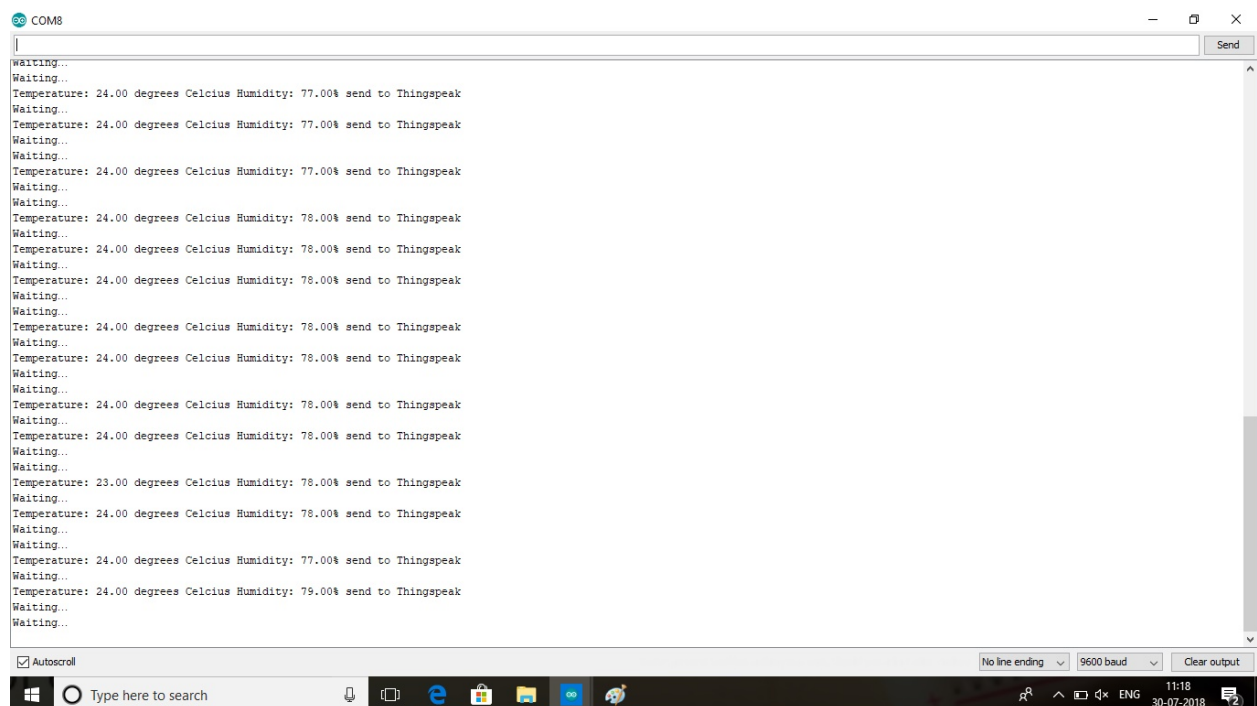
REGISTERING WITH THINGSPEAK

Getting API Key

1. Go to <https://thingspeak.com/> and create an account if you do not have one. Login to your account.
2. Create a new channel by clicking on the button.**Enter basic details of the channel.**Then **Scroll down and save the channel.**
3. Channel Id is the identity of your channel. Note down this. Then go to API keys copy and paste this key to a separate notepad file will need it later.

After successful compilation of the code we upload it to the node mcu using the USB CABLE.It will take some time and after that we go to the serial monitor and check for the temperature details.Then we go to our thingspeak account and login to it and find the graphs.

OUTPUT



```
COM8
Waiting...
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 77.00% send to Thingspeak
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 77.00% send to Thingspeak
Waiting...
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 77.00% send to Thingspeak
Waiting...
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 78.00% send to Thingspeak
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 78.00% send to Thingspeak
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 78.00% send to Thingspeak
Waiting...
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 78.00% send to Thingspeak
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 78.00% send to Thingspeak
Waiting...
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 78.00% send to Thingspeak
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 78.00% send to Thingspeak
Waiting...
Waiting...
Temperature: 23.00 degrees Celcius Humidity: 78.00% send to Thingspeak
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 78.00% send to Thingspeak
Waiting...
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 77.00% send to Thingspeak
Waiting...
Temperature: 24.00 degrees Celcius Humidity: 79.00% send to Thingspeak
Waiting...
Waiting...
```

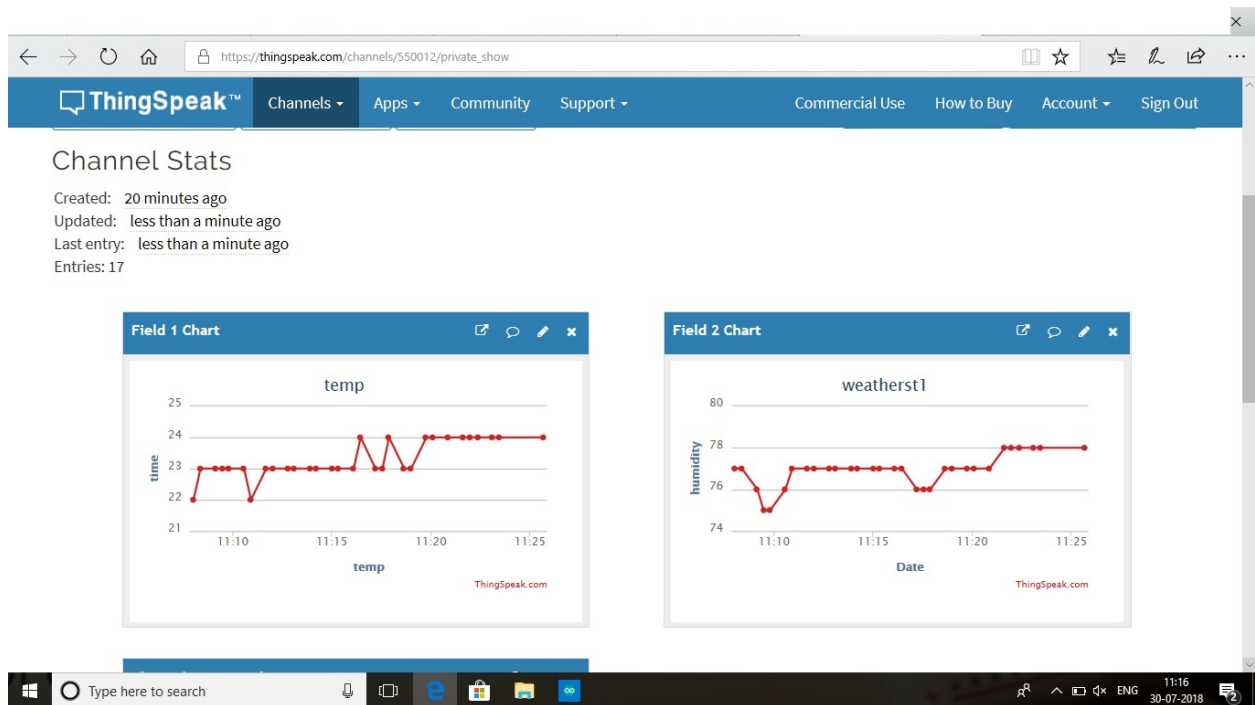
Autoscroll No line ending 9600 baud Clear output

Type here to search

11:18 30-07-2018

HERE WE CAN SEE THE TEMPERATURE AND RELATIVE HUMIDITY BEING DISPALYED ON THINGS SPEAK SERVER WHICH MAKES THE SUCCESSFUL COMPLETION OF THE FULL PROJECT.

THINGSPEAK GRAPHS



THE GRAPHS ARE ALSO ATTACHED ABOVE WHICH THINGSPEAK PROVIDES US. WITH THE HELP OF THINGSPEAK SERVER WE GET THE PERFECT OUTPUT AND GRAPHS.

CONCLUSION

The article presented here is based on the project work carried by us. This project is an implementation of Internet of Things. The said project is able to sense the temperature and monitor it remotely. The proposed use of Internet of Things will help the researchers in the field to come up with solutions that are inexpensive and more reliable. The project can be further expanded to control things. The sensed temperature can be used for various projects such as home automation, lab monitoring, etc. We have done our project both by using lm35 and dht11. lm35 measures the temperature and dht11 measures both the temperature and humidity.

REFERENCES

1. ^ Jump up to:^a ^b Brown, Eric (13 September 2016). "Who Needs the Internet of Things?". *Linux.com*. Retrieved 23 October 2016.
2. ^ Jump up to:^a ^b Brown, Eric (20 September 2016). "21 Open Source Projects for IoT". *Linux.com*. Retrieved 23 October 2016.
- 3.www.wikipedia.org
- 4.[instructables](http://instructables.com)