# Contents

# List of Tables

# List of Figures

# List of Acronyms

| | |
|---|---|
| ADC | Analog to Digital converter |
| CPU | Central Processing Unit |
| DHT11 | Digital Humidity Sensor |
| DMA | Direct Memory Access |
| GND | Ground |
| GPIO | General Purpose Input-Output |
| I2C | Inter-Integrated Circuit |
| I2S | Inter-Integrated Circuit |
| IDE | Integrated Development Environment |
| IFTTT | If This Then That |
| IoT | Internet of Things |
| IP | Internet Protocol |
| LED | Light Emitting Diode |
| MCU | Microcontroller Unit |
| PWM | Pulse-Width Modulation |
| RFID | Radio Frequency Identification |
| RISC | Reduced Instruction Set Computer |
| ROM | Read Only Memory |
| SPI | Serial Peripheral Interface |
| TCP | Transmission Control Protocol |
| UART | Universal Asynchronous Receiver/Transmitter |
| WPA | Wi-Fi Protected Access |

# *Chapter 1*

# *INTRODUCTION*

# *Chapter 1*

# INTRODUCTION

## 1.1 Introduction to the Internet of Things

**Internet of Things (IoT)** is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment. In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.

Over 9 billion 'Things' (physical objects) are currently connected to the Internet, as of now. In the near future, this number is expected to rise to a whopping 20 billion.

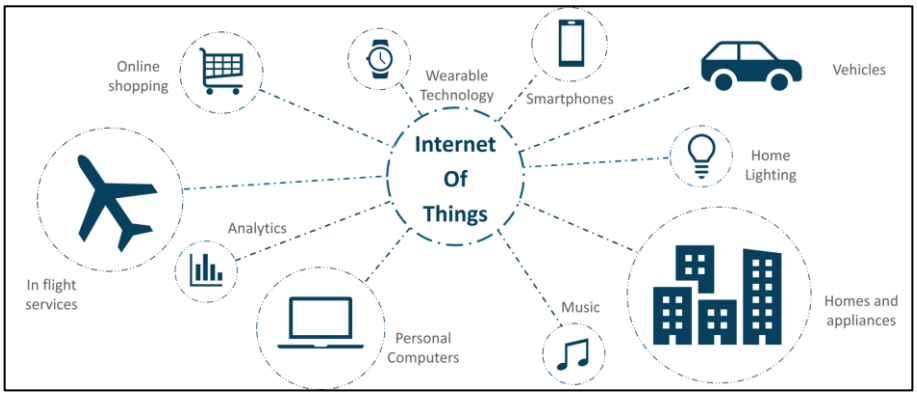



***Figure 1.1:*** Network of IoT

There are four main components used in IoT:

- **Low-power embedded systems –**
  Less battery consumption, high performance are the inverse factors that play a significant role during the design of electronic systems.
- **Cloud computing –**
  Data collected through IoT devices is massive and this data has to be stored on a reliable storage server. This is where cloud computing comes into play. The data is processed and learned, giving more room for us to discover where things like electrical faults/errors are within the system.
- **Availability of big data –**
  We know that IoT relies heavily on sensors, especially in real-time. As these electronic devices spread throughout every field, their usage is going to trigger a massive flux of big data.
- **Networking connection –**
  In order to communicate, internet connectivity is a must where each physical object is represented by an IP address. However, there are only a limited number of addresses available according to IP naming. Due to the growing number of devices, this naming system will not be feasible anymore. Therefore, researchers are looking for another alternative naming system to represent each physical object.

***There are two ways of building an IoT:***

1. Form a separate internetwork including only physical objects.
2. Make the Internet ever more expansive, but this requires hard-core technologies such as rigorous cloud computing and rapid big data storage (expensive).

### 1.1.1 IoT Enablers

- ***RFIDs:*** uses radio waves in order to electronically track the tags attached to each physical object.
- ***Sensors:*** devices that are able to detect changes in an environment (ex: motion detectors).
- ***Nanotechnology:*** as the name suggests, these are extremely small devices with dimensions usually less than a hundred nanometers.
- ***Smart networks:*** (ex: mesh topology).

### 1.1.2 Characteristics of IoT

- Massively scalable and efficient
- IP-based addressing will no longer be suitable in the upcoming future.
- An abundance of physical objects is present that does not use IP, so IoT is made possible.
- Devices typically consume less power. When not in use, they should be automatically programmed to sleep.
- A device that is connected to another device right now may not be connected in another instant of time.
- Intermittent connectivity – IoT devices aren't always connected. In order to save bandwidth and battery consumption, devices will be powered off periodically when not in use. Otherwise, connections might turn unreliable and thus prove to be inefficient.

### 1.1.3 Application Domain of IoT

IoT is currently found in four different popular domains:

- Manufacturing/Industrial business - 40.2%
- Healthcare - 30.3%
- Security - 7.7%
- Retail - 8.3%

***Modern Applications:***

- Smart Grids
- Smart cities
- Smart homes
- Healthcare
- Earthquake detection

- Radiation detection/hazardous gas detection
- Smartphone detection
- Water flow monitoring

## 1.2 Project Objectives

- To build a system to control the LED with a virtual button.
- To build a system that monitors the temperature of the room and displays on the screen.

# *Chapter 2*

# *HARDWARE REQUIREMENTS*

# *Chapter 2*
# HARDWARE REQUIREMENTS

## 2.1 NodeMCU (ESP8266)

***Figure 2.1:*** NodeMCU DEVKIT 1.0

| | |
|---|---|
| **Memory** | 128 KB |
| **Storage** | 4 MB |
| **Operating Voltage** | 3V - 3.6V |
| **Operating Current** | 80 mA |
| **Operating Frequency** | 80 MHz – 160 MHz |

***Table 2.1:*** NodeMCU Specifications

NodeMCU is an open-source LUA based firmware developed for ESP8266 Wi-Fi chip. By exploring functionality with ESP8266 chip, NodeMCU firmware comes with ESP8266 Development Kit i.e. NodeMCU Development board.
Since NodeMCU is an open-source platform, their hardware design is open for edit/modify/build.

NodeMCU Dev Kit consist of ESP8266 wifi enabled chip. The ESP8266 is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol.

It is mostly used for the development of IoT (Internet of Things) embedded applications.

ESP8266 comes with capabilities of

- 2.4 GHz Wi-Fi (802.11 b/g/n, supporting WPA/WPA2),
- general-purpose input/output (16 GPIO),
- Inter-Integrated Circuit ($I^2C$) serial communication protocol,
- analog-to-digital conversion (10-bit ADC)
- Serial Peripheral Interface (SPI) serial communication protocol,

- I²S (Inter-IC Sound) interfaces with DMA(Direct Memory Access) (sharing pins with GPIO),
- UART (on dedicated pins, plus a transmit-only UART can be enabled on GPIO2), and
- pulse-width modulation (PWM).

It employs a 32-bit RISC CPU based on the Tensilica Xtensa L106 running at 80 MHz (or overclocked to 160 MHz). It has a 64 KB boot ROM, 64 KB instruction RAM, and 96 KB data RAM. External flash memory can be accessed through SPI.

The ESP8266 module is a low cost standalone wireless transceiver that can be used for end-point IoT developments.

To communicate with the ESP8266 module, the microcontroller needs to use a set of AT commands. The microcontroller communicates with the ESP8266-01 module using UART having a specified Baud rate.

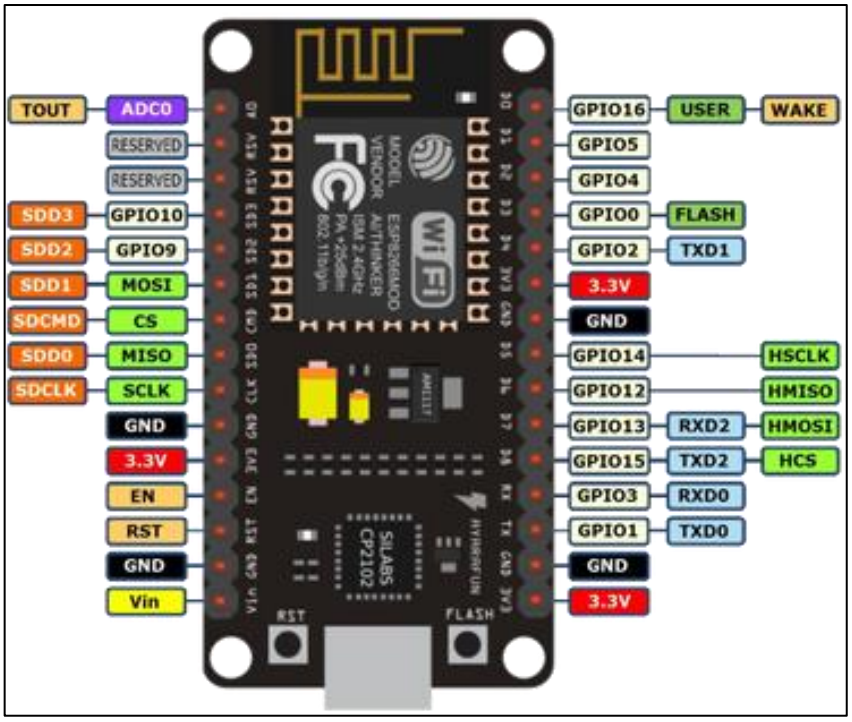



***Figure 2.2:*** Pinout of NodeMCU

## 2.2 LIGHT EMITTING DIODE (LED)

In the simplest terms, a light-emitting diode (LED) is a semiconductor device that emits light when an electric current is passed through it. Light is produced when the particles that carry the current (known as electrons and holes) combine together within the semiconductor material.

Since light is generated within the solid semiconductor material, LEDs are described as solid-state devices. The term solid-state lighting, which also encompasses organic LEDs (OLEDs), distinguishes this lighting technology from other sources that use heated filaments (incandescent and tungsten halogen lamps) or gas discharge (fluorescent lamps).

***Figure 2.3:*** LED

***Terminals of an LED***

A simple single color LED has 2 terminals: *Anode* and *Cathode*.

Anode being the longer terminal of the diode and Cathode being the shorter terminal.

The figure explains different parts of an LED.

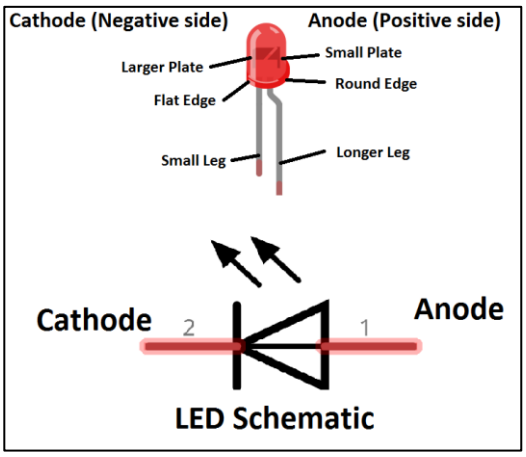



***Figure 2.4:*** Terminals of LED

***Different Colors***

Inside the semiconductor material of the LED, the electrons and holes are contained within energy bands. The separation of the bands (i.e. the bandgap) determines the energy of the photons (light particles) that are emitted by the LED.

The photon energy determines the wavelength of the emitted light, and hence it's color. Different semiconductor materials with different bandgaps produce different colors of light. The precise wavelength (color) can be tuned by altering the composition of the light-emitting, or active, region.

LEDs are comprised of compound semiconductor materials, which are made up of elements from group III and group V of the periodic table (these are known as III-V materials). Examples of III-V materials commonly used to make LEDs are gallium arsenide (GaAs) and gallium phosphide (GaP).

Until the mid-90s LEDs had a limited range of colors, and in particular commercial blue and white LEDs did not exist. The development of LEDs based on the gallium nitride (GaN) material system completed the palette of colors and opened up many new applications.

***Main LED Materials***

The main semiconductor materials used to manufacture LEDs are:

**Indium gallium nitride (InGaN):** blue, green and ultraviolet high-brightness LEDs

**Aluminum gallium indium phosphide (AlGaInP):** yellow, orange and red high-brightness LEDs

**Aluminum gallium arsenide (AlGaAs):** red and infrared LEDs

**Gallium phosphide (GaP):** yellow and green LED

## 2.3 DHT11 Sensor

The DHT11 is a basic, low-cost digital temperature and humidity sensor. It is a single wire digital humidity and temperature sensor, which provides humidity and temperature values serially with the one-wire protocol.

DHT11 sensor provides relative humidity value in percentage (20 to 90% RH) and temperature values in degree Celsius (0 to 50 °C). It uses a resistive humidity measurement component and an NTC temperature measurement component.

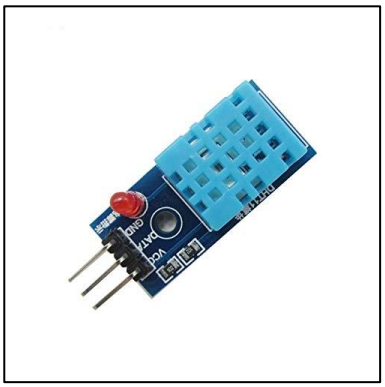


***Figure 2.5:*** DHT11 Sensor Module

***Pin Description***

DHT11 is a 4-pin sensor, these pins are VCC, DATA, GND and one pin is not in use shown in fig below.

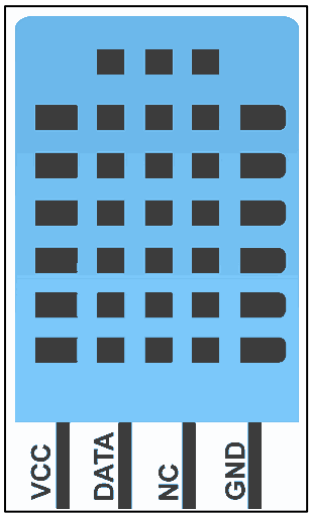



***Figure 2.6:*** Pin Diagram of DHT11

| Pin Number | Pin Name | Pin Description |
|---|---|---|
| 1 | VCC | Power supply<br>3.3 to 5.5 Volt DC |
| 2 | DATA | Digital Output Pin |
| 3 | NC | No Connection |
| 4 | GND | Ground |

***Table 2.2:*** Pin Description of DHT11

### *Communication with Microcontroller*

DHT11 uses only one wire for communication. The voltage levels with certain time values define the logic one or logic zero on this pin.

The communication process is divided into three steps, first is to send a request to the DHT11 sensor then the sensor will send a response pulse and then it starts sending data of a total of 40 bits to the microcontroller.

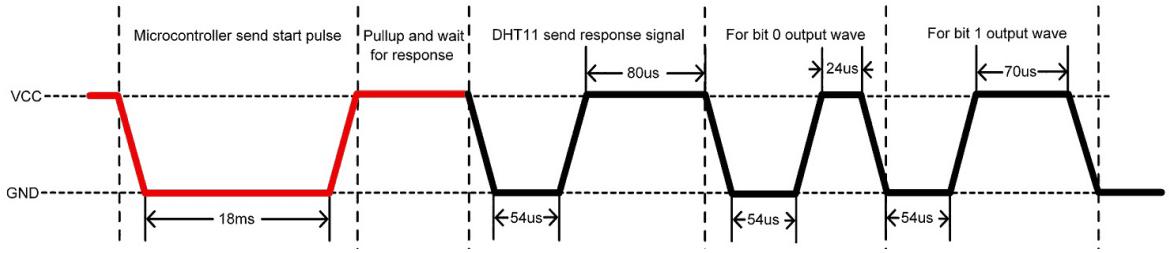



***Figure 2.7:*** Communication Process with Microcontroller

## 2.4 Relay(5V)

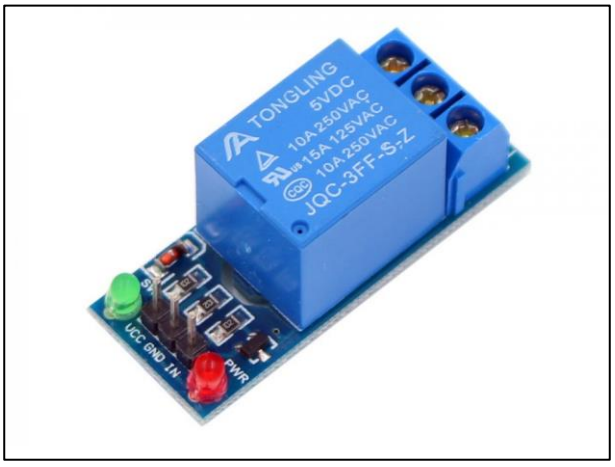



***Figure 2.8:*** 5V Relay Module

A relay is an electrically operated switch. It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals.

Relays are used where it is necessary to control a circuit by an independent low-power signal, or where several circuits must be controlled by one signal.

| | |
|---|---|
| **Coil Voltage** | 5V |
| **Maximum Current** | 10 A |
| **Maximum Voltage** | 30V DC/ 250V AC |

***Table 2.3:*** Relay Specifications

### *Working of a Relay*

The working of a Relay can be better understood by looking at the following images.

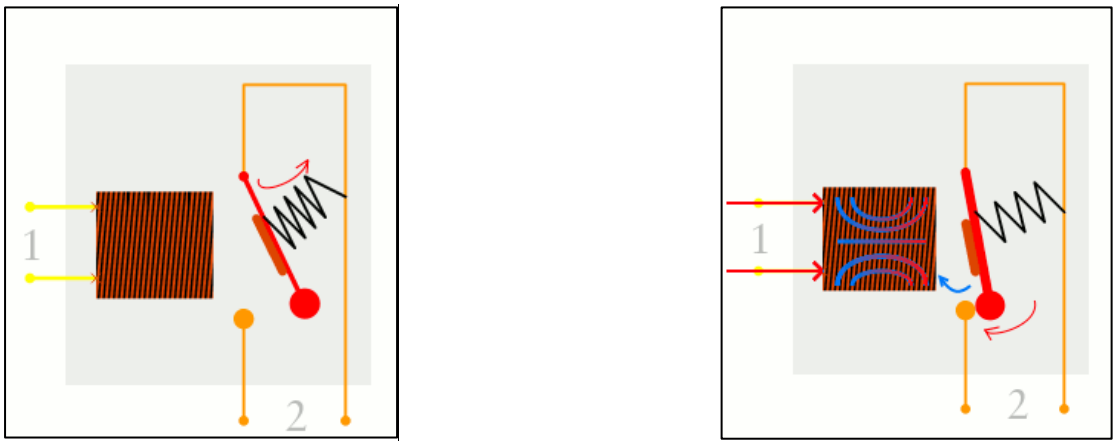



***Figure 2.9:*** Working of Relay (a)

When power flows through the first circuit (1), it activates the electromagnet (brown), generating a magnetic field (blue) that attracts a contact (red) and activates the second circuit (2). When the power is switched off, a spring pulls the contact back up to its original position, switching the second circuit off again.

This is an example of a "normally open" (NO) relay: the contacts in the second circuit are not connected by default and switch on only when a current flows through the magnet. Other relays are "normally closed" (NC; the contacts are connected, so a current flows through them by default) and switch off only when the magnet is activated, pulling or pushing the contacts apart. Normally open relays are the most common.

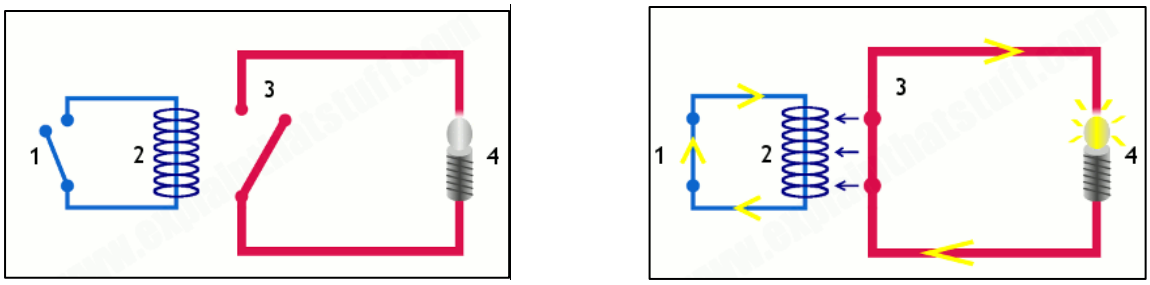



***Figure 2.9:*** Working of Relay (b)

On the left side, there's an input circuit powered by a switch or a sensor of some kind. When this circuit is activated, it feeds current to an electromagnet that pulls a metal switch closed and activates the second, output circuit (on the right side). The relatively small current in the input circuit thus activates the larger current in the output circuit:

- The input circuit (blue loop) is switched off and no current flows through it until something (either a sensor or a switch closing) turns it on. The output circuit (red loop) is also switched off.
- When a small current flows in the input circuit, it activates the electromagnet (shown here as a dark blue coil), which produces a magnetic field all around it.
- The energized electromagnet pulls the metal bar in the output circuit toward it, closing the switch and allowing a much bigger current to flow through the output circuit.
- The output circuit operates a high-current appliance such as a lamp or an electric motor.

# *Chapter 3*

# *SOFTWARE REQUIREMENTS*

# *Chapter 3*

# SOFTWARE REQUIREMENTS

## 3.1 Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also with the help of 3rd party cores, other vendor development boards.

The Arduino IDE supports the languages C and C++ using special rules of code structuring.

### 3.1.1 Installing the Software

1) Visit ***http://www.arduino.cc/en/main/software*** to download the latest Arduino IDE version for your computer's operating system. There are versions for Windows, Mac, and Linux systems. At the download page, click on the "Windows Installer" option for the easiest installation.
2) Save the .exe file to your hard drive.
3) Open the .exe file.
4) Click the "I agree" button to agree to the licensing agreement.
5) Decide which components to install, then click "Next":

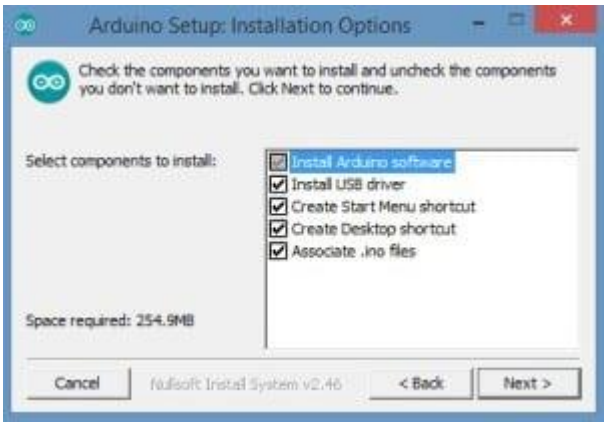



***Figure 3.1:*** Arduino setup 1

6) Select which folder to install the program to, then click "install":


***Figure 3.2:*** Arduino setup 2

7) Wait for the program to finish installing, then click “Close”.
8) Now find the Arduino shortcut on your Desktop and click on it. The IDE will open up and you’ll see the code editor:

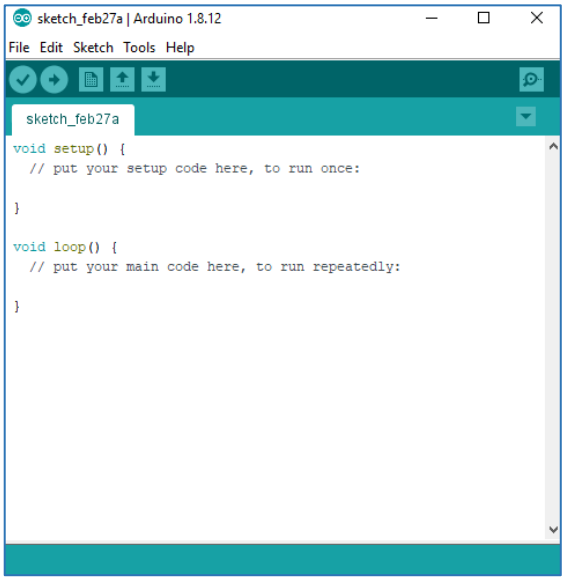



***Figure 3.3:*** Arduino setup 3

### 3.1.2 Board Setup and Installing Libraries

The next thing to do is to make sure the software is set up for your particular Arduino board. Go to the “Tools” drop-down menu, and find “Board”. Another menu will appear where you can select from a list of Arduino models.

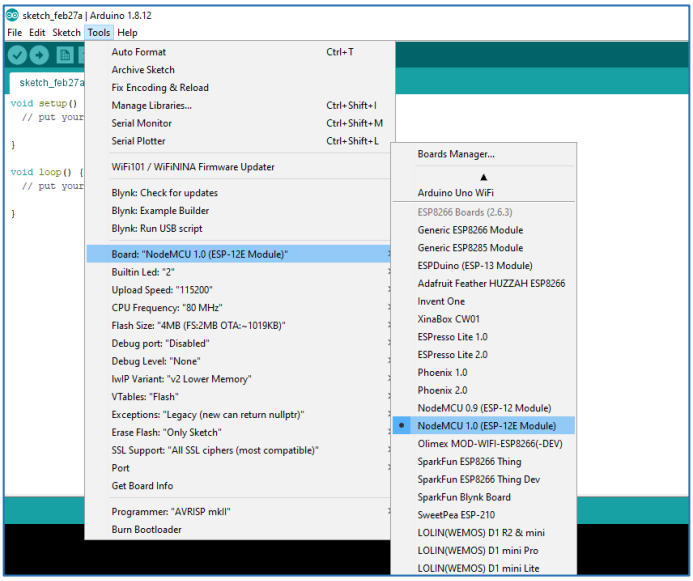



***Figure 3.4:*** Board setup

To install the required libraries, go to “Sketch” and click on “Include Libraries” and then “Manage Libraries”.

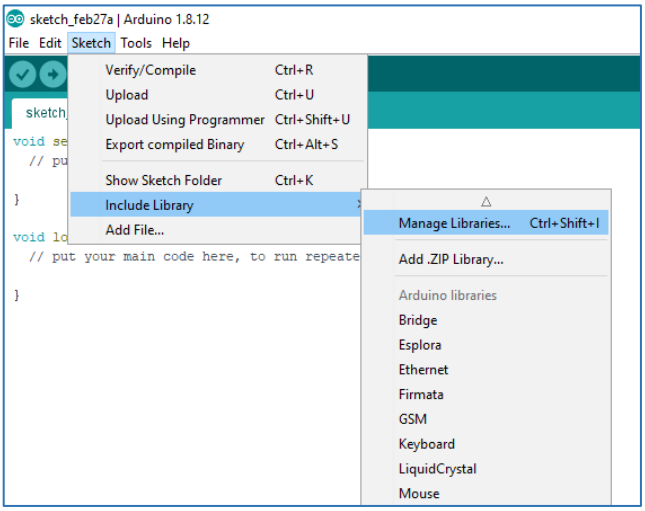



***Figure 3.5:*** Installing Libraries 1

Find the required libraries and click “install”.

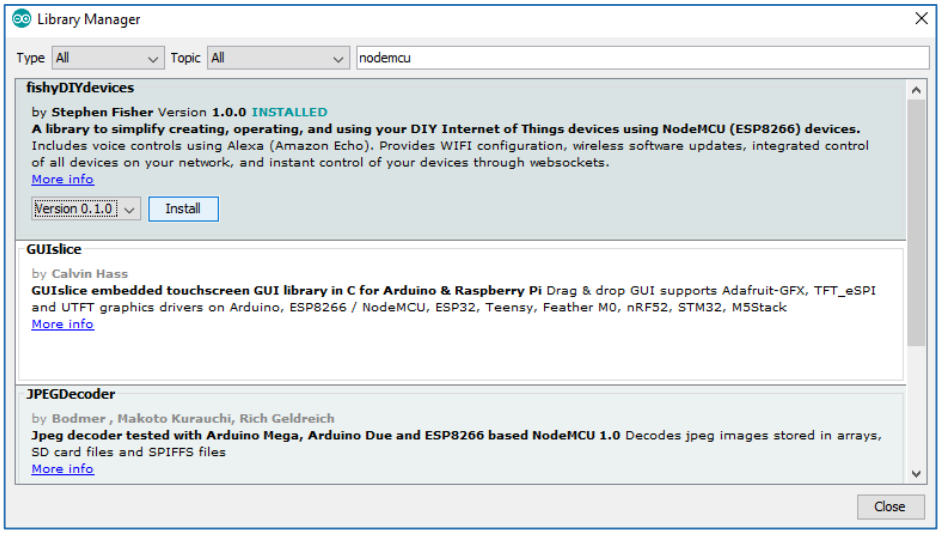



***Figure 3.6:*** Installing Libraries 2

## 3.2 BLYNK APPLICATION

Blynk is an IoT platform with customizable mobile apps, private cloud, rules engine, and device management analytics dashboard, designed for easy and customizable Internet of Things applications. Designing a dashboard on Blynk App for IoT projects is really easy, you just have to organize buttons, sliders, graphs, and alternative widgets onto the screen. We can also edit the widgets as per our requirements.

With the help of Blynk, the software side gets easier than the hardware. Blynk is perfect for interfacing with simple projects like monitoring the temperature of your room or turning lights on and off remotely

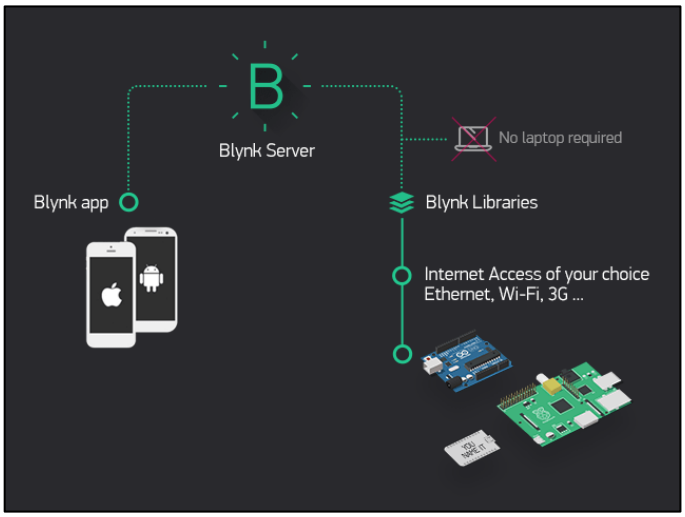



***Figure 3.7:*** BLYNK flow diagram

## 3.3 IFTTT APPLICATION

If This Then That, also known as IFTTT is a free web-based service to create chains of simple conditional statements, called applets.

An applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, or in our case BLYNK application.

# *Chapter 4*

# *DESIGN AND METHODOLOGY*

# *Chapter 4*

# DESIGN AND METHODOLOGY

## 4.1 Block Diagram and Explanation

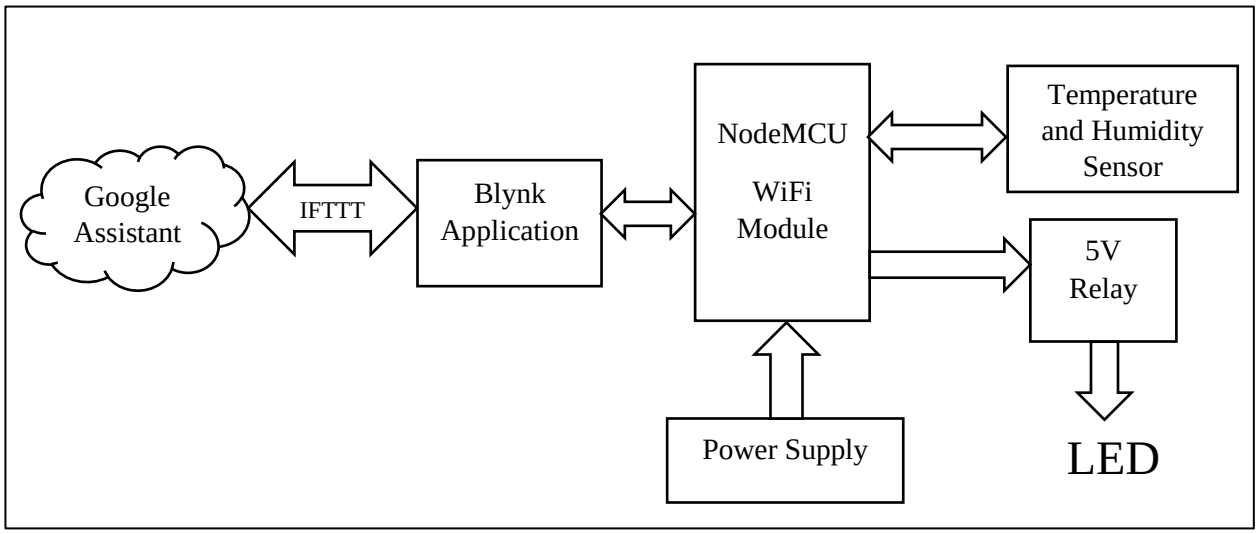



***Figure 4.1:*** Block Diagram of Home Automation System

The main Block of this system is the Microcontroller, ie. the NodeMCU WiFi Module. The microcontroller requires 3V to 3.6V for operation and this is given by a power supply. A temperature-humidity sensor and a 5V relay is connected to the microcontroller which is intern connected to an LED.

The required code is dumped on to the microcontroller and the microcontroller communicates with the BLYNK Application through the Blynk Server.

Since the Google Assistant cannot directly communicate with the Blynk application or NodeMCU, the IFTTT platform acts as a bridge of communication between Google Assistant and BLYNK.

# *Chapter 5*

# *IMPLEMENTATION*

# *Chapter 5*

# IMPLEMENTATION

## 5.1 Connections and Setup

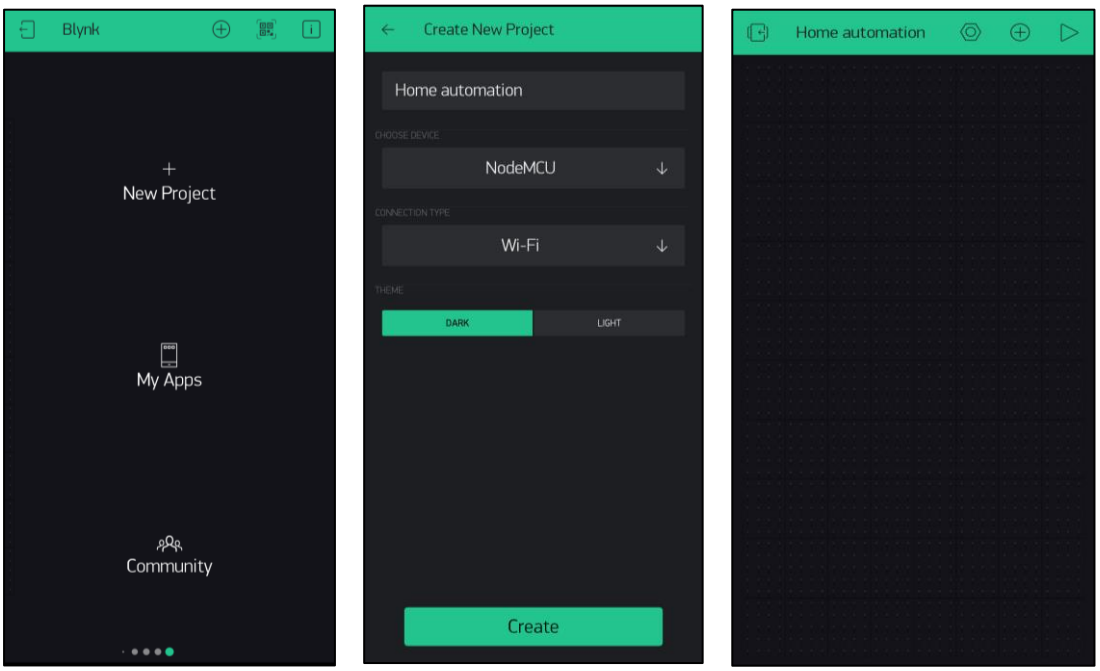
***BLYNK App Dashboard setup for Controlling LED and monitoring Temperature***

Download and install the Blynk app from Google or Apple app store. Then, create a new account by using your email and password. After signing up click on the 'New Project' to start the project.

Now provide a name for the project and as Blynk app works with different hardware models. Here, we are using ESP8266 NodeMCU, so we proceed with NodeMCU.

After these steps click on the 'Create' button to form the project.

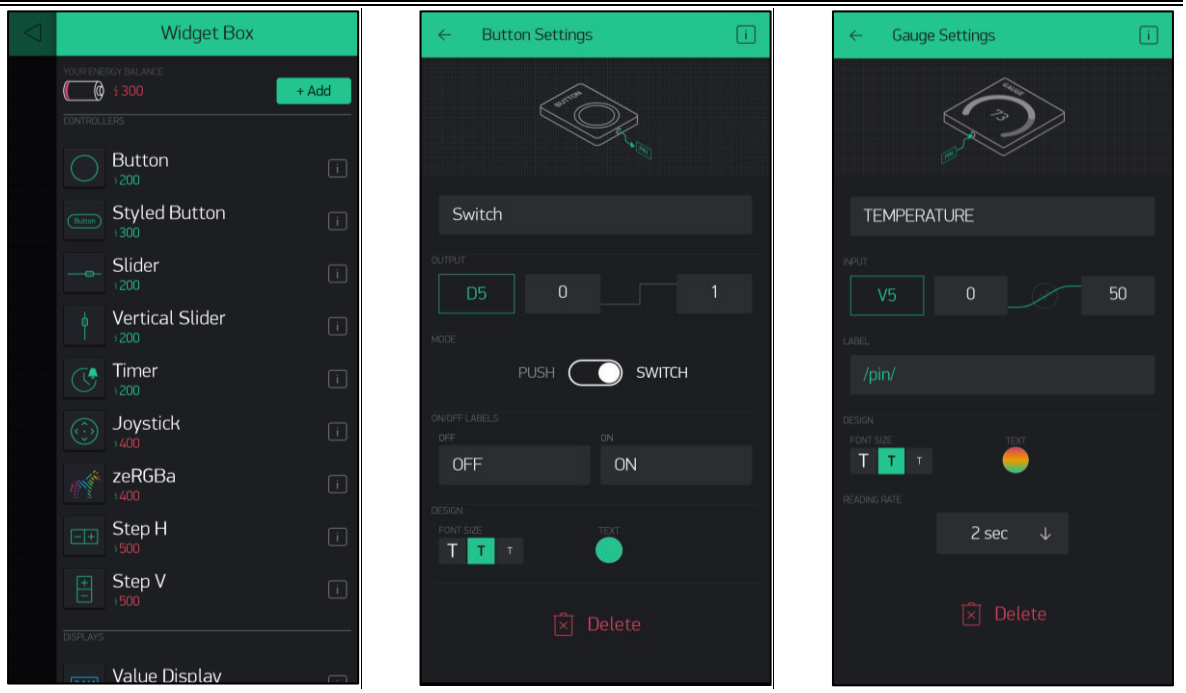
As the blank project opens, add Widgets to it by clicking on the Add button (Plus sign button).




(A) (B) (C)

***Figure 5.1:*** BLYNK setup

Now after this click on 'Button' to add a button in the project. Now in button settings provide a name to your button. After assigning the pin number to the 'OUTPUT'. Also, give names to your On/Off labels. Similarly, Choose 2 'Gauge' widgets for Temperature and Humidity monitoring.




(D) (E) (F)

***Figure 5.1:*** BLYNK setup

## *Connections of NodeMCU, DHT11 and LED*

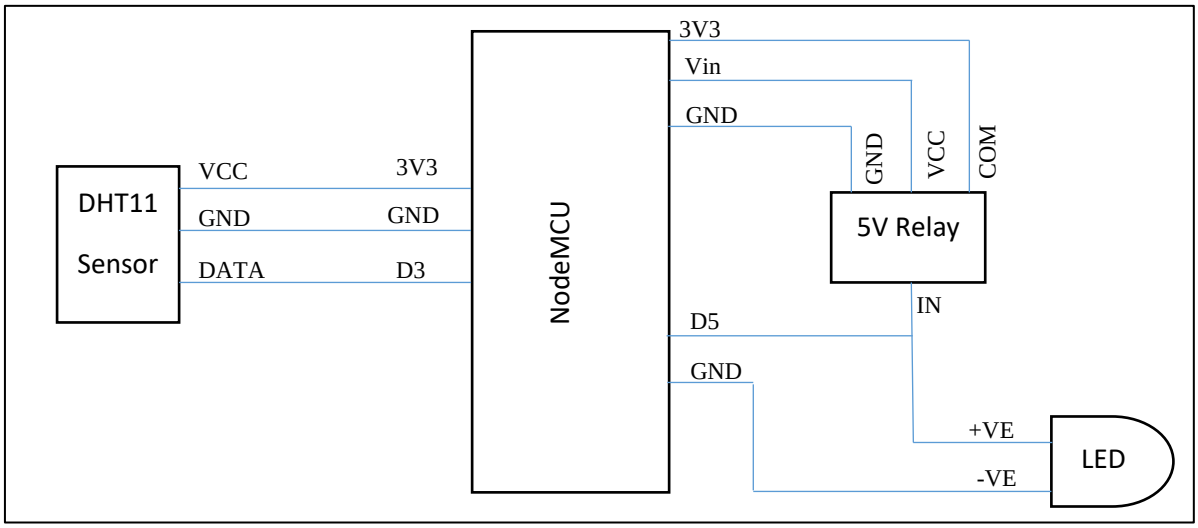



***Figure 5.2:*** Interfacing Connections

| NodeMCU | DHT11 | Relay |
|---|---|---|
| 3V3 | VCC | COM |
| Vin | - | VCC |
| GND | GND | GND, LED(-ve) |
| D3 | DATA | - |
| D5 | - | IN, LED(+ve) |

***Table 5.1:*** Pin connections of NodeMCU, DHT11 and Relay

## 5.2 Code and Explanation

Create a new project in Arduino IDE and write appropriate code in C language for controlling the light and monitoring the temperature and Humidity on BLYNK.

**CODE:**

```
#define BLYNK_PRINT Serial
 #include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>

char auth[] = "L5ag_-kce8edxG7oFpJTq6KPNDS1MiNh"; //Authentication code sent to mail
char ssid[] = "Vinzi";              // WiFi Name
char pass[] = "vinzi123";       // WiFi Password

#define DHTPIN 0         //Data pin of DHT11 is connected to GPIO0 ie, D3
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;
// This function sends Arduino's up time every second to Virtual Pin (5).
void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();       // in degree Celsius
  if (isnan(h) || isnan(t))
 {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Blynk.virtualWrite(V5, t);
  Blynk.virtualWrite(V6, h);
}

void setup()
{
```

```
  Serial.begin(9600); //set the Baud rate
 pinMode(14,OUTPUT); // Positive of LED is connected to GPIO14 ie. D5
  Blynk.begin(auth, ssid, pass);
  dht.begin();
  timer.setInterval(1000L, sendSensor);
}

void loop()
{
  Blynk.virtualWrite(14,HIGH);
  Blynk.virtualWrite(14,LOW);
  Blynk.virtualWrite(14,255);
  Blynk.run();
  timer.run();
}
```

After typing the code, Compile and Upload the code to NodeMCU setup done earlier.

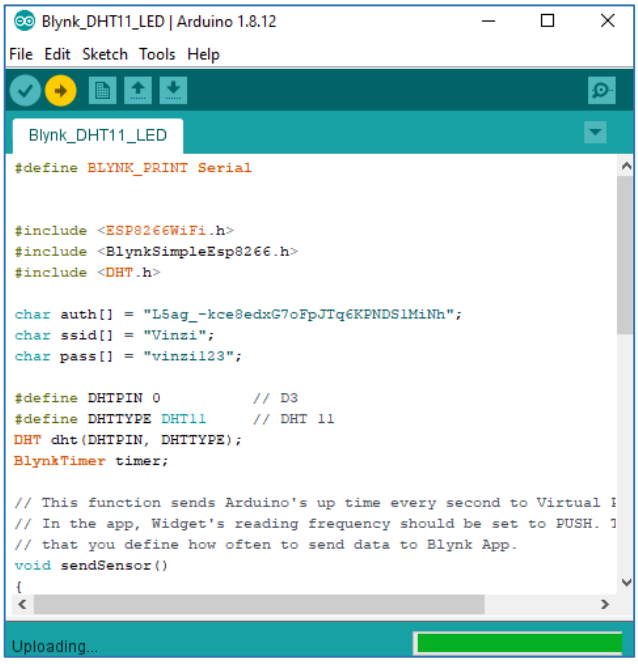


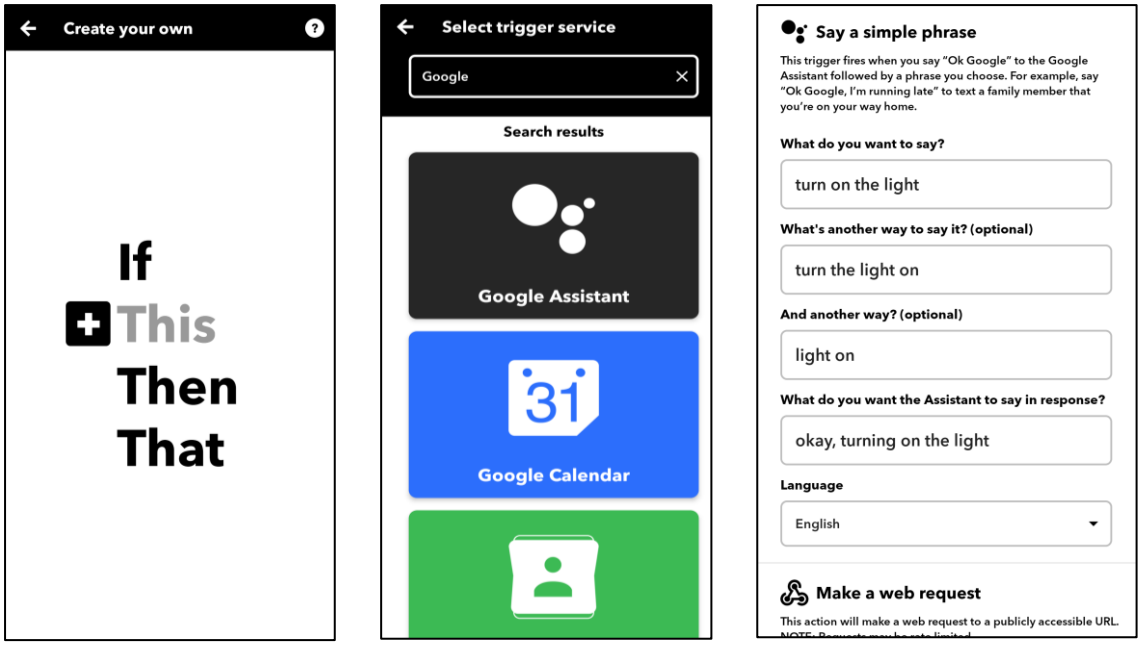

***Figure 5.3:*** Compiling and uploading the code

Now open the project created in BLYNK and set the digital/Virtual pins for each widget onboard corresponding to the connections made in the setup earlier.

### *Setting up on IFTTT Platform*

Open IFTTT Application and click on "Create your own Applet". Now click on "This" to create a trigger.

Select "Google Assistant" as the Trigger and click on "Say a Simple phrase" to type in your own trigger.

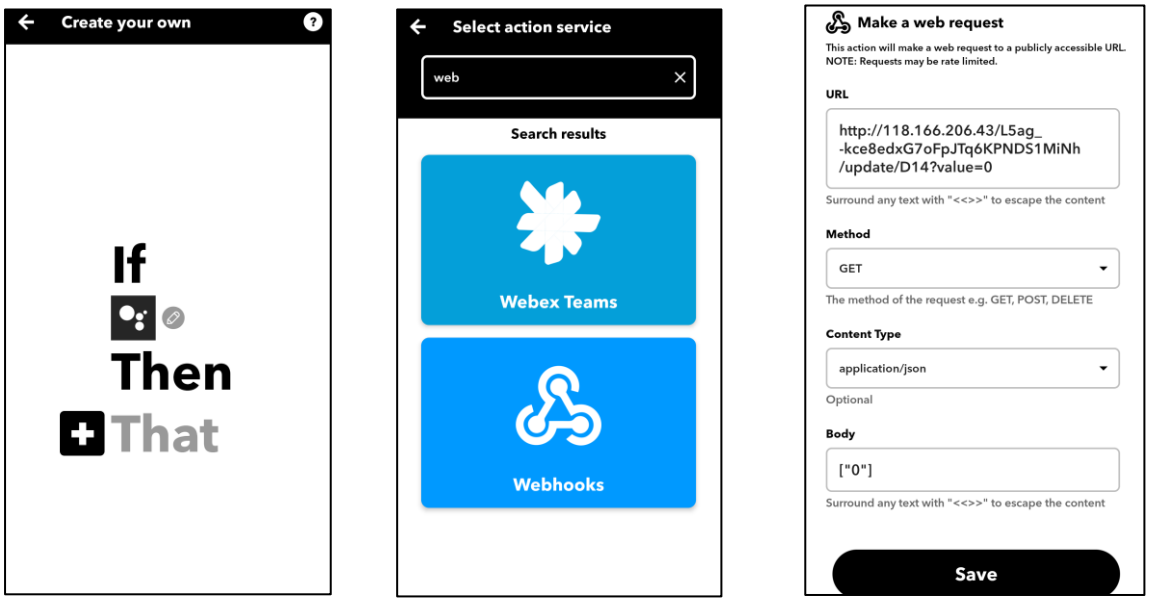



(A) (B) (C)

***Figure 5.4:*** IFTTT application setup

Now, click on "That" to make a web request to Blynk to turn ON/OFF the LED. Then, select the action service as "Webhooks" and type the URL of the BLYNK server of India followed by the Auth code and the pin number of NodeMCU that needs to be controlled.

Then select the method of request as "GET" and in the Body type ["0"] for turning ON the LED and type ["1"] for turning OFF the LED. Now Click "Save".




(D) (E) (F)

***Figure 5.4:*** IFTTT application setup

Create two such applets, one for turning ON and another for turning OFF the LED.

Once everything is set up, we are ready to check the output.

# *Chapter 6*

# *RESULTS AND CHALLENGES FACED*

# *Chapter 6*

# RESULTS AND CHALLENGES FACED

## 6.1 Obtained Result

To see the output turn ON the Wi-Fi and power up the NodeMCU. Open the project created in BLYNK to see the Temperature and Humidity readings. To control the light using Google assistant we need to trigger it.

Now trigger the Google Assistant by saying "OK GOOGLE, Turn on the light" to turn ON the light.

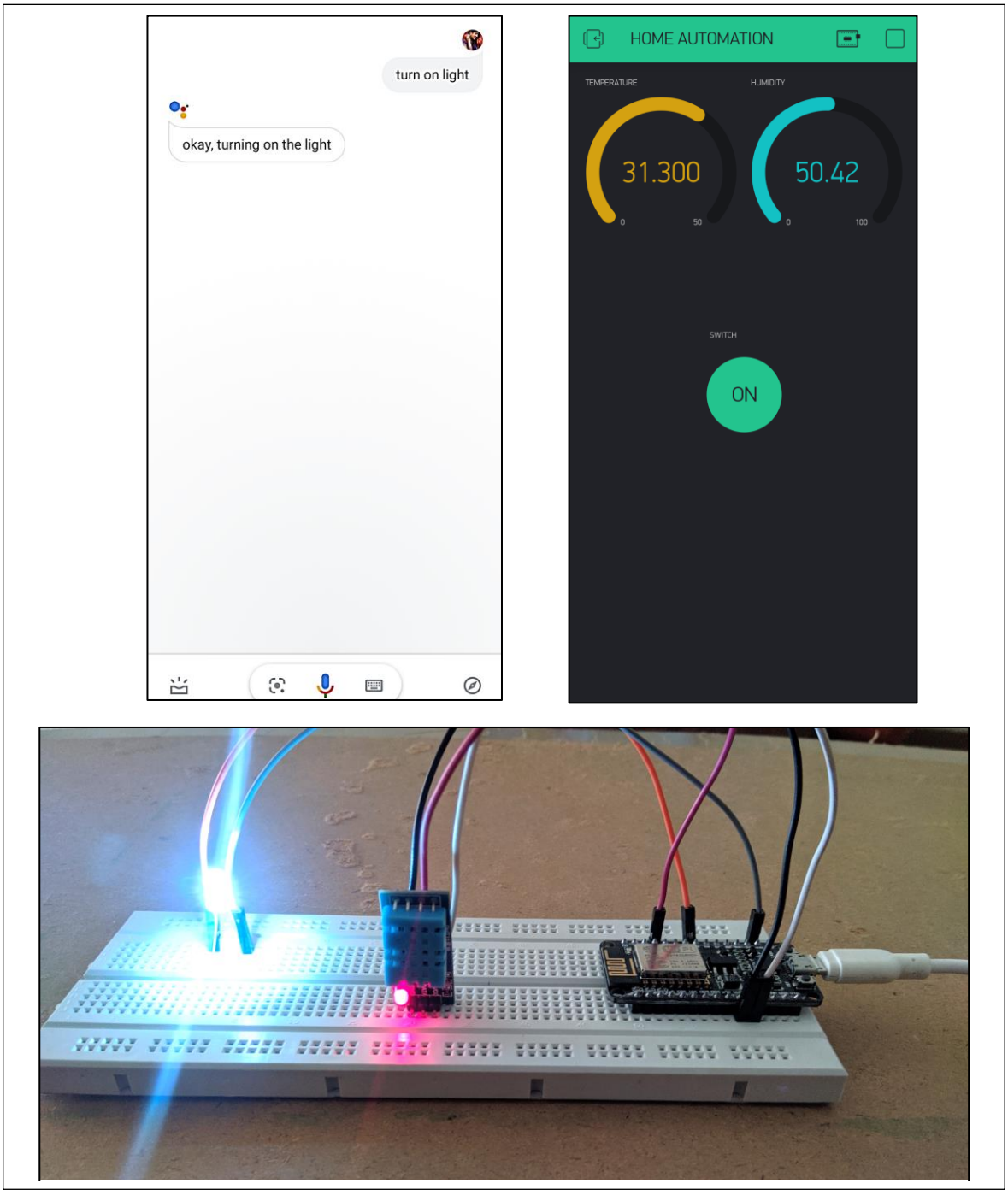



***Figure 6.1:*** Results when the LED is ON

Similarly, say “OK GOOGLE, Turn off the light” to turn OFF the light.

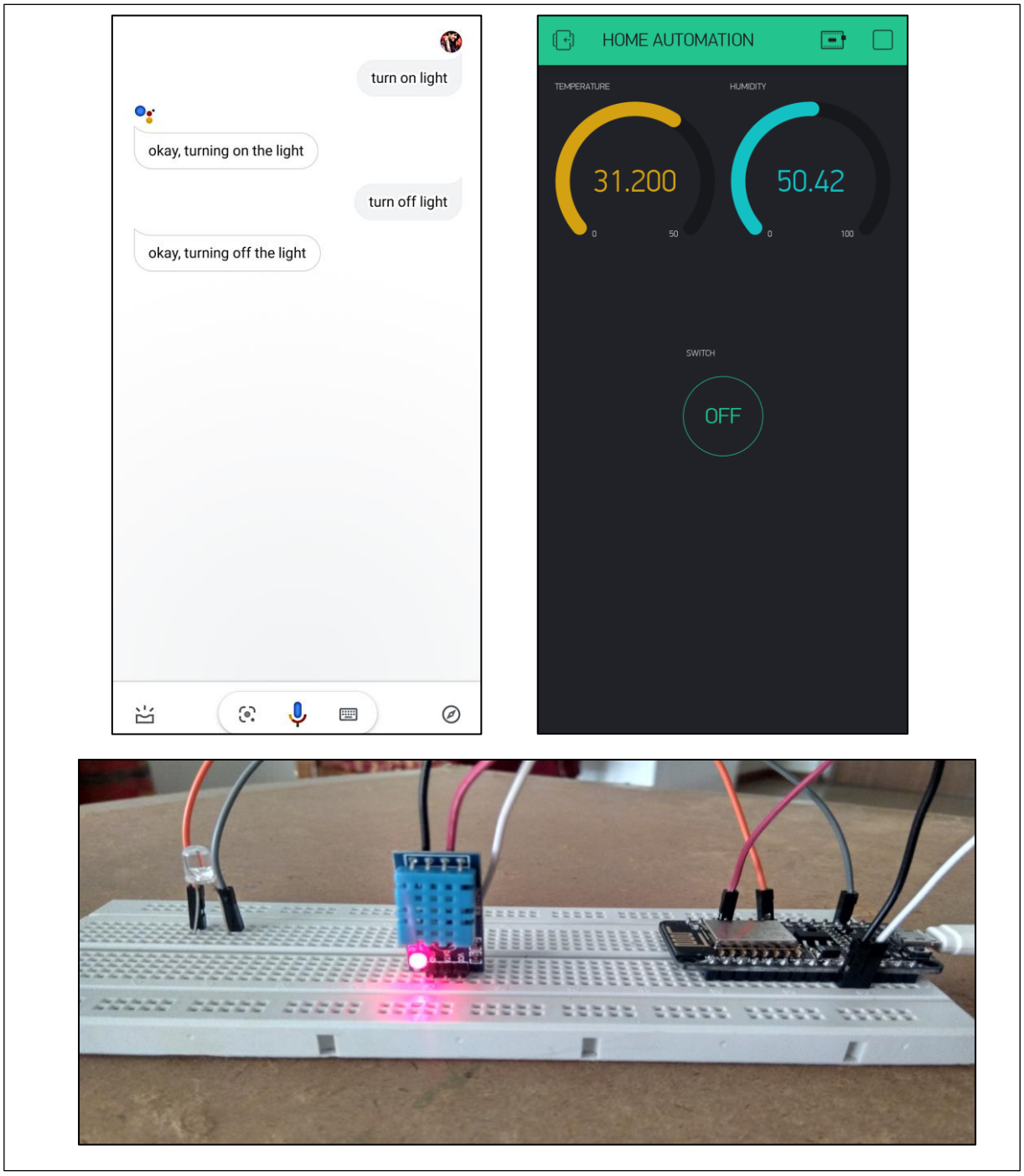



***Figure 6.2:*** Results when LED is OFF

## 6.2 Challenges Faced and overcoming it

The two main challenges faced while doing this project were:

- ***Installing the correct Libraries***
  Downloading and installing the correct libraries for the project was a tricky task. Downloading the wrong library and running the code would throw an error.
  To overcome this challenge, I was recommended to download the libraries only from trusted sites like *GitHub*.

- ***Reading errors***
  Reading errors was another challenge. Some errors that popped up were difficult to understand and debug by a beginner like me.
  The best way to overcome this challenge was to google the error and find the solution for it on platforms like the *Arduino forum* and *Stack Exchange.*

# *Chapter 7*

# *CONCLUSION AND FUTURE SCOPE*

# *Chapter 7*

# CONCLUSION AND FUTURE SCOPE

## 7.1 Conclusion

The Project "IoT based Smart-Home Control and Monitoring using NodeMCU" is successfully completed by interfacing a Temperature & Humidity Sensor (DHT11), a 5V relay and an LED with NodeMCU. The Temperature and humidity of the room are monitored on the BLYNK application which can also be used to control the light with its virtual button.

The light is also controlled by google assistant by telling appropriate commands. When the google assistant is triggered, the light turns ON/OFF depending on the type of trigger that we give.

## 7.2 Future Scope

The future scope of this project involves making homes even smarter. A variety of different sensors can be interfaced with the microcontroller to control and monitor different things at home. For example, we can use a soil moisture sensor for smart gardening or Precision sensors for automatic turn ON and OFF the lights in the room when entered. We can also improvise this project by sending the power consumption data to cloud and monitoring the power usage. This helps us understand the power consumption in different areas of our homes which intern helps us control the power usage.

In the future, 6 out of 10 homes are likely to be smart homes. Hence, home automation undoubtedly has a great scope in the near future.