*ABSTRACT:*

With advancement of Automation technology, life is getting simpler and easier in all aspects. In today’s world Automatic systems are being preferred over manual system. With the rapid increase in the number of users of internet over the past decade has made Internet a part and parcel of life, and IoT is the latest and emerging internet technology. Internet of things is a growing network of everyday object-from industrial machine to consumer goods that can share information and complete tasks while you are busy with other activities. Wireless Home Automation system(WHAS) using IoT is a system that uses computers or mobile devices to control basic home functions and features automatically through internet from anywhere around the world, an automated home is sometimes called a smart home. It is meant to save the electric power and human energy. The home automation system differs from other system by allowing the user to operate the system from anywhere around the world through internet connection.CAN WE CONTROL OUR ELECTRICAL DEVICES THROUGH AN APP OR A WEBSITE? The answer is an obvious YES!

Our project provides an home automated system by web based that employs the integration of IOT and web development, to provide user with a webpage to control the electrical devices through the options provided, that he/she desire to have. This system is designed to be low cost and expandable allowing a variety of devices to be controlled.

**Introduction:**

 Nowadays, we have remote controls for our television sets and other electronic systems, which have made our lives real easy. Have you ever wondered about home automation which would give the facility of controlling tube lights, fans and other electrical appliances at home using a remote control? Off-course, Yes! But, are the available options cost-effective? If the answer is No, we have found a solution to it. We have come up with a new system called Arduino based home automation. This system is super-cost effective and can give the user, the ability to control any electronic device without even spending for a remote control. This project helps the user to control all the electronic devices using his/her smartphone. Time is a very valuable thing. Everybody wants to save time as much as they can. New technologies are being introduced to save our time. To save people’s time we are introducing Home Automation system. With the help of this system you can control your home appliances from your mobile phone.

We are creating this application instead of any remote for automation because every person have their personal android phone and they can easily control their home appliances, and sometimes remote device is misplaced or destroyed. Another reason is that remote device is hard to carry when you are outside from home but your personal phone is always with you.

What is homo automation?

*Home automation or domoticsis building automation for a home, called a smart home or smart house. A home automation system will control lighting, climate, entertainment systems, and appliances. It may also include home security such as access control and alarm systems.*[*[2]*](https://en.wikipedia.org/wiki/Home_automation#cite_note-2)*When connected with the Internet, home devices are an important constituent of the Internet of Things..*

Homo automation through web page

*IOT based Web Controlled Home Automation using Raspberry Pi. ... Using this IoT based home automation system, you can control your Home appliances from anywhere in the world. This web server can be run from any device which can run HTML applications, like Smart Phone, tablet, computer etc.*

Languages used to create web page :

C/C++

Css

HTML

Components used to create :

Nodemcu

Breadboard

Connecting wires

Led bulbs

Software used:

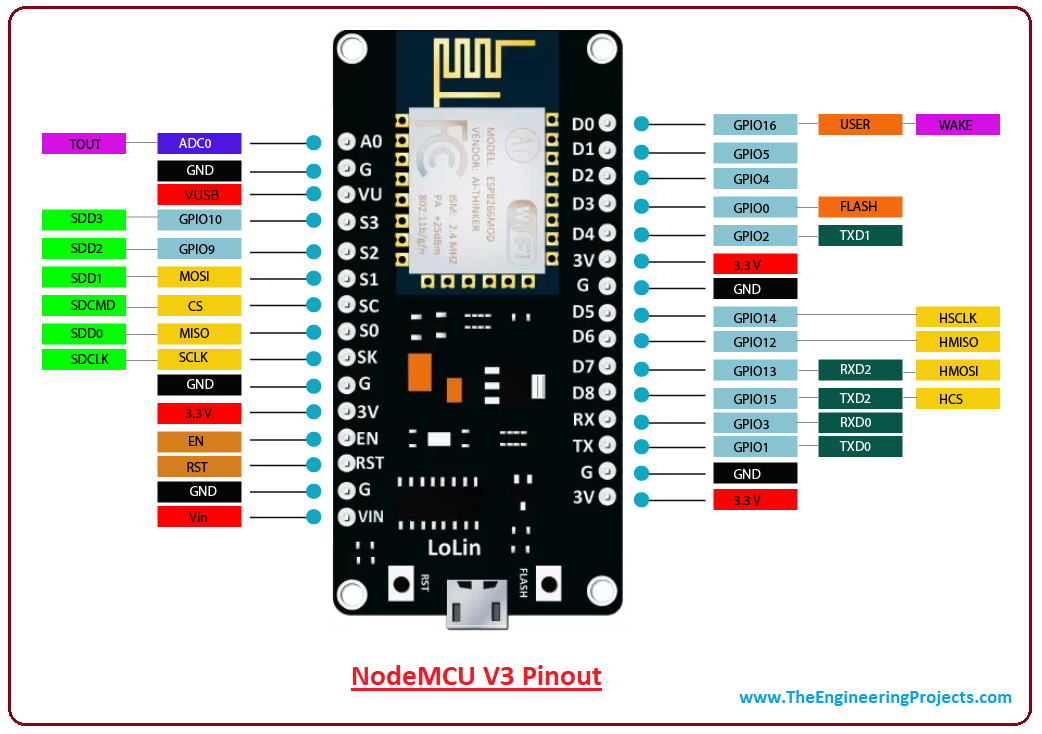
Arduinio

Sublime text editor

Thingspeak

Nodemcu:

NodeMCU is an open source LUA based firmware developed for ESP8266 wifi chip,NodeMCU firmware comes with ESP8266 development kit i.e. NodeMCUDevelopment board.The term “NodeMCU” by default refers to the firmfae rather than the developments 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.It uses many open source project,such as lua-cjson and SPIFFS.



## *The above diagram is a pin diagram of nodemcu*

PINS:

|  |  |
| --- | --- |
| **I/O index** | **ESP8266 pin** |
| 0 [\*] | GPIO16 |
| 1 | GPIO5 |
| 2 | GPIO4 |
| 3 | GPIO0 |
| 4 | GPIO2 |
| 5 | GPIO14 |
| 6 | GPIO12 |
| 7 | GPIO13 |
| 8 | GPIO15 |
| 9 | GPIO3 |
| 10 | GPIO1 |
| 11 | GPIO9 |
| 12 | GPIO10 |

**ESP8266**

The ESP8266 is a low-cost wifi microchip with full Tcp/Ip stack and *microcontro*ller capability.This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using *Hayes* style commands. However, at first there was almost no English-language documentation on the chip and the commands it accepted.[*[2]*](https://en.wikipedia.org/wiki/ESP8266#cite_note-2) The very low price and the fact that there were very few external components on the module, which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.[*[3]*](https://en.wikipedia.org/wiki/ESP8266#cite_note-3)

The ESP8285 is an ESP8266 with 1 MiB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi.

Pinout :

VCC, Voltage (+3.3 V; can handle up to 3.6 V)

GND, Ground (0 V)

RX, Receive data bit X

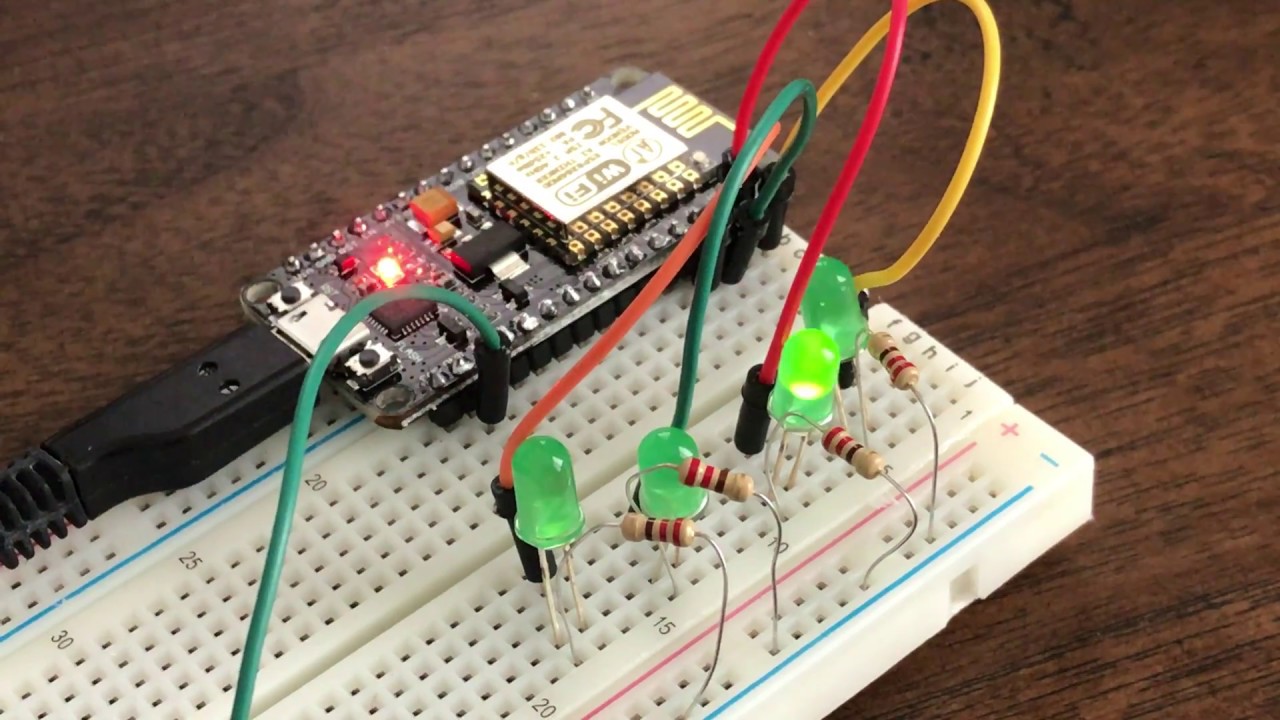
TX, Transmit data bit X

CH\_PD, Chip power-down

RST, Reset

GPIO 0, General-purpose input/output No. 0

GPIO 2, General-purpose input/output No. 2



Connection:

One end of LED’s are connected to 16,14,12,5 pins of ESP8266

Other end are connected to gnd

HTML:

Hyper text markup language is the standard language for documents desingned to be displayed in a web browser.It can be assited by technologies such as cascading style sheets and scripting language suh as javascript.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets Tags such as <**img** /> and <**input** /> directly introduce content into the page. Other tags such as <**p**> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

We used html to create web page which has buttons to turn off/on lights from any where.

CSS: cascading style sheets is language used for describing the presentation of document written in markup language like HTML.CSS is cornerstone technology of the world wide web,alongside HTML and Javascript.

CSS is designed to enable the separation of presentation and content, including layout, colors and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Thingspeak:

*ThingSpeak is an*[***open-source***](https://en.wikipedia.org/wiki/Open-source_software)[***Internet of Things***](https://en.wikipedia.org/wiki/Internet_of_Things)*(IoT) application and*[***API***](https://en.wikipedia.org/wiki/API)*to store and retrieve data from things using the*[***HTTP***](https://en.wikipedia.org/wiki/HTTP)*protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates".*



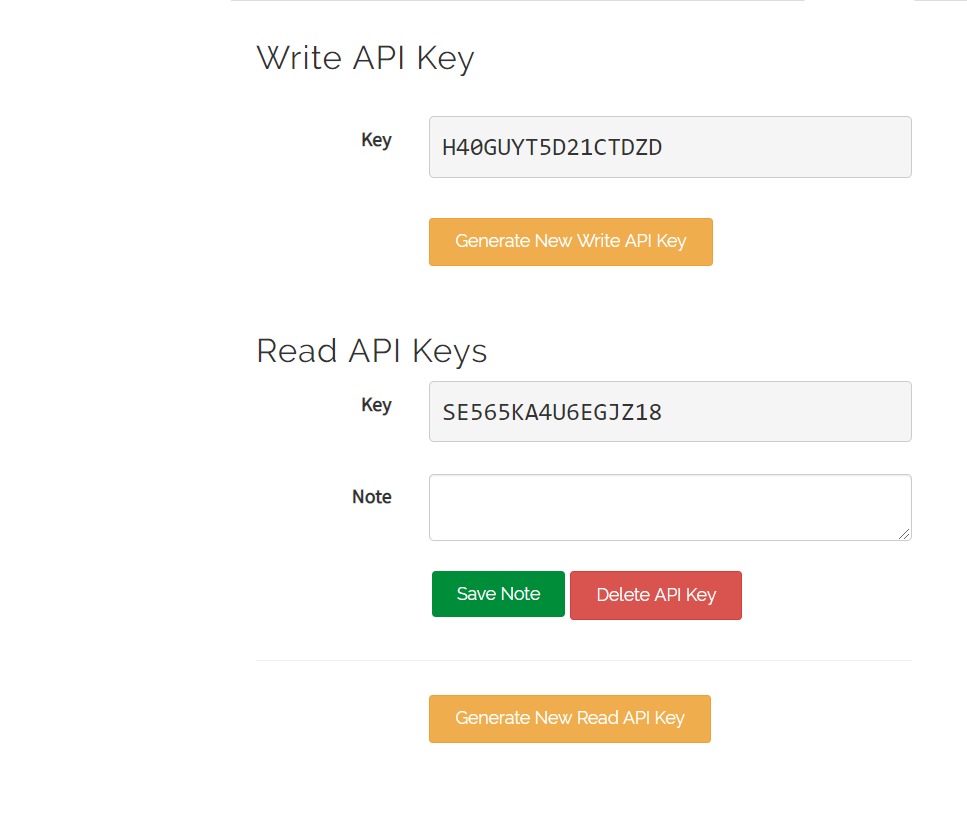
Connecting thingspeak to ESP8266:

*Create an account in thingspeak*

Connect and LED to pin

Step 2: Read API Key

in addition to things you remembered from the last parts, this time you need to remember your **Read API Key**, because you are willing to read data from a channel field which are ( 0 and 1 )'s

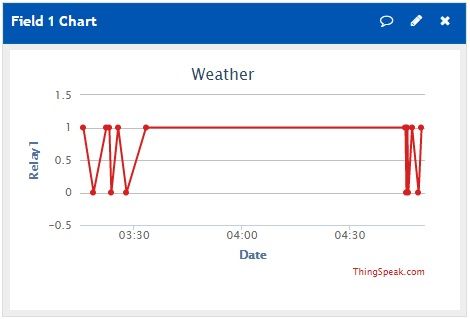


Step 3: Android Mobile Application

Download this free application [ESP8266 IoT Control Devices](https://play.google.com/store/apps/details?id=esp8266.iot.control)**OR**this "[Relay Control Internet](https://play.google.com/store/apps/details?id=appinventor.ai_Muhandi_prasetyo.Reconnet&hl=en)". This application is very simple, all you need is to set your **Channel ID** and **Field.**Unfortunately, your are only allowed to control two device ( actually you will send data to maximum two channel fields).

Note that you need to set your **Write API Key** in the Android Application and **Read API Key**in the Arduino code.

Step 4: Code

[](https://cdn.instructables.com/FFC/5551/IPQ4KPHP/FFC5551IPQ4KPHP.LARGE.jpg)

Download my source code.

Inside the program, you will find this code:

readValue =**ThingSpeak.readIntField**(myChannelNumber, 1, myReadAPIKey);

Number**1** stands for the field number. this ThingSpeaks's function returns the last value ( in our case **0**or **1**) which has already written from the mobile application

Create an inferface for webpage using html amd css.Code used by us for creating a webpage interface is given below;

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript" src="https://gc.kis.v2.scr.kaspersky-labs.com/FD126C42-EBFA-4E12-B309-BB3FDD723AC1/main.js" charset="UTF-8"></script><style>

body {

background-image:url("https://img.freepik.com/free-vector/spot-light-background\_1284-4685.jpg?size=626&ext=jpg") ;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

.button {

background-color: #4CAF50; /\* Green \*/

border: none;

color: white;

padding: 15px 32px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

}

.blue{

color:blue;

font-style: oblique;

}

.green{

color:#006400;

font-style: oblique;

}

.yellow{

color:yellow;

font-style: oblique;

}

h1 {

color: black;

text-align: center;

font-family: "Times New Roman", Times, serif;

font-style: italic;

font-weight: bold;

}

.red{

color:red;

font-style: oblique;

}

p {

font-family: verdana;

font-size: 20px;

}

</style>

</head>

<body>

<h1><u>HOME AUTOMATION</u></h1>

<h2 class="red"><u>RED LIGHT</u></h2>

<div class="b">

<center>

<a href ="https://api.thingspeak.com/update?api\_key=H40GUYT5D21CTDZD&field1=1"> <button class=button>Turn ON</button> </a>

<a href ="https://api.thingspeak.com/update?api\_key=H40GUYT5D21CTDZD&field1=0"> <button class=button> Turn OFF</button> </a>

</center>

</div>

<h2 class="blue"><u>BLUE LIGHT</u></h2>

<div>

<center>

<a href ="https://api.thingspeak.com/update?api\_key=H40GUYT5D21CTDZD&field2=1"> <button class=button>Turn ON</button> </a>

<a href ="https://api.thingspeak.com/update?api\_key=H40GUYT5D21CTDZD&field2=0"> <button class=button> Turn OFF</button> </a>

</center>

</div>

<h2 class="green"><u>GREEN LIGHT</u></h2>

<div>

<center>

<a href="https://api.thingspeak.com/update?api\_key=H40GUYT5D21CTDZD&field3=1"><button class=button>Turn On</button></a>

<a href="https://api.thingspeak.com/update?api\_key=H40GUYT5D21CTDZD&field3=0"><button class=button>Turn Off</button></a>

</center>

</div>

<h2 class="yellow"><u>YELLOW LIGHT</u></h2>

<div>

<center>

<a href=\"/1\"><button class=button>Turn On</button></a>

<a href=\"/0\"><button class=button>Turn Off</button></a>

</center>

</div>

</body>

</html>

Here the following syntax is used to send commans to thingspeak according to which the module acts

Syntax;

<a href ="https://api.thingspeak.com/update?api\_key=H40GUYT5D21CTDZD&field1=1"> <button class=button>Turn ON</button> </a>

When we click on the button Turn ON,href link is loaded which send 1 as input to the thinkspeak server.

Syntax;

<a href ="https://api.thingspeak.com/update?api\_key=H40GUYT5D21CTDZD&field1=0"> <button class=button>Turn OFF</button> </a>

When we click Turn OFF button,href link is loaded which sends 0 as input to the thingspeak server.

**Code for the Arduino used is given below, This code is used to design backend of the project;**

**#include "ThingSpeak.h"**

**#include <ESP8266WiFi.h> //Replace your wifi credentials here**

**const char\* ssid = "3vedh";**

**const char\* password = "tri12vedh"; //change your channel number here**

**unsigned long channel = 824364; //1,2 and 3 are channel fields. You don't need to change if you are following this tutorial. However, you can modify it according to your application**

**unsigned int led1 = 1;**

**unsigned int led2 = 2;**

**unsigned int led3 = 3;**

**WiFiClient client;**

**void setup() {**

**Serial.begin(115200);**

**delay(100);**

**pinMode(16, OUTPUT);**

**pinMode(14, OUTPUT);**

**pinMode(5, OUTPUT);**

**digitalWrite(16, 0);**

**digitalWrite(14, 0);**

**digitalWrite(5, 0); // We start by connecting to a WiFi network**

**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");**

**Serial.println("IP address: ");**

**Serial.println(WiFi.localIP());**

**Serial.print("Netmask: ");**

**Serial.println(WiFi.subnetMask());**

**Serial.print("Gateway: ");**

**Serial.println(WiFi.gatewayIP());**

**ThingSpeak.begin(client);**

**}**

**void loop() { //get the last data of the fields**

**int led\_1 = ThingSpeak.readFloatField(channel, led1);**

**int led\_2 = ThingSpeak.readFloatField(channel, led2);**

**int led\_3 = ThingSpeak.readFloatField(channel, led3);**

**if(led\_1 == 1){**

**digitalWrite(16, 1);**

**Serial.println("D1 is On..!");**

**}**

**else if(led\_1 == 0){**

**digitalWrite(16, 0);**

**Serial.println("D1 is Off..!");**

**}**

**if(led\_2 == 1){**

**digitalWrite(14, 1);**

**Serial.println("D2 is On..!");**

**}**

**else if(led\_2 == 0){**

**digitalWrite(14, 0);**

**Serial.println("D2 is Off..!");**

**}**

**if(led\_3 == 1){**

**digitalWrite(12, 1);**

**Serial.println("D3 is On..!");**

**}**

**else if(led\_3 == 0){**

**digitalWrite(12, 0);**

**Serial.println("D3 is Off..!"); }**

**Serial.println(led\_1);**

**Serial.println(led\_2);**

**Serial.println(led\_3);**

**delay(5000);**

**}**

**The following syntax is used to include thingspeak library;**

**Syntax;**

**#include "ThingSpeak.h"**

**The following logic is used to switch on or off a lead based upon the command imported from the thingspeak server**

**if(led\_1 == 1){**

**digitalWrite(16, 1);**

**Serial.println("D1 is On..!");**

**}**

**else if(led\_1 == 0){**

**digitalWrite(16, 0);**

**Serial.println("D1 is Off..!");**

**}**

**According to the following syntax if the module gets a command as 1:light glows**

**Else if the module gets the command as 0:light turns off;**

**In the following syntax channel number is assigned to the program;**

**Syntax;**

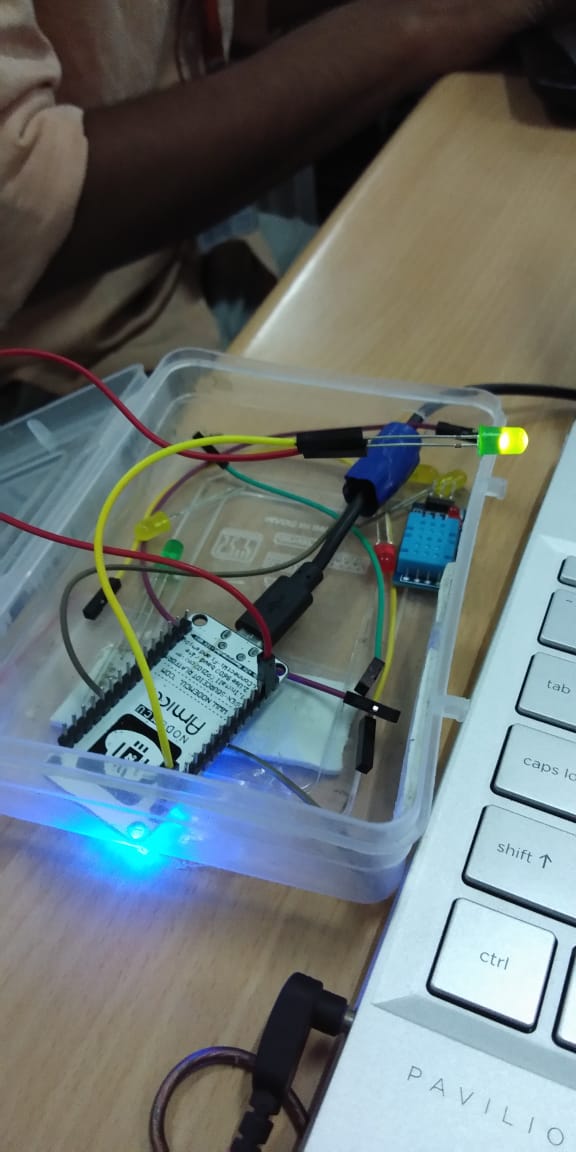
**unsigned long channel = 824364;**

**the following syntax is used to assign channel and their name;**

**int led\_1 = ThingSpeak.readFloatField(channel, led1);**

**int led\_2 = ThingSpeak.readFloatField(channel, led2);**

**int led\_3 = ThingSpeak.readFloatField(channel, led3);**



*CONCLUSION*:The aim of this project was to build a home automation system so that the householder could control all the electrical devices in his/her residence through a developed webpage. Our project provides user a congenial facility to have control on his/her electrical devices, through their mobile or computer using a webpage designed for controls. Altogether, this automation system has its fruitful advantages and depicts our spectacular advancement in emerging technology.

References:

W3schools.com

Youtube.com

Github.com