ASSIGNMENT - MARKING SCHEME

TECHNOLOGY PARK MALAYSIA

CT120-3-2-LPAN

LoWPAN and Ad-Hoc Networking

UC2F2008IT(IoT)

STUDENT NAME: KEANE PUTRA SETIAWAN

STUDENT ID: TP052117

LECTURER NAME: KAMALANATHAN SHANMUGAM

HAND OUT DATE: 22 February 2021

HAND IN DATE: 23 May 2021

WEIGHTAGE: 50%

Table of Contents

1.0 Introduction

Figure 1.1: Smart Home

Source: www.themobilerealestateblog

In the recent years, the popularity of a smart environment that depends on technologies such as Internet of Things, Artificial Intelligence and Digital Visualization have been rising (Han, Kim, & Kim ,2021). One of the major applications is the home industry. A general definition of Smart Home is integrating different types of technological services within a home boundary. These technologies can be tailored based on the services required by the users. (Sovacool, Martiskainen, & Del Rio , 2021). These services can be controlled from anywhere and anytime as long as the user has an internet connection in the mobile phone. Some of the services that the user can control are door, light, window, TV, and air conditioners. The main goal of a Smart Home is to provide homeowners with convenience and practicality. Another interesting feature of Smart Home is monitoring total energy consumption to reduce energy waste and environmental pollution (Zamanloo, et al. 2021).Since Smart Home allow user to control electronical devices remotely, it can reduce electricity bills. Security features can also be embedded inside the Smart Home, it can lock door via passwords to prevent any intruder from breaching in.

In this project, a simulated smart home system will be created using esp8266 as the microcontroller.

1.1 IoT wireless communication protocols

Since this project is just a simulation of a smart home, the wireless communication protocol will be used is Wi-Fi. However, in the real implementation of IoT, not only Wi-Fi can be used but there are other wireless network protocols such as LoRaWAN and BLE that can be used.

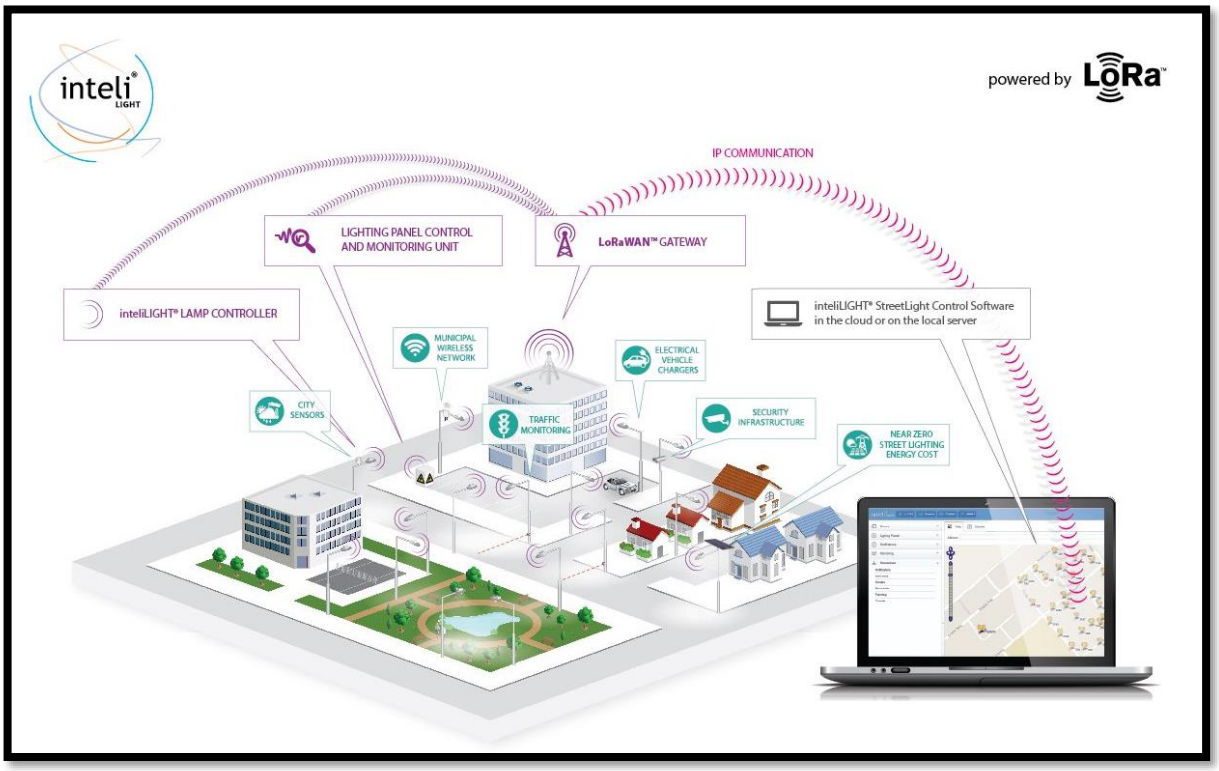

inteli LIGHT
powered by LoRa
IP COMMUNICATION
LIGHTING PANEL CONTROL AND MONITORING UNIT
LoRaWAN™ GATEWAY
inteliLIGHT® LAMP CONTROLLER
inteliLIGHT® StreetLight Control Software in the cloud or on the local server
MUNICIPAL WIRELESS NETWORK
ELECTRICAL VEHICLE CHARGERS
CITY SENSORS
TRAFFIC MONITORING
SECURITY INFRASTRUCTURE
NEAR ZERO STREET LIGHTING ENERGY COST

Figure 1.1.1: LoRaWAN
Source: electronics360.globalspec.com

LoRaWAN (Low Power Wide Area) is one of the most popular wireless protocol for IoT application as it is cost-friendly and low power consumption. (Islam et al, 2021). This protocol is suitable for long range purposes (Zahmatkesh & Al-Turjman, 2020) due to its ability in providing reliable communication despite being more than 10 kilometres away. (Queralta et al 2019). Protocols that are suitable for IoT are protocols with less power consumption to reduce number of times replacing batteries in the sensors.

Figure 1.1.2: BLE
Source: www.libelium.com

There is another wireless communication protocol that works well for IoT which is BLE (Bluetooth Low Energy). BLE has the same communication capacity with its predecessor Bluetooth but with reduced cost and energy consumption (Babun et al 2021). The main reason why BLE is suitable for IoT applications is that it uses lesser amount of energy in comparison to other wireless protocol (Ruiz & Gomez-Nieto, 2017).

1.2 Project Scope

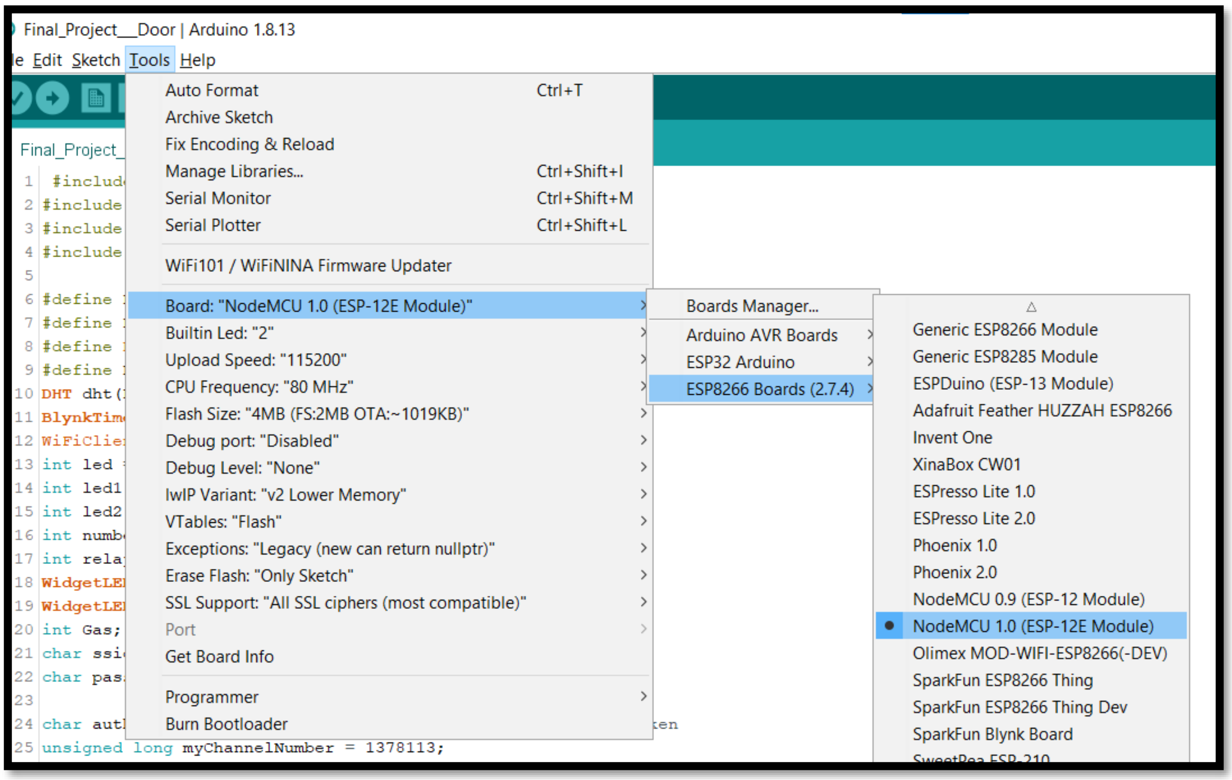
- The user can turn on light via the Blynk app.
- The DHT11 sensor will measure the temperature and humidity and show these values in the Blynk app and ThingSpeak website.
- If temperature is above 30 degree Celsius, the air conditioner will turn on.
- If humidity level is above 75%, the fan will turn on.
- The MQ2 gas sensor will measure the concentration of gases in the air and show this value in the Blynk app and ThingSpeak website.
- If concentration of gas is above 600, a buzzer, LED and water pump will turn on.
- A solenoid door can be open via the Blynk app using a password.

2.0 Hardware

2.1 ESP8266

The “heart” of any IoT project will be the microcontroller . A microcontroller is a small and cost friendly device specially designed to carry a specific action in an embedded system (George, 2020). Using a microprocess is beneficial for IoT project as it is small in size and does not require long time to execute task (Technical Editor, 2017). These microcontrollers exist everywhere in daily life applications such as washing machine and air conditioners. There are a lot of microcontrollers exist in the market, but in this project, the ESP8266 microcontroller version 1.0 will be used.

ESP8266 is a cost-friendly microcontroller that has Wi-Fi module that allows user to access the circuit from anywhere and anytime (Niv, 2019). It follows the IEEE 802.11 b/g/n network protocol. The frequency rate is 80 MHz, and it has 17 GPIO pins.



```
 #includ
#include
#include
#include

#define
#define
#define
#define
DHT dht(
BlynkTim
WiFiClie
int led
int led1
int led2
int numb
int rela
WidgetLE
WidgetLE
int Gas;
char ssi
char pas

char aut                                              ten
unsigned long myChannelNumber = 1378113;
```

Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Legacy (new can return nullptr)"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"

Figure 2.1.1: Board for ESP8266

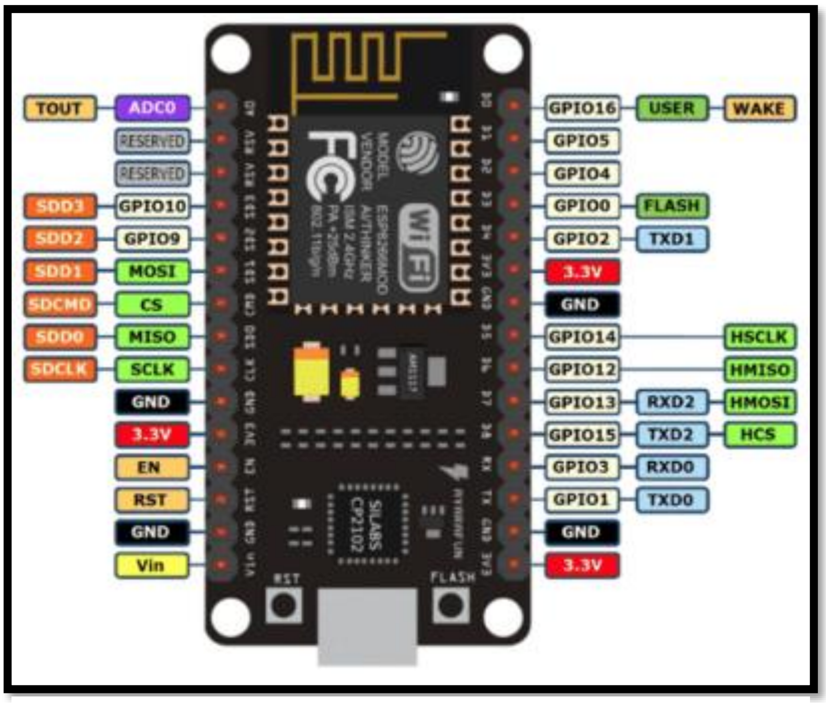


Figure 2.1.2: ESP8266 GPIO Pins

Source: www.nyebarilmu.com

1. 3.3 V = Provide voltage of 3.3 volt
2. Vin = There are 3 of them. The function is to act as source pin to add external source of power
3. GND = There are 4 of them. It is a common reference point to measure voltage against any other point as it has no voltage. To complete the project, all electronical components must be connected to it.
4. GPIO 0 – GPIO 16 = Digital pins of ESP8266
5. ADC0 = Analogue pin
6. MOSI (Master Output Slave Input) = Data that travels from the “master” to the “slave”
7. MISO (Master Input Slave Output) = Data that travels from the slaver” to the “master”
8. SCLK (Serial Clock) = provide serial clock to synchronize the master and slave communication
9. RX and TX = Communication pins between transmitter and receiver

2.2 Sensor

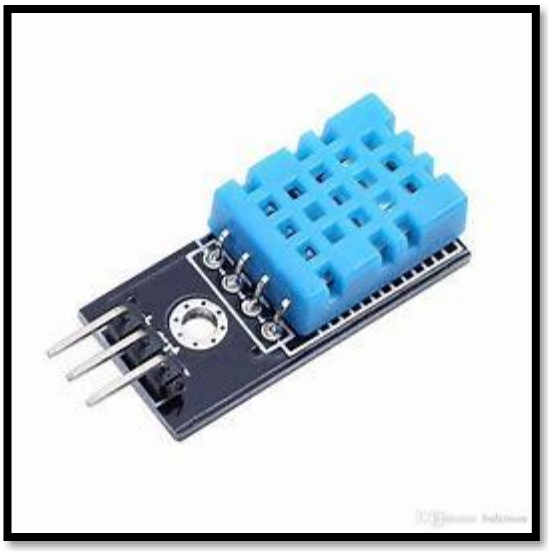


Figure 2.2.1: DHT 11

Source: www.institutodigital.com

In this project, the sensor that will be used is DHT 11. Before discussing further on what is DHT11, the word “sensor” must be defined. Sensor is a small device within a system that detect and measure variables or changes in a certain location. These variables can be anything from temperature, humidity, distance, speed and others. The sensor will convert these physical variables into a digital signal that is processable by the system (Sharief, 2020).

The sensor that will be used in this project is a DHT11 humidity and temperature sensor. It detects the temperature and humidity changes in the environment (Campbell, 2015). The main application of DHT 11 is in weather stations. A higher humidity level tends to indicate that rain might occur.

In this project DHT 11 is a suitable sensor for this smart home project because when there are any changes to the temperature and humidity, home appliances will be automatically turn on. If the humidity level is more than 75%, the fan will turn on. If the temperature is more than 30 degree Celsius, the air conditioner will turn on.

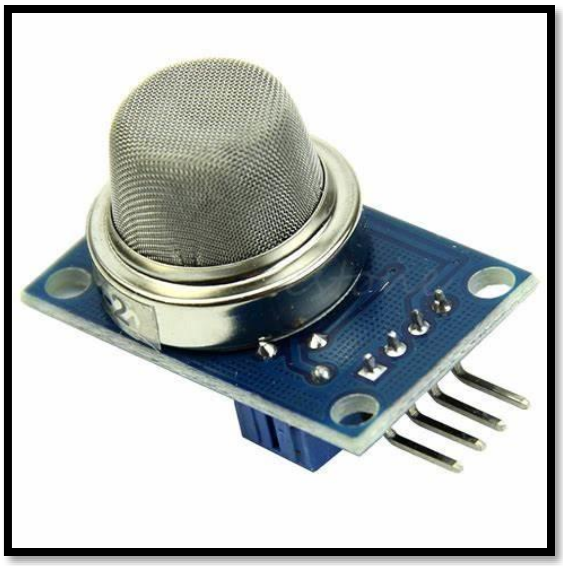


Figure 2.2.2: Sensor MQ2

Source: www.institutodigital.com

The MQ-2 sensor is detecting the concertation of gas that is available in the air. (Mukherjee,2016). It can detect gases such as Butane, Hydrogen, Alcohol, Propane, LPG, and Methane. If the gas concentration level is above 600, a warning light will turn on, water pump will start pouring water and the buzzer will product an alert sound.

2.3 Others

Aside from the ESP8266 and sensor, the rest of the electronical components are simple yet necessary to complete the circuit.

1. Bread board

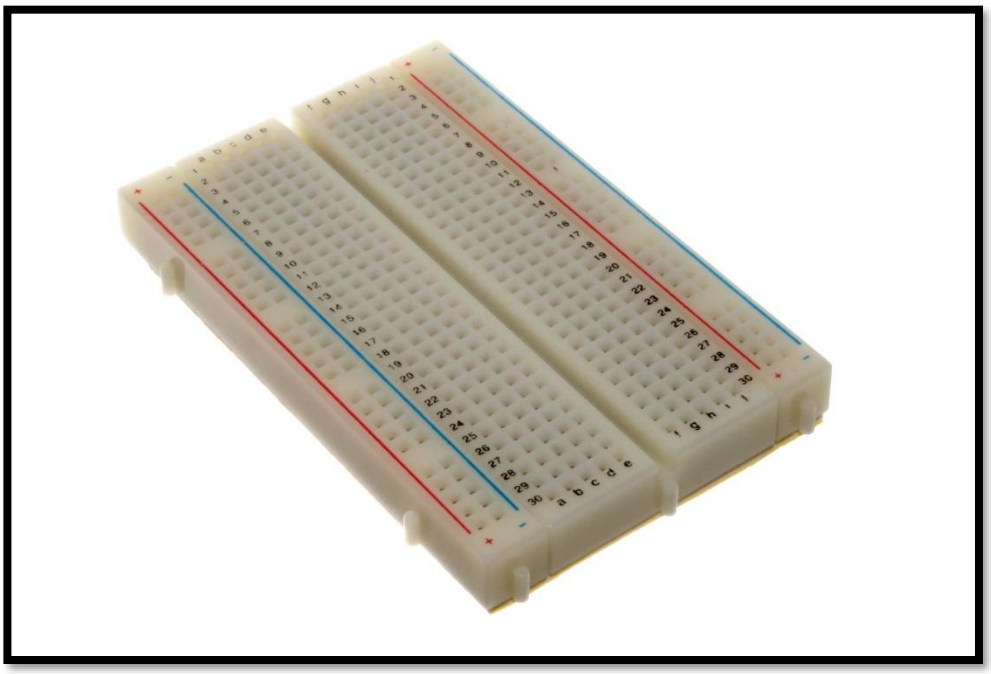


Figure 2.3.1: Breadboard

Source: www.wikipedia.com

2. LED

Figure 2.3.2: LED

Source: www.hallroad.com

3. Jumper wires

Figure 2.3.3: Jumper Wires

Source: www.nullspacegroup.com

4. Buzzer

Figure 2.3.4: Buzzer

Source: www.staticrapidonline.com

5. Solenoid door lock

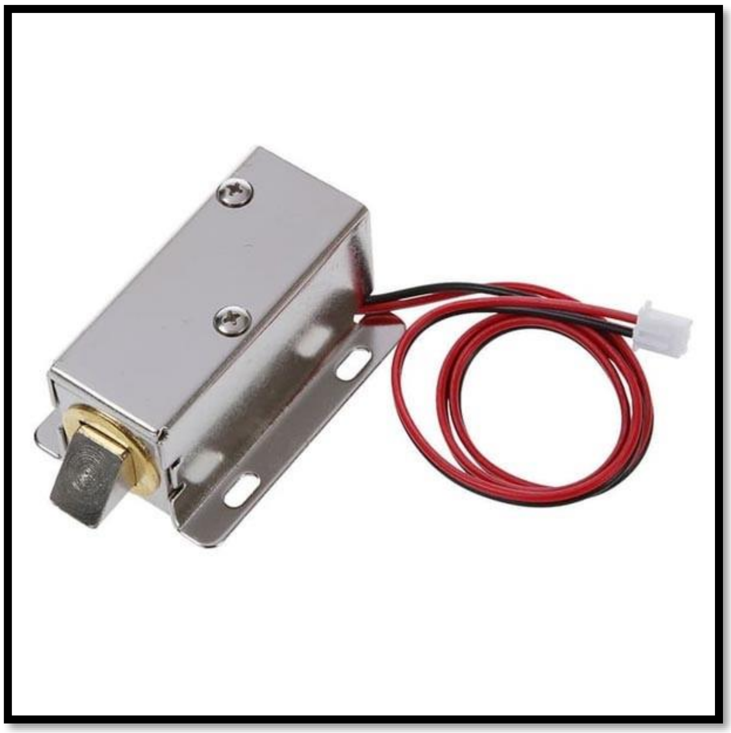


Figure 2.3.5: Solenoid Door Lock

Source: www.digiwarestore.com

6. 5v channel relay

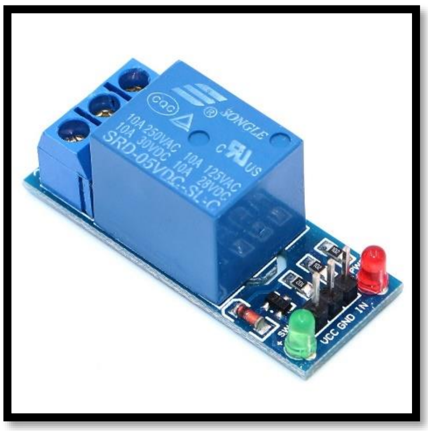


Figure 2.3.6: 5V Relay

Source: www.storagegoogleapis.com

7. 9V power adaptor

Figure 2.3.7: 9V power adaptor

Source: www.ebay.com

8. Mini water pump and a hose

Figure 2.3.8: Mini water pump

Source: www.ebay.com

2.4 Circuit Diagram

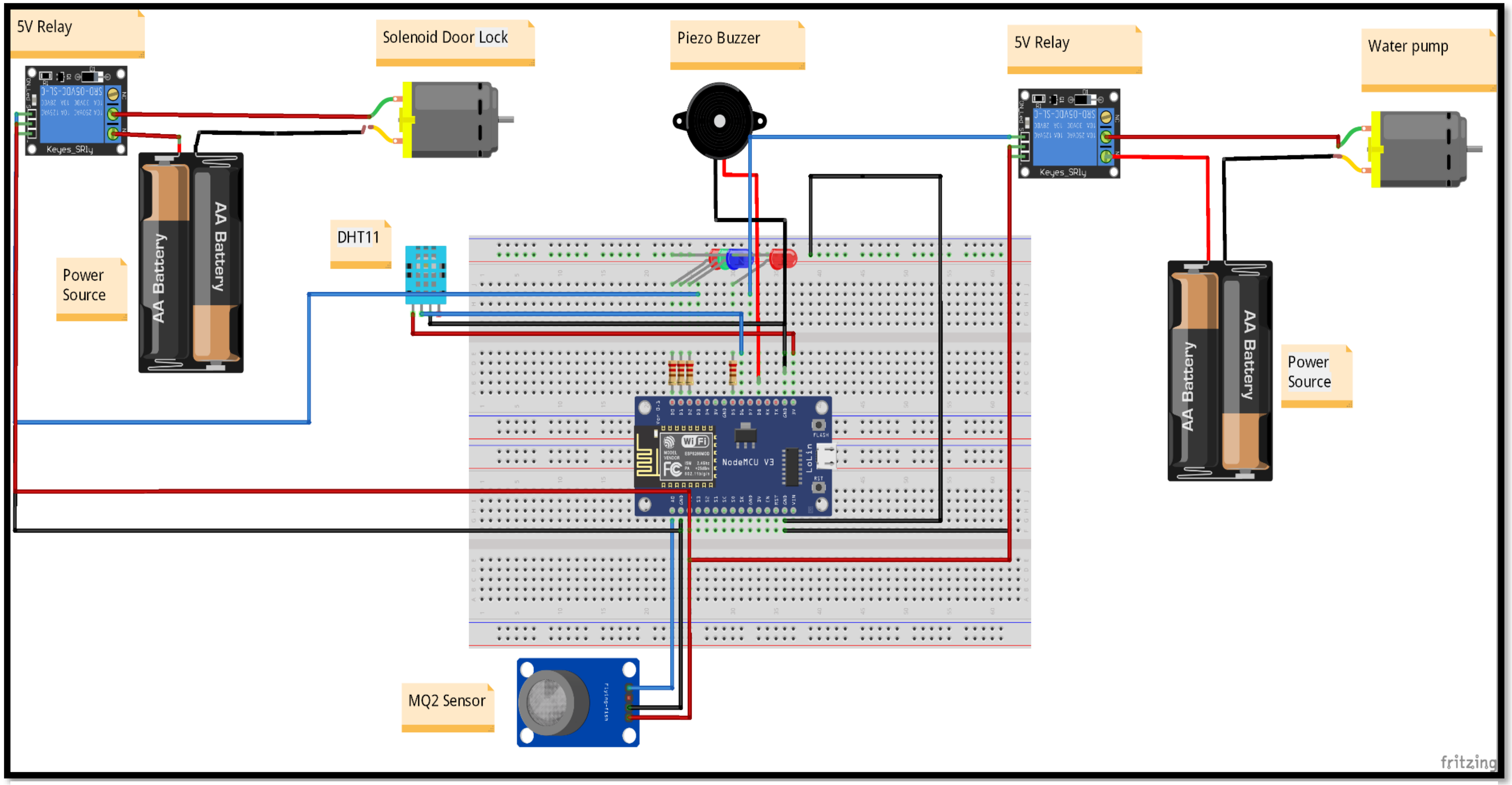


Figure 2.4.1: Circuit Diagram of the Project

ESP8266 PINS	COMPONENT
D0	Smart LED
D1	Humidity LED
D2	Temperature LED
D3	Data in of Relay (door lock)
D5	Gas LED
D6	DHT out pin
D7	Data in of Relay (water pump)
D8	Positive leg of buzzer
G	Negative leg of buzzer DHT negative pin MQ2 negative pin Both relay ground pin
3V	DHT positive pin
A0	A0 of MQ2 sensor
VU	VCC of MQ2 sensor VCC of both relays

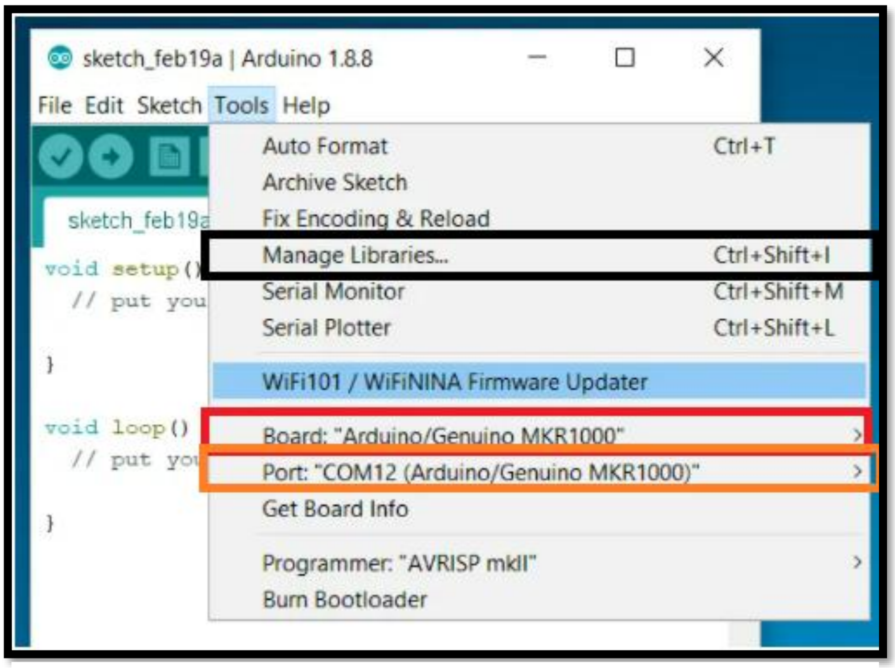
Table 2.4.1: PINS

3.0 Software & Connectivity

3.1 Arduino IDE

Arduino IDE (Integrated Development Environment), it is an official software to create, verify and upload code into various microcontrollers such as ESP 32, ESP 8266 and Arduino Maker Uno (Aqeel, 2018). Before uploading, port and board type must be chosen (figure 3.1.1, orange and red boxes). Arduino use embedded C as its programming language. Embedded C is a programming language that is commonly used to communicate with hardware or an embedded system (Teja,2021). The main reason embedded C is the more popular compared to Assembly, BASIC, C++ and Python for embedded system is because it is more efficient, shorter compiling time and more portable.

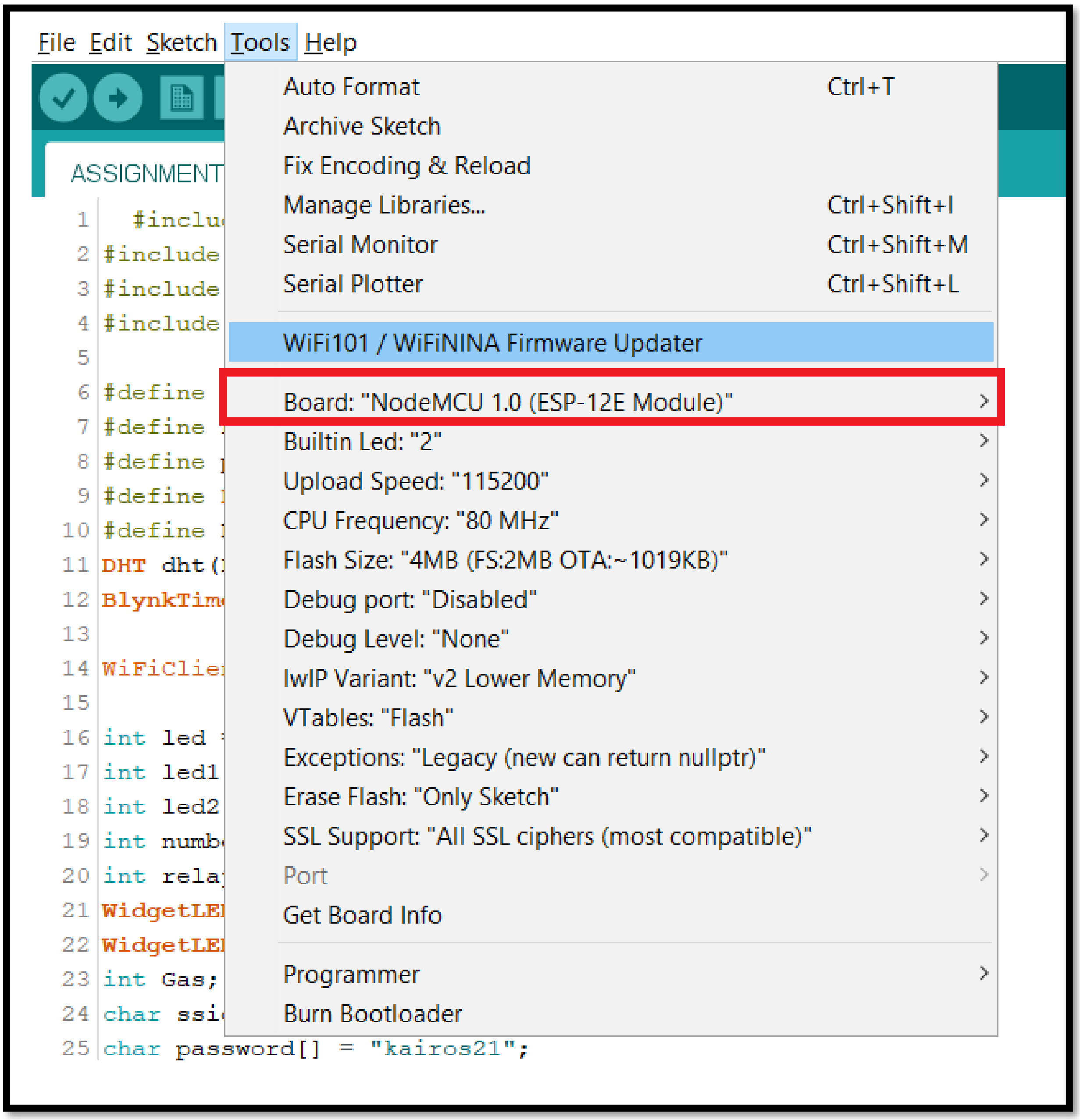
In Arduino IDE, a lot of libraries are available to simplify users when coding (figure 3.1.1 black box). Libraries are a collection of code that provide users with the interface to directly communicate to a specific device such as sensors and modules. There are specific functions to reduce the effort required for users to code (Naveenraj, 2020).


sketch_feb19a | Arduino 1.8.8
File Edit Sketch Tools Help
sketch_feb19a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader

```
void setup()
  // put you
}

void loop()
  // put you
}
```

Board: "Arduino/Genuino MKR1000"
Port: "COM12 (Arduino/Genuino MKR1000)"

Figure 3.1.1: Arduino IDE


File Edit Sketch Tools Help

ASSIGNMENT

```
 #inclu
#include
#include
#include

#define
#define
#define
#define
#define
DHT dht(
BlynkTim

WiFiClie

int led
int led1
int led2
int numb
int rela
WidgetLE
WidgetLE
int Gas;
char ssi
char password[] = "kairos21";
```

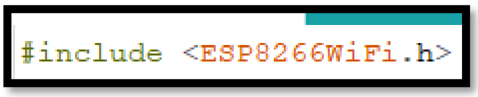
Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Manage Libraries...	Ctrl+Shift+I
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater	
Board: "NodeMCU 1.0 (ESP-12E Module)"	>
Builtin Led: "2"	>
Upload Speed: "115200"	>
CPU Frequency: "80 MHz"	>
Flash Size: "4MB (FS:2MB OTA:~1019KB)"	>
Debug port: "Disabled"	>
Debug Level: "None"	>
lwIP Variant: "v2 Lower Memory"	>
VTables: "Flash"	>
Exceptions: "Legacy (new can return nullptr)"	>
Erase Flash: "Only Sketch"	>
SSL Support: "All SSL ciphers (most compatible)"	>
Port	>
Get Board Info	
Programmer	>
Burn Bootloader	

Figure 3.1.2: Board Type

In the Arduino IDE, the NodeMCU 1.0 (ESP-12E Module) is not available, it is necessary to install the board library from the browser. This is the link to install the board library http://arduino.esp8266.com/stable/package_esp8266com_index.json.

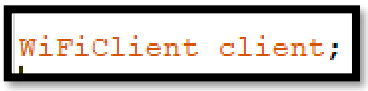
3.2 Wi-Fi

Wi-Fi plays a huge role in development of Internet of Things. It is used to trigger the machine-to-machine communication (Maravedis, 2020). IoT appliances such as cameras, watch and others will require bandwidth of a wireless broadband network to enable them to share data with each other. Currently the most appropriate type of Wi-Fi to be used is Wi-Fi HaLow (IEEE 802.11ah) (Administrator, 2018). The main reason why it works wonderfully with IoT is that is solves the range and power issues of IoT. IoT applications are spread in a huge area and to provide a steady communication between those range can be proven to be an issue.



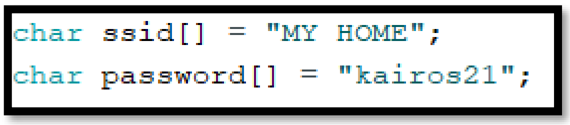
```
#include <ESP8266WiFi.h>
```

Figure 3.2.1: Library



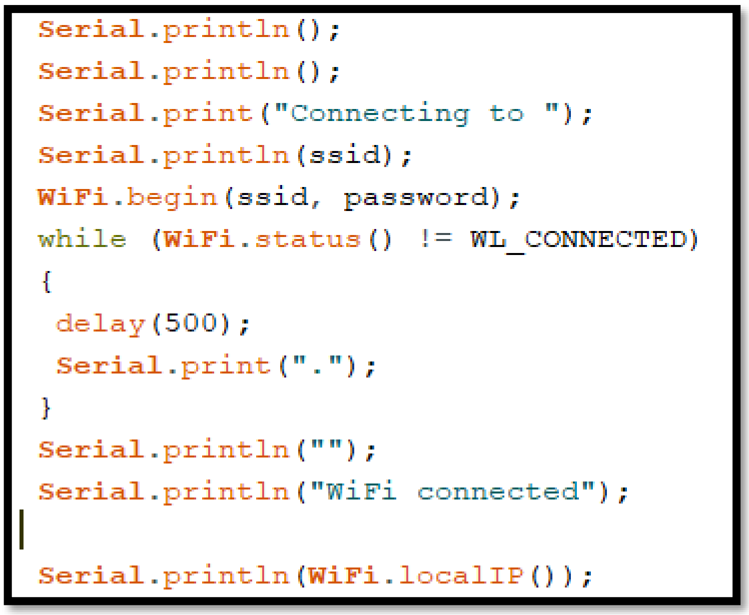
```
WiFiClient client;
```

Figure 3.2.2: Establishing a client



```
char ssid[] = "MY HOME";
char password[] = "kairos21";
```

Figure 3.2.3: Wi-Fi name and 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");

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

Figure 3.2.4: Printing IP address

3.3 Microsoft Excel

Microsoft Excel was created since 1982 and has rose in terms of popularity along with other Microsoft Office's products (Excelhelp, 2021). Microsoft excel is arguably the most common, flexible, and widely used business software to do business activities

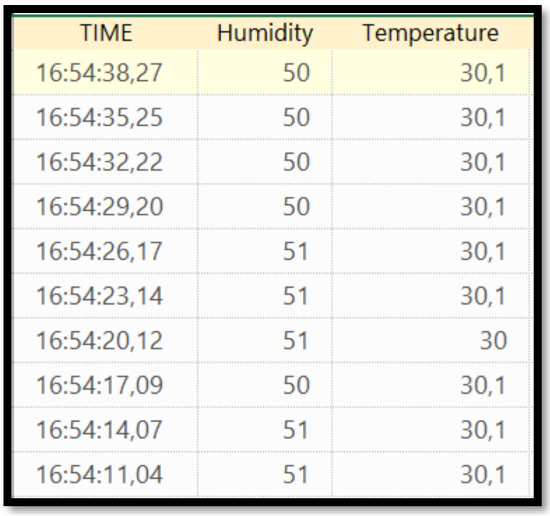
Here are several top features of Microsoft Excel:

1. Provide a visual model to analyze data. These models can be in various forms of graph and chart
2. Multiple people can work together on the same file.
3. Accessible from any devices from phone to table to laptop.

In this project, Microsoft Excel Data Streamer is used to store the change of data **locally** in the user's own computer. Not only that, but it is also possible to do some data visualization and data analysis by displaying the data in table in a form of a graph. The most ideal graph for this project will be a line graph as the relation between time and temperature, humidity and gas concentration level are desired (Bruce, 2012). Using another visualization model such as bar graph or pie chart does not bring any purpose as changes are not seen. A normal excel file does not have the Data Streamer functionality immediately, hence, it is mandatory to download before attempting to transfer the data.

Here are the steps to transfer data to the Data Streamer in Microsoft Excel:

1. To download Microsoft Excel Data Streamer
   File => Option => Add-ins => Manage COM Add-ins => Tick Excel Data Streamer => Ok.
2. After download is complete, click the Data Streamer tab and connect a device. Connect to the port where it is connected to the microcontroller.
3. Start data
4. If desired, plot a line graph.



TIME	Humidity	Temperature
16:54:38,27	50	30,1
16:54:35,25	50	30,1
16:54:32,22	50	30,1
16:54:29,20	50	30,1
16:54:26,17	51	30,1
16:54:23,14	51	30,1
16:54:20,12	51	30
16:54:17,09	50	30,1
16:54:14,07	51	30,1
16:54:11,04	51	30,1

Figure 3.3.1: Table Data

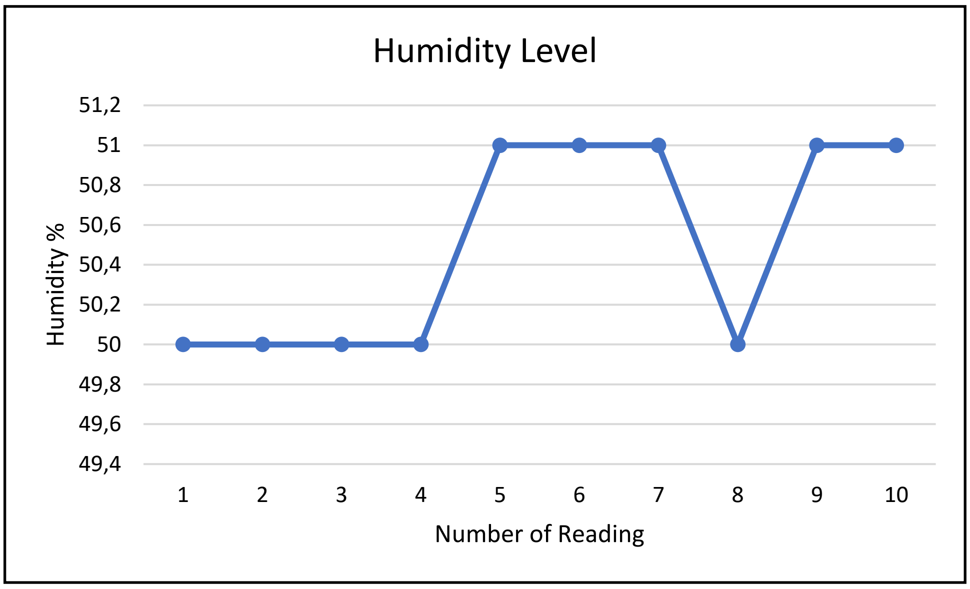


Figure 3.3.2: Line graph of Humidity Level

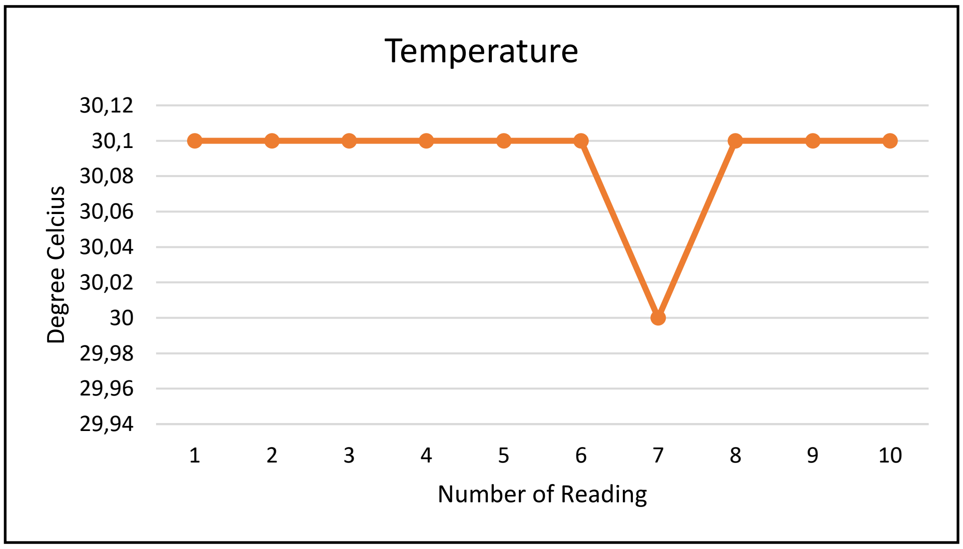
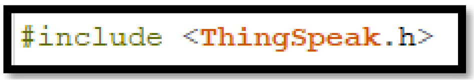


Figure 3.3.3: Line graph of Temperature

3.4 ThingSpeak

ThingSpeak is an online based platform to visualise and analyze data stored in the **cloud**. Cloud storage is remote platform that provide users with the ability to store data virtually without using any storage from the device. Users can also expand or scale the amount of storage depending on the requirement (Morgan, 2018). ThingSpeak is one of MatLab product where it allows the processing and analyzing of live data coming from the sensors (Ali, 2021). These are the main steps in ThingSpeak:

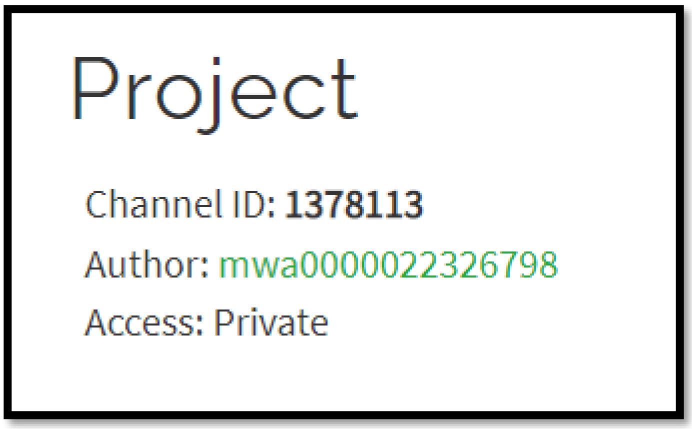
1. Read and collect data.
2. Provide visualisation and analysis.
3. Perform actions according to result.



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

Figure 3.4.1: Library

There are two main components from ThingSpeak that must be added inside the Arduino IDE which are write API key and channel id.



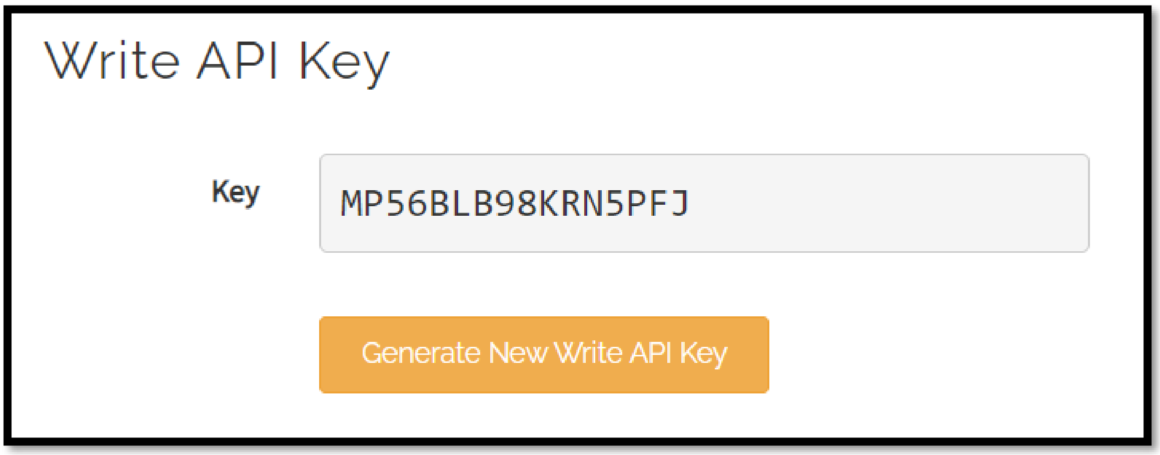
Project

Channel ID: **1378113**

Author: mwa0000022326798

Access: Private

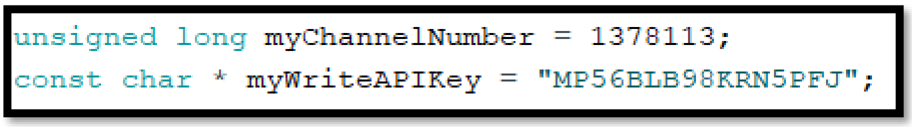
Figure 3.4.2: Channel ID



Write API Key

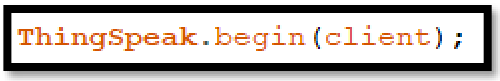
Key: MP56BLB98KRN5PFJ

Figure 3.4.3: Write API Key



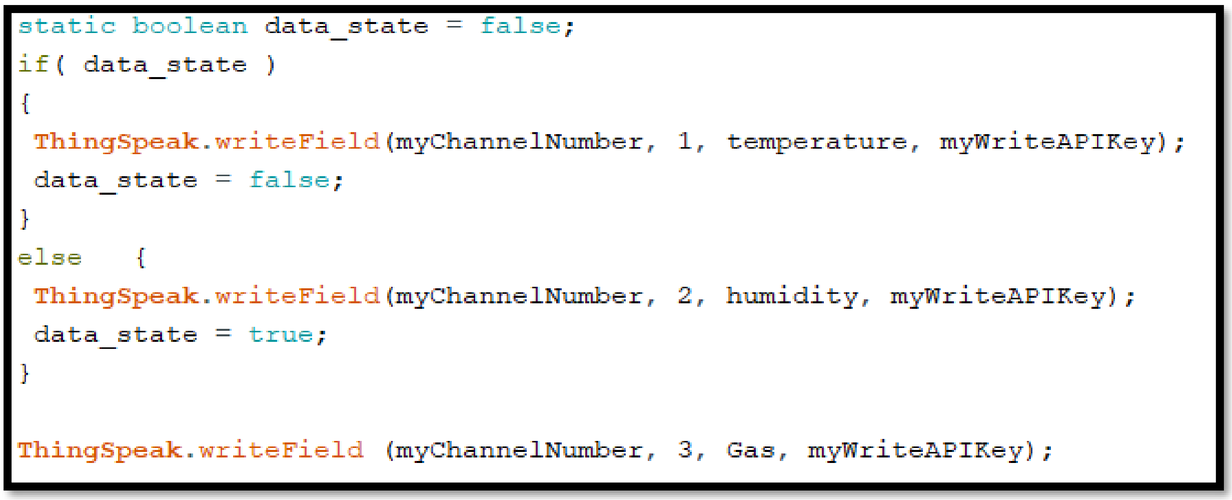
```
unsigned long myChannelNumber = 1378113;
const char * myWriteAPIKey = "MP56BLB98KRN5PFJ";
```

Figure 3.4.4: Both channel ID and Write API key stored.



```
ThingSpeak.begin(client);
```

Figure 3.4.5: Begin ThingSpeak Connection



```
static boolean data_state = false;
if( data_state )
{
 ThingSpeak.writeField(myChannelNumber, 1, temperature, myWriteAPIKey);
 data_state = false;
}
else   {
 ThingSpeak.writeField(myChannelNumber, 2, humidity, myWriteAPIKey);
 data_state = true;
}

ThingSpeak.writeField (myChannelNumber, 3, Gas, myWriteAPIKey);
```

Figure 3.4.6: To determine which Channel ID should the parameter.

Since DHT11 is measuring both humidity and temperature, a data state must be determined to differentiate these two parameters. The temperature will be written in channel 1, humidity will be written in channel 2 and gas concentration level will be written in channel 3.

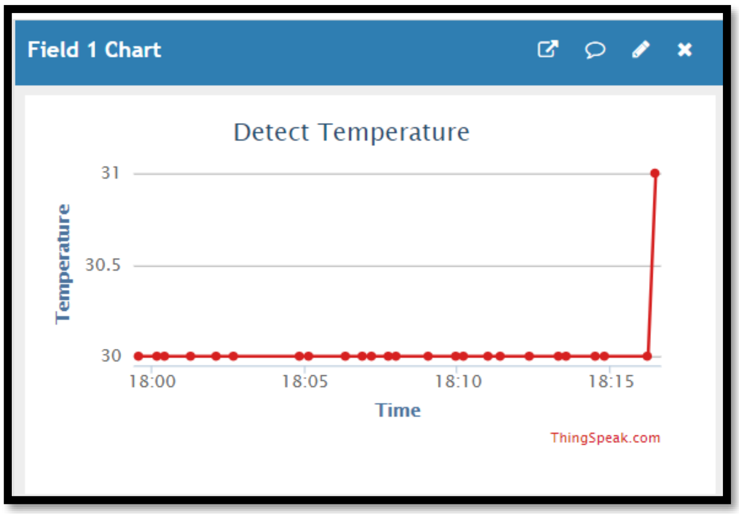


Figure 3.4.7: Temperature in Channel 1

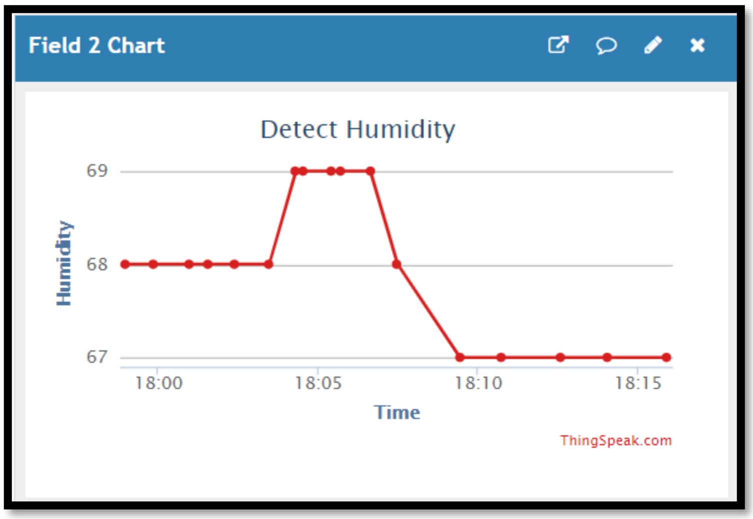


Figure 3.4.8: Humidity in Channel 2

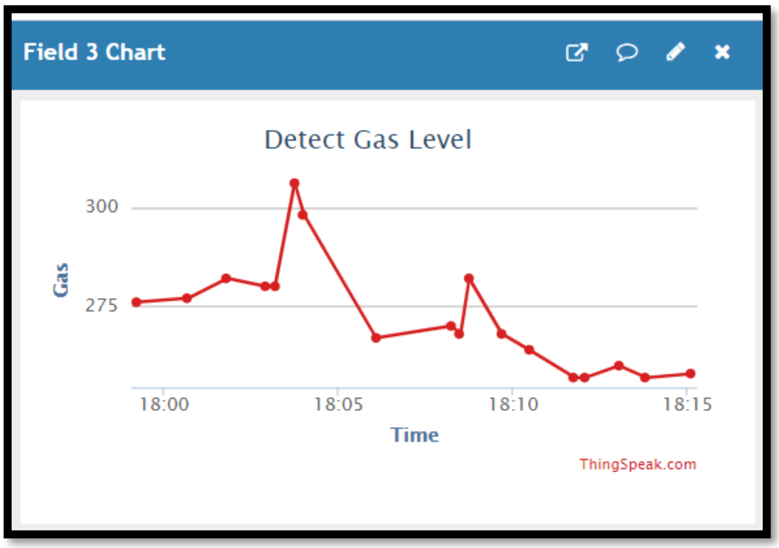


Figure 3.4.9: Gas Concentration Level in Channel 3

3.5 Blynk

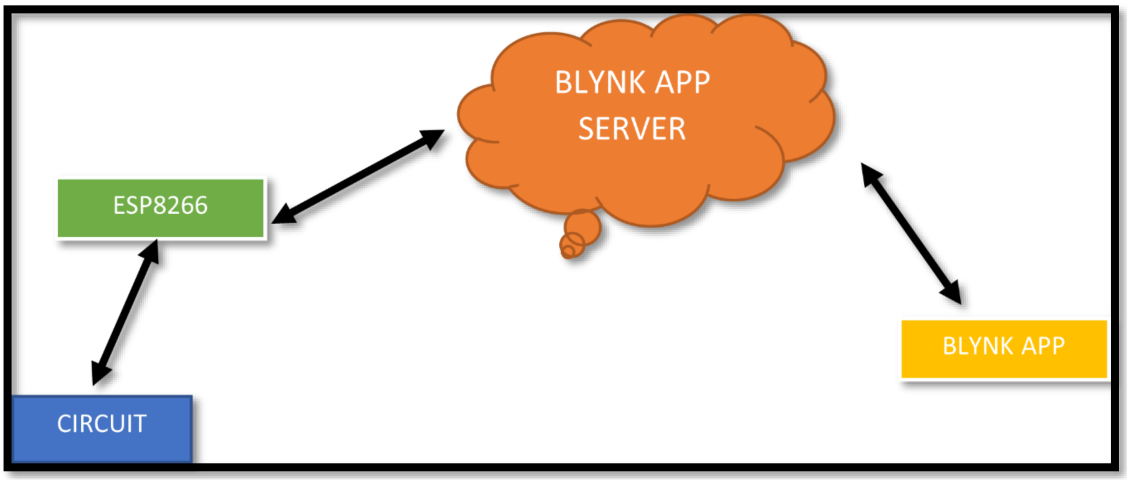
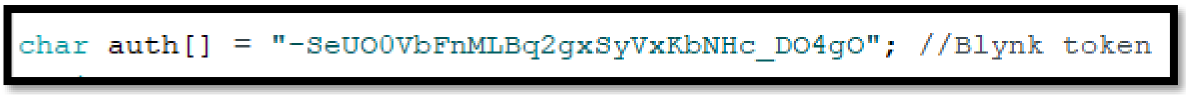


Figure 3.5.1: Blynk app to ESP8266 framework

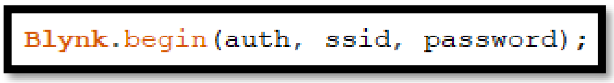
Blynk is a mobile based application that allows users to create their own smart system to control all the IoT devices over the internet (Holmes, 2020).



```
char auth[] = "-SeUO0VbFnMLBq2gxSyVxKbNHc_DO4gO"; //Blynk token
```

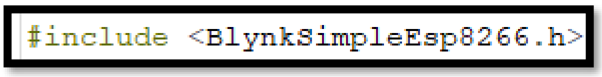
Figure 3.5.2a: Authentication Token from Blynk

Upon creating a project in the Blynk app, it will send an authentication token to the registered email where this token must be entered into the ARUDINO IDE. Without entering the token, the connection will not be established.



```
Blynk.begin(auth, ssid, password);
```

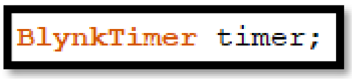
Figure 3.5.2b: Authentication, SSID, password is required.



```
#include <BlynkSimpleEsp8266.h>
```

Figure 3.5.3: Library

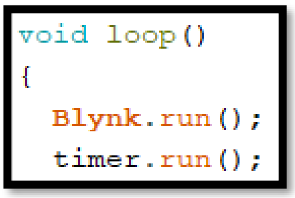
This library must be included in the ARDUINO IDE.



```
BlynkTimer timer;
```

Figure 3.5.4: Blynk Timer

BlynkTimer is used to perform periodic actions in the void loop.



```
void loop()
{
  Blynk.run();
  timer.run();
```

Figure 3.5.5: Running Blynk app

To run the Blynk app, Blynk.run and timer.run must be added.

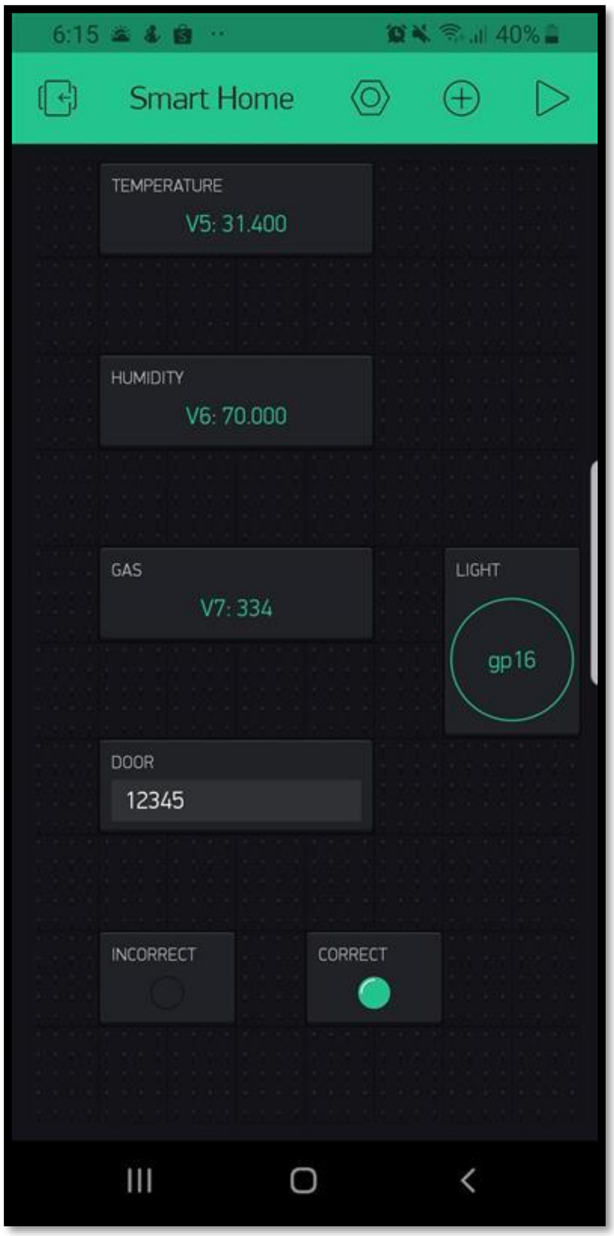
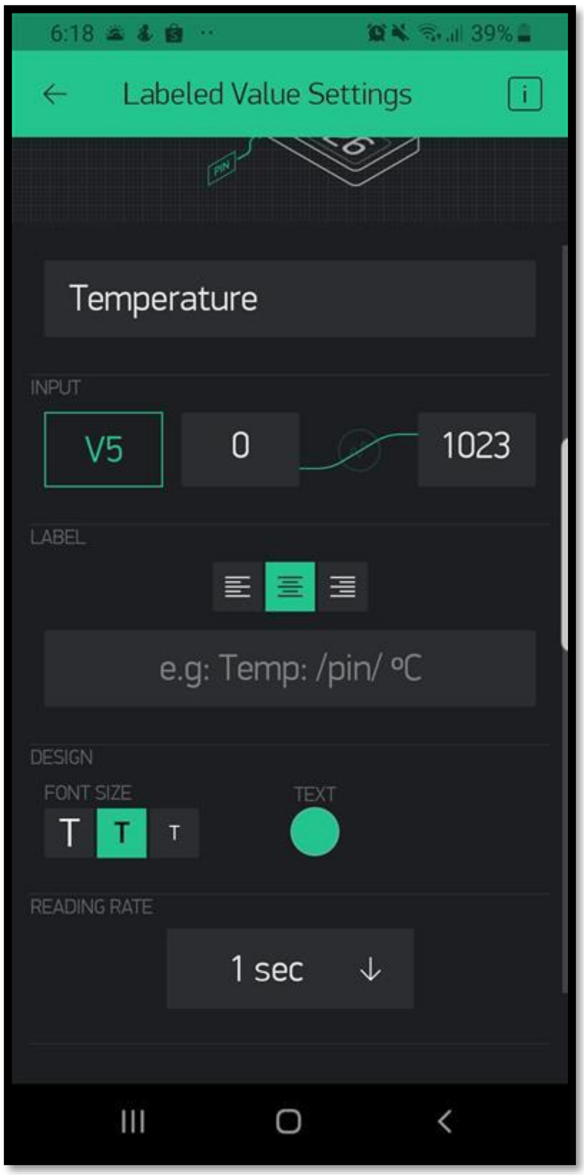


Figure 3.5.6: Smart Home user interface

This is the user interface of the smart home system where the user can control the smart home from anywhere and anytime as long as there is an internet connection.

3.5.1 Variables


6:18 39%

Labeled Value Settings

Temperature

INPUT

V5 0 1023

LABEL

e.g: Temp: /pin/ ºC

DESIGN

FONT SIZE

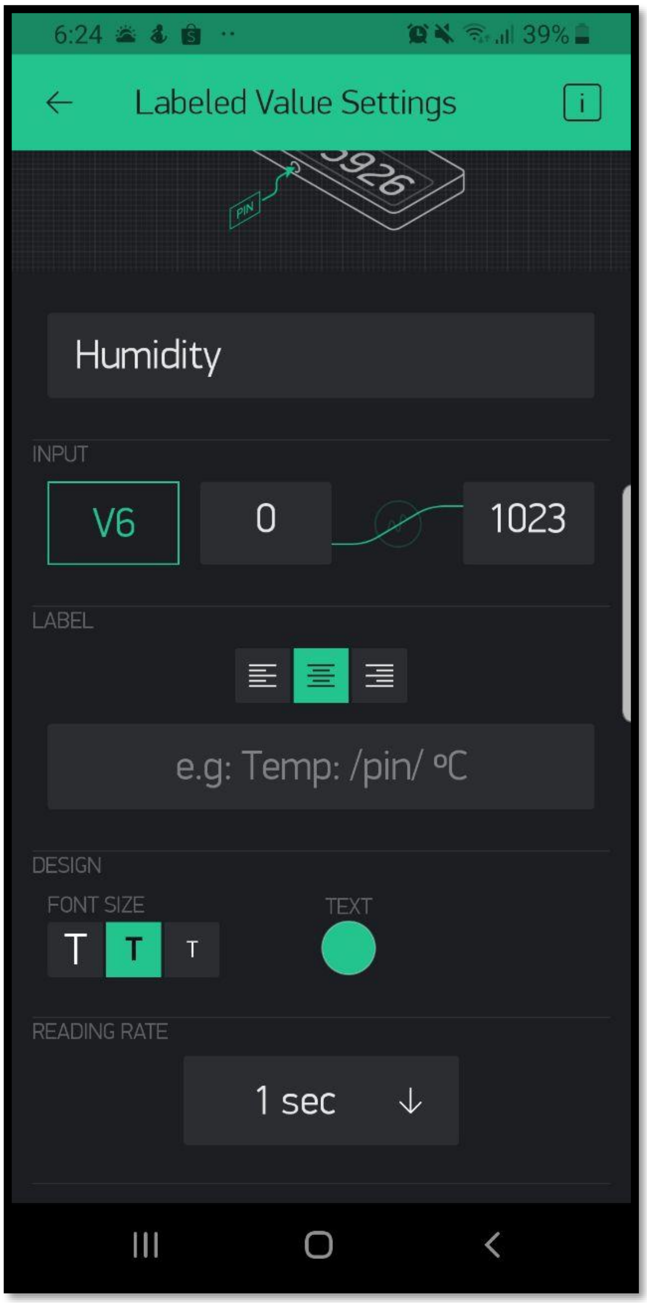
TEXT

READING RATE

1 sec

Figure 3.5.1.1: Labeled value of temperature

These are the settings of the labeled value temperature. The input is virtual pin 5, and it will read the value of temperature every second.


6:24 39%

Labeled Value Settings

Humidity

INPUT

V6 0 1023

LABEL

e.g: Temp: /pin/ ºC

DESIGN

FONT SIZE

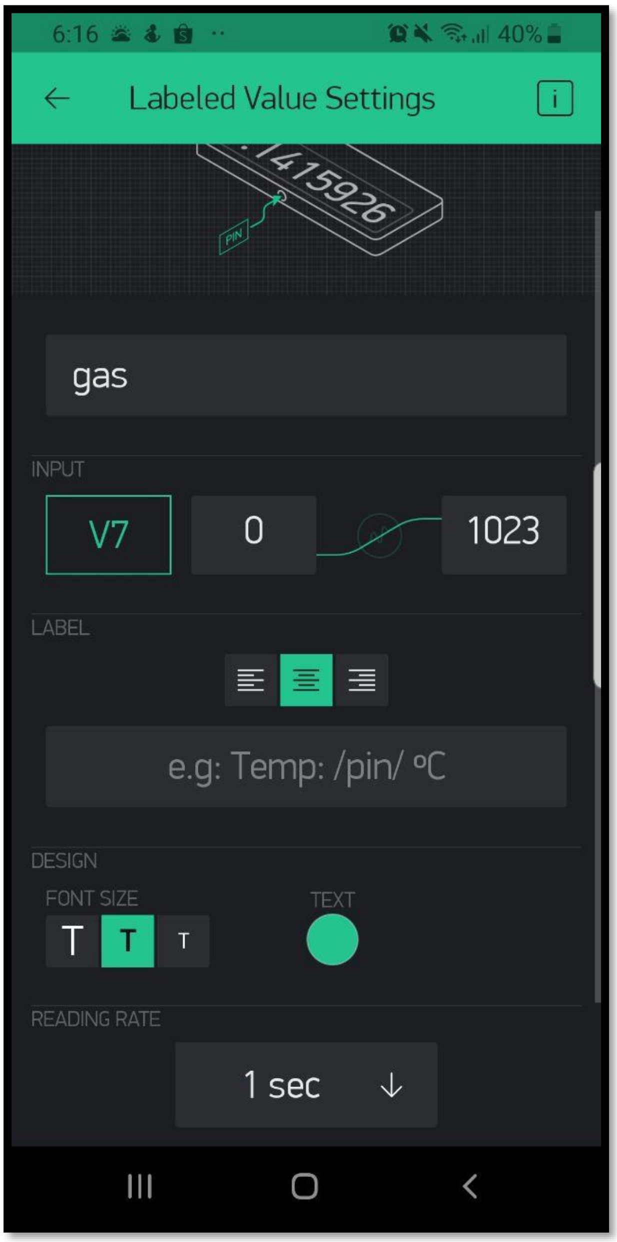
TEXT

READING RATE

1 sec

Figure 3.5.1.2: Labeled value of humidity

These are the settings of the labeled value humidity. The input is virtual pin 6, and it will read the value of humidity every second.



6:16 40%

Labeled Value Settings

gas

INPUT

V7 0 1023

LABEL

e.g: Temp: /pin/ ºC

DESIGN

FONT SIZE T T T

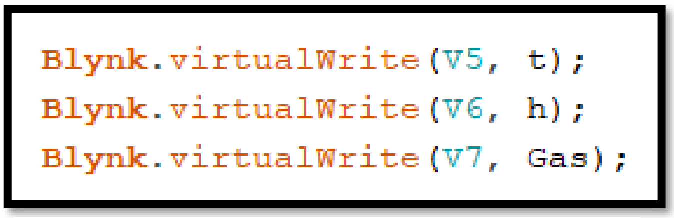
TEXT

READING RATE

1 sec

Figure 3.5.1.3: Labeled value of gas concentration level

These are the settings of the labeled value gas. The input is virtual pin 7, and it will read the gas concentration level every second.

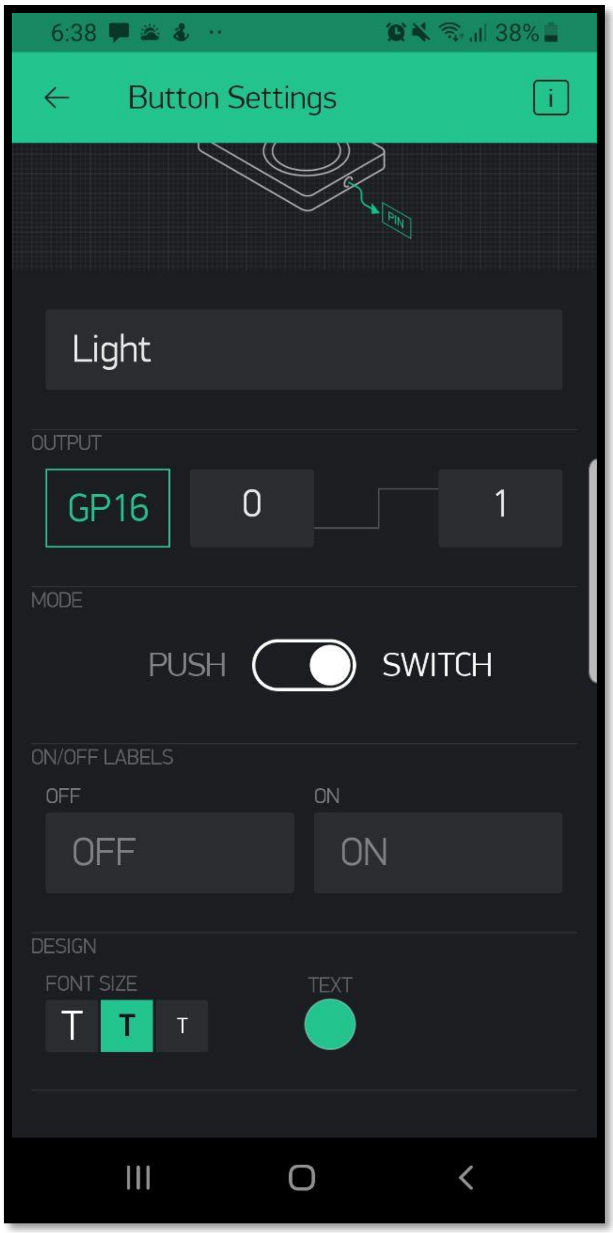


```
Blynk.virtualWrite(V5, t);
Blynk.virtualWrite(V6, h);
Blynk.virtualWrite(V7, Gas);
```

Figure 3.5.1.4: Command in Arduino IDE

This is the command use in Arduino IDE to write the value into the Blynk app.

3.5.2 Light



6:38 38%

Button Settings

Light

OUTPUT

GP16 0 1

MODE

PUSH [✓ SWITCH]

ON/OFF LABELS

OFF: OFF

ON: ON

DESIGN

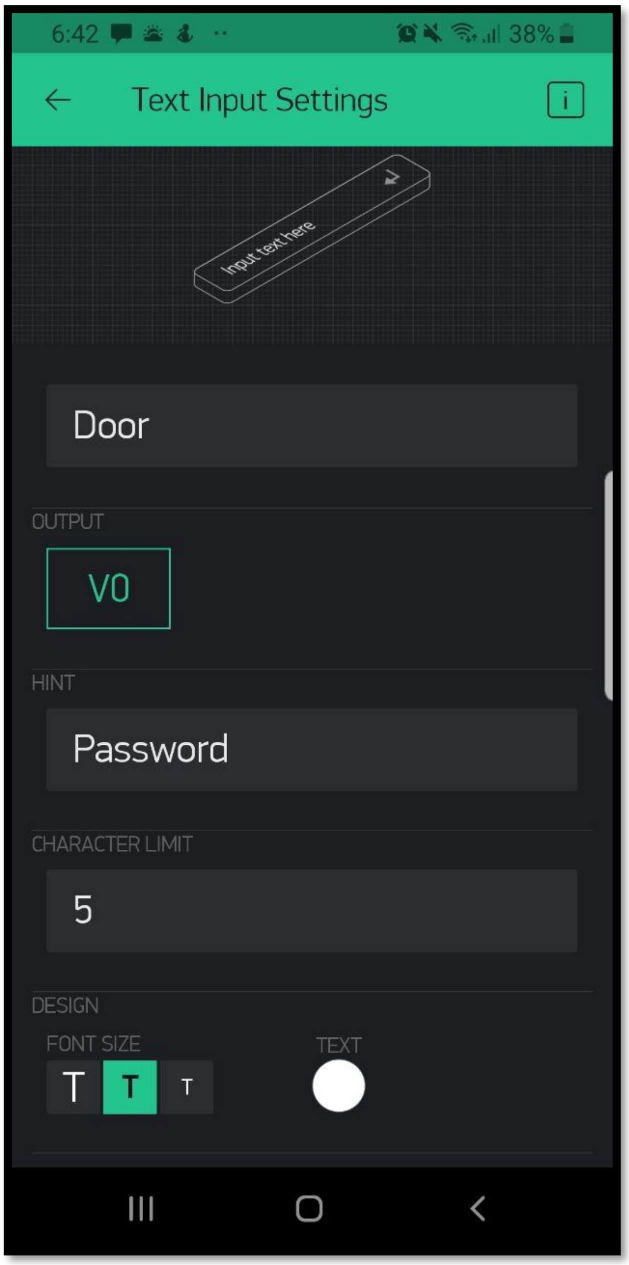
FONT SIZE

TEXT

Figure 3.5.2.1: Light

This is the detailed setting for the button light. There are no specific commands in Arduino IDE to turn on the light. However, in the circuit, the LED must be connected to D0 as it is the GPIO 16.

3.5.3 Door


6:42 38%

Text Input Settings

Door

OUTPUT

V0

HINT

Password

CHARACTER LIMIT

5

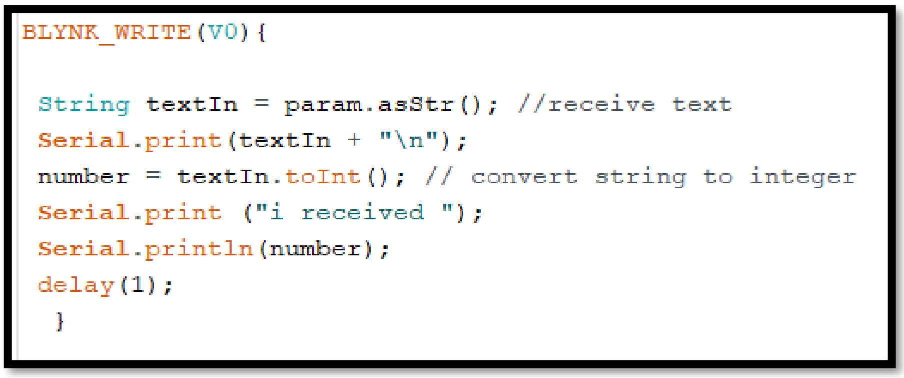
DESIGN

FONT SIZE

TEXT

Figure 3.5.3.1: Door

As mentioned in the project scope, the user can open the door via Blynk app using a password. The virtual pin of this input will be V0. The maximum character limit is 5. The password has been set to “12345”.



```
BLYNK_WRITE(V0){

 String textIn = param.asStr(); //receive text
 Serial.print(textIn + "\n");
 number = textIn.toInt(); // convert string to integer
 Serial.print ("i received ");
 Serial.println(number);
 delay(1);
  }
```

Figure 3.5.3.2: Read password entered from Blynk.

This is the command to show how the Arduino IDE is able to read the input from the Blynk app.

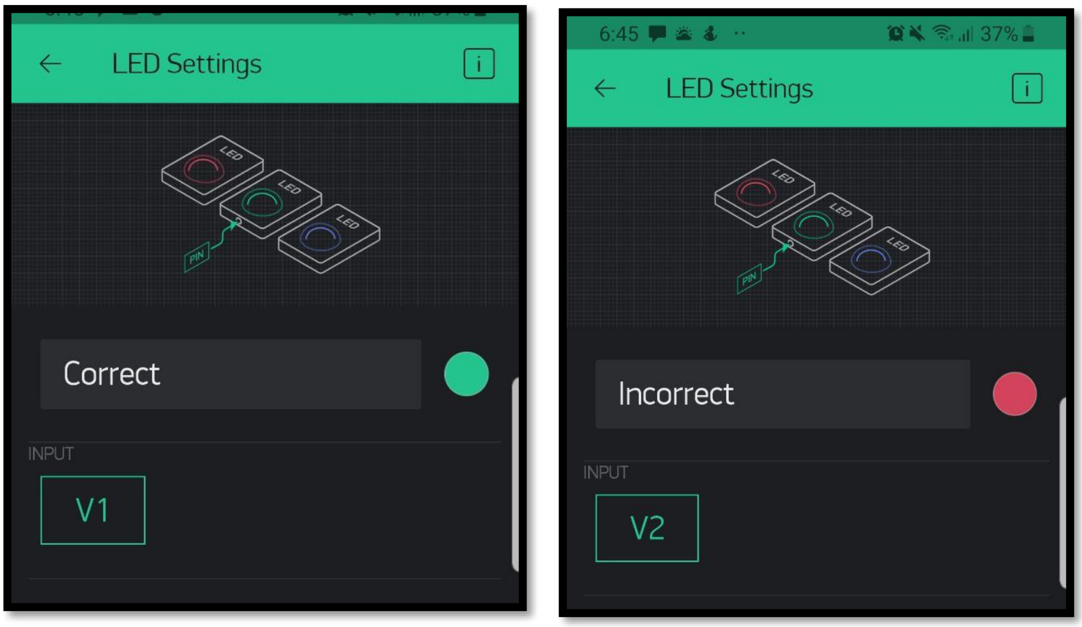
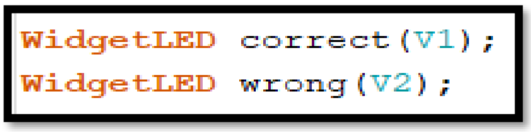


Figure 3.5.3.3: LED indicator

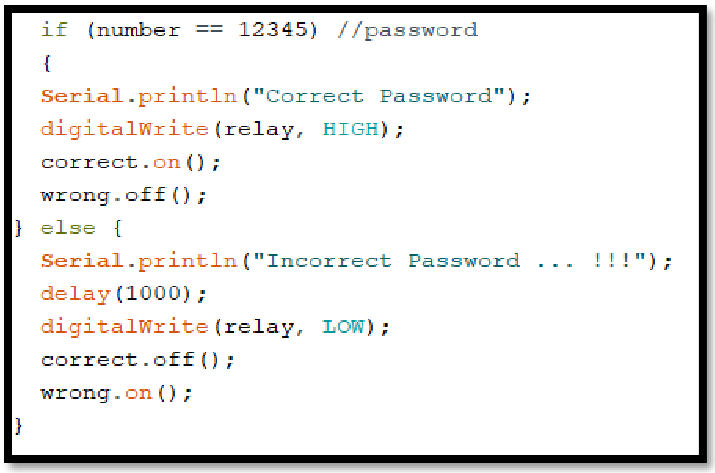
If the password is correct, the Blynk LED indicator will produce green light. The virtual pin will be V1. If the user enters an incorrect password, another Blynk LED indicator will produce a red light.



```
WidgetLED correct(V1);
WidgetLED wrong(V2);
```

Figure 3.5.3.4: Widget LED

Two widgets were created to control the LED from the Arduino IDE.

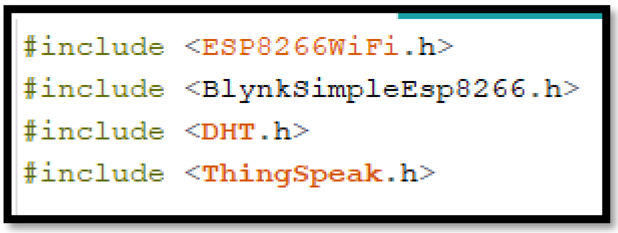


```
  if (number == 12345) //password
  {
  Serial.println("Correct Password");
  digitalWrite(relay, HIGH);
  correct.on();
  wrong.off();
} else {
  Serial.println("Incorrect Password ... !!!");
  delay(1000);
  digitalWrite(relay, LOW);
  correct.off();
  wrong.on();
}
```

Figure 3.5.3.5: Command to determine correct or incorrect password.

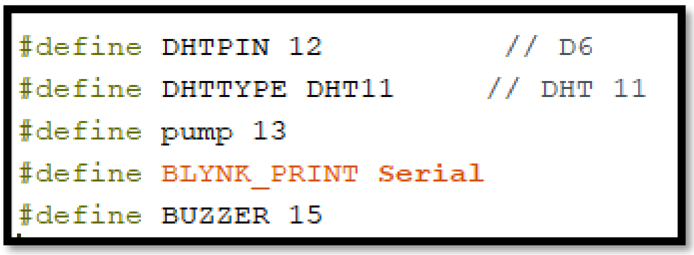
In figure 3.5.3.5, it shows how will the Arduino IDE in the serial monitor will responds if the correct or incorrect password is being entered.

4.0 Arduino IDE Code



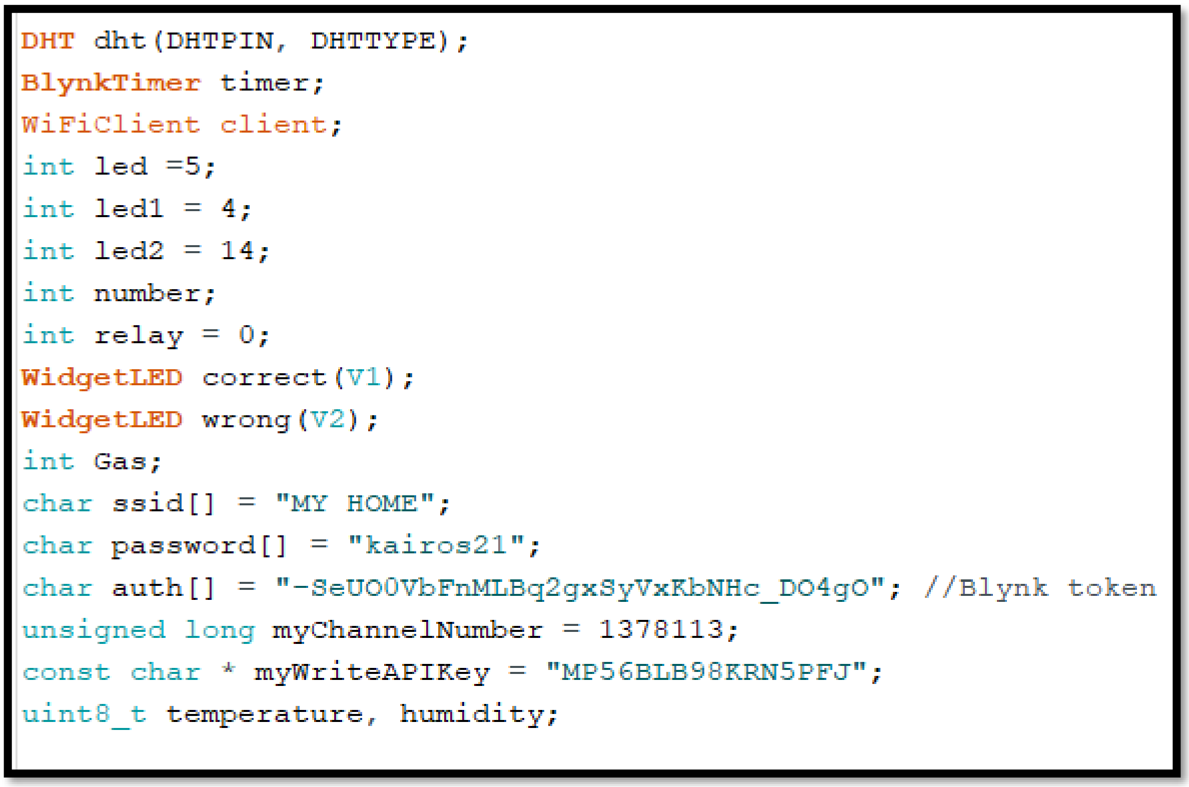
```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
#include <ThingSpeak.h>
```

Figure 4.1: library



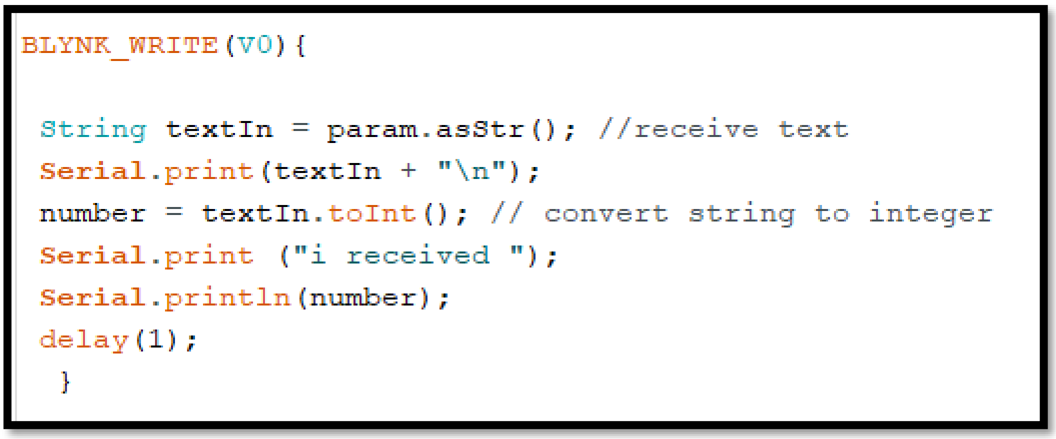
```
#define DHTPIN 12           // D6
#define DHTTYPE DHT11      // DHT 11
#define pump 13
#define BLYNK_PRINT Serial
#define BUZZER 15
```

Figure 4.2: Define



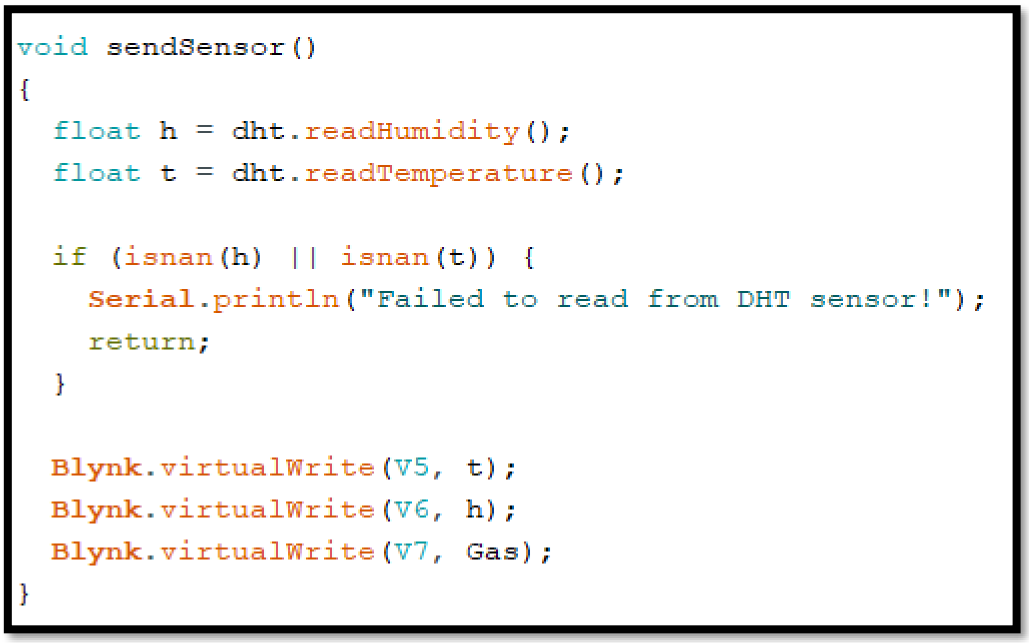
```
DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;
WiFiClient client;
int led =5;
int led1 = 4;
int led2 = 14;
int number;
int relay = 0;
WidgetLED correct(V1);
WidgetLED wrong(V2);
int Gas;
char ssid[] = "MY HOME";
char password[] = "kairos21";
char auth[] = "-SeUO0VbFnMLBq2gxSyVxKbNHc_DO4gO"; //Blynk token
unsigned long myChannelNumber = 1378113;
const char * myWriteAPIKey = "MP56BLB98KRN5PFJ";
uint8_t temperature, humidity;
```

Figure 4.3: Variables



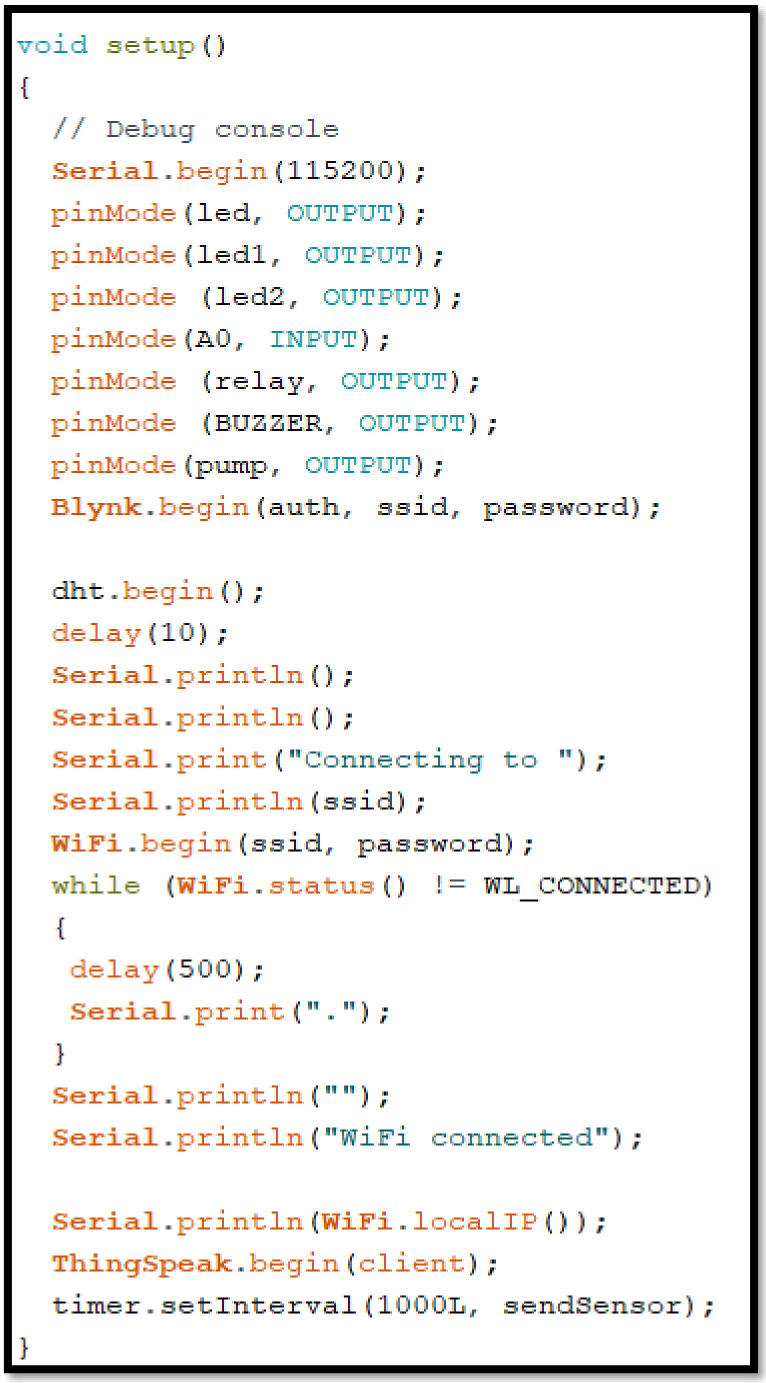
```
BLYNK_WRITE(V0){

 String textIn = param.asStr(); //receive text
 Serial.print(textIn + "\n");
 number = textIn.toInt(); // convert string to integer
 Serial.print ("i received ");
 Serial.println(number);
 delay(1);
  }
```

Figure 4.4: Blynk Write



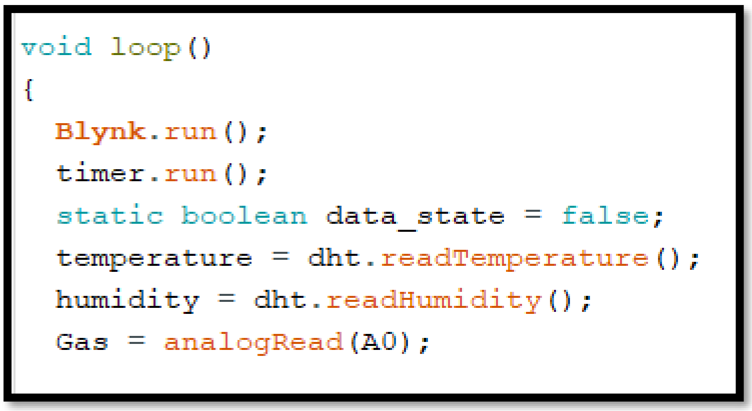
```
void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();

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

  Blynk.virtualWrite(V5, t);
  Blynk.virtualWrite(V6, h);
  Blynk.virtualWrite(V7, Gas);
}
```

Figure 4.5: send Sensor function



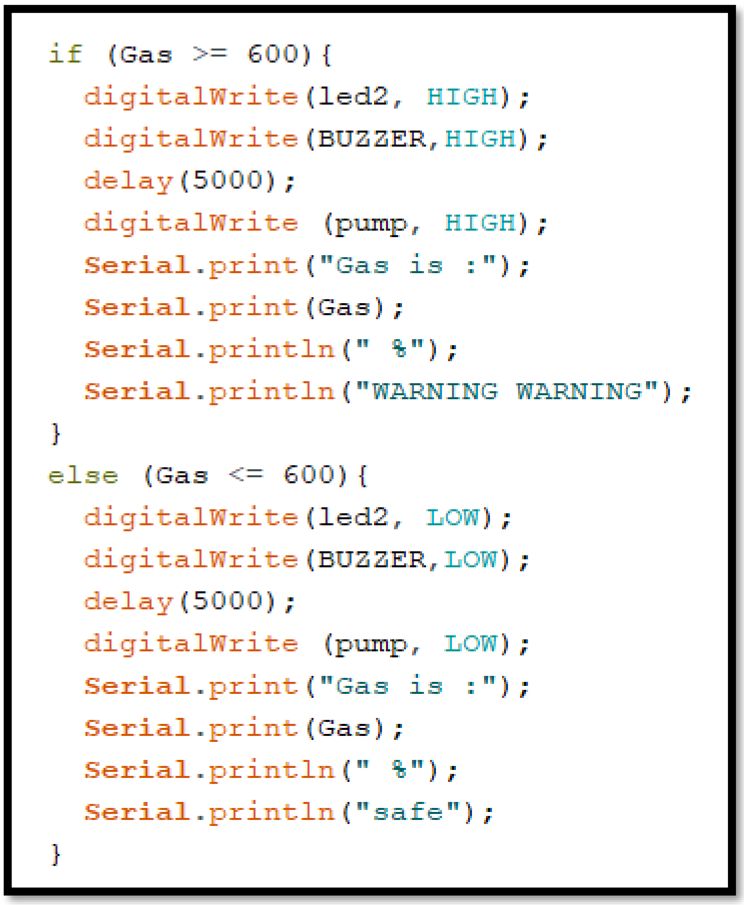
```
void setup()
{
  // Debug console
  Serial.begin(115200);
  pinMode(led, OUTPUT);
  pinMode(led1, OUTPUT);
  pinMode (led2, OUTPUT);
  pinMode(A0, INPUT);
  pinMode (relay, OUTPUT);
  pinMode (BUZZER, OUTPUT);
  pinMode(pump, OUTPUT);
  Blynk.begin(auth, ssid, password);

  dht.begin();
  delay(10);
  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(WiFi.localIP());
  ThingSpeak.begin(client);
  timer.setInterval(1000L, sendSensor);
}
```

Figure 4.6: void setup



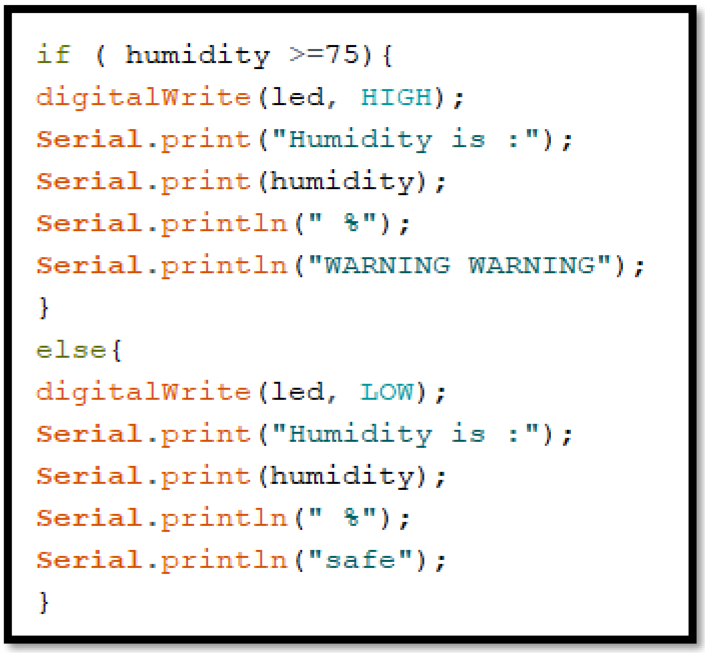
```
void loop()
{
  Blynk.run();
  timer.run();
  static boolean data_state = false;
  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
  Gas = analogRead(A0);
```

Figure 4.7a: void loop part 1



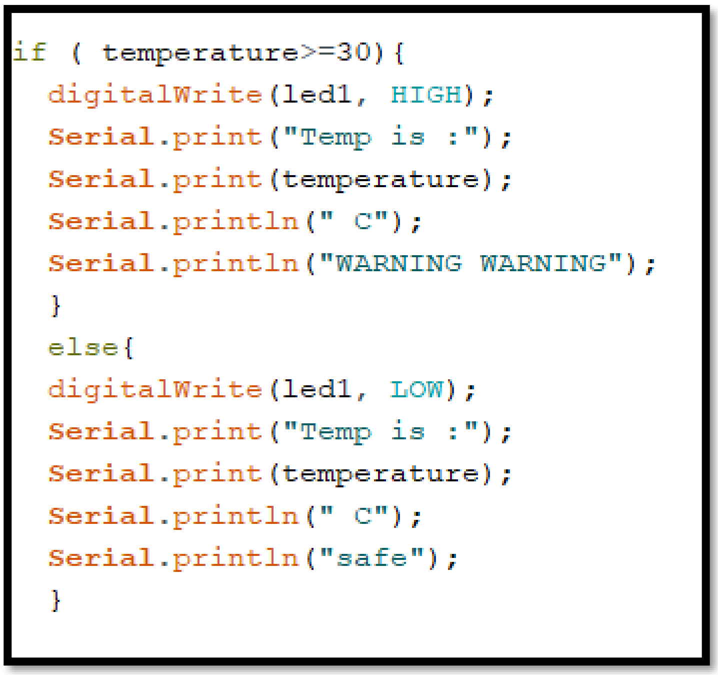
```
 if (Gas >= 600){
   digitalWrite(led2, HIGH);
   digitalWrite(BUZZER,HIGH);
   delay(5000);
   digitalWrite (pump, HIGH);
   Serial.print("Gas is :");
   Serial.print(Gas);
   Serial.println(" %");
   Serial.println("WARNING WARNING");
 }
 else (Gas <= 600){
   digitalWrite(led2, LOW);
   digitalWrite(BUZZER,LOW);
   delay(5000);
   digitalWrite (pump, LOW);
   Serial.print("Gas is :");
   Serial.print(Gas);
   Serial.println(" %");
   Serial.println("safe");
 }
```

Figure 4.7b: void loop part 2



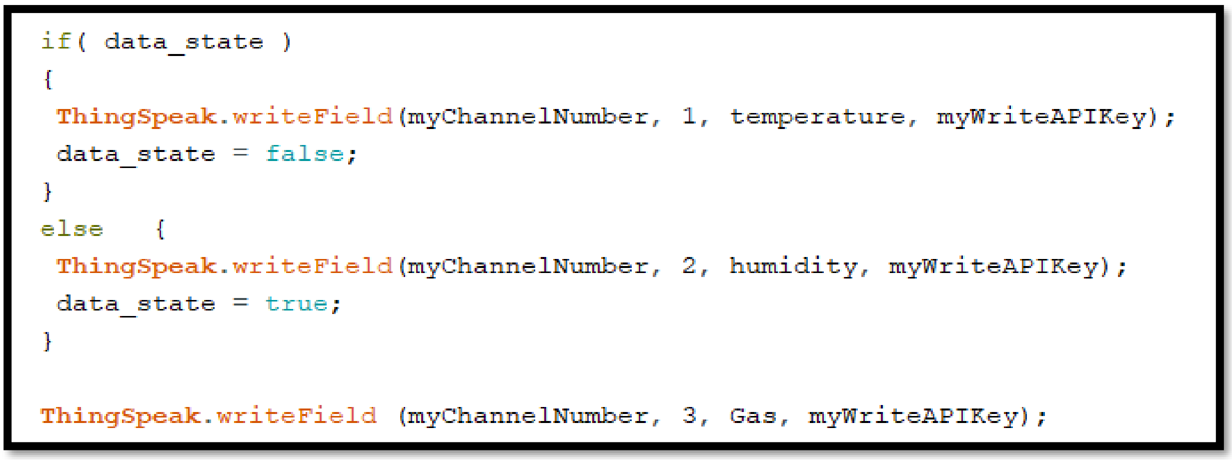
```
if ( humidity >=75){
digitalWrite(led, HIGH);
Serial.print("Humidity is :");
Serial.print(humidity);
Serial.println(" %");
Serial.println("WARNING WARNING");
}
else{
digitalWrite(led, LOW);
Serial.print("Humidity is :");
Serial.print(humidity);
Serial.println(" %");
Serial.println("safe");
}
```

Figure 4.7c: void loop part 3



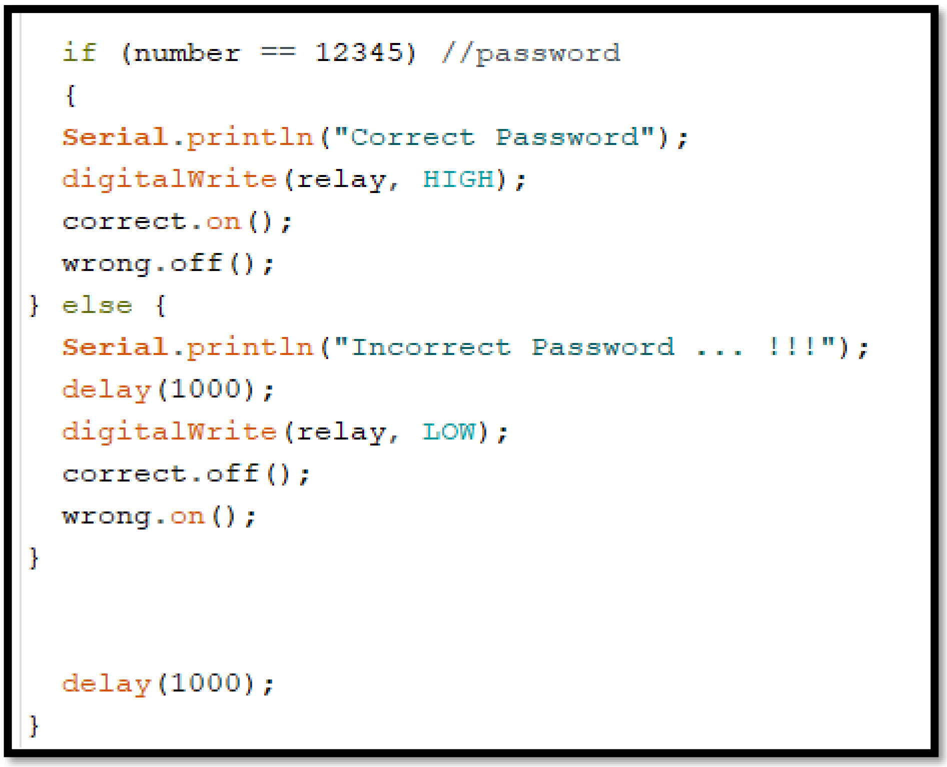
```
if ( temperature>=30){
  digitalWrite(led1, HIGH);
  Serial.print("Temp is :");
  Serial.print(temperature);
  Serial.println(" C");
  Serial.println("WARNING WARNING");
  }
  else{
  digitalWrite(led1, LOW);
  Serial.print("Temp is :");
  Serial.print(temperature);
  Serial.println(" C");
  Serial.println("safe");
  }
```

Figure 4.7d: void loop part 4



```
if( data_state )
{
 ThingSpeak.writeField(myChannelNumber, 1, temperature, myWriteAPIKey);
 data_state = false;
}
else   {
 ThingSpeak.writeField(myChannelNumber, 2, humidity, myWriteAPIKey);
 data_state = true;
}

ThingSpeak.writeField (myChannelNumber, 3, Gas, myWriteAPIKey);
```

Figure 4.7e: void loop part 5



```
  if (number == 12345) //password
  {
  Serial.println("Correct Password");
  digitalWrite(relay, HIGH);
  correct.on();
  wrong.off();
} else {
  Serial.println("Incorrect Password ... !!!");
  delay(1000);
  digitalWrite(relay, LOW);
  correct.off();
  wrong.on();
}


  delay(1000);
}
```

Figure 4.7f: void loop part 6

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
#include <ThingSpeak.h>
#define DHTPIN 12          // D6
#define DHTTYPE DHT11     // DHT 11
#define pump 13
#define BLYNK_PRINT Serial
#define BUZZER 15
DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;
WiFiClient client;
int led =5;
int led1 = 4;
int led2 = 14;
int number;
int relay = 0;
WidgetLED correct(V1);
WidgetLED wrong(V2);
int Gas;
char ssid[] = "MY HOME";
char password[] = "kairos21";
char auth[] = "-SeUO0VbFnMLBq2gxSyVxKbNHc_DO4gO"; //Blynk token
unsigned long myChannelNumber = 1378113;
const char * myWriteAPIKey = "MP56BLB98KRN5PFJ";
uint8_t temperature, humidity;
```

```
BLYNK_WRITE(V0){

 String textIn = param.asStr(); //receive text
 Serial.print(textIn + "\n");
 number = textIn.toInt(); // convert string to integer
 Serial.print ("i received ");
 Serial.println(number);
 delay(1);
  }
void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Blynk.virtualWrite(V5, t);
  Blynk.virtualWrite(V6, h);
  Blynk.virtualWrite(V7, Gas);
}
void setup()
{
  // Debug console
  Serial.begin(115200);
  pinMode(led, OUTPUT);
  pinMode(led1, OUTPUT);
  pinMode (led2, OUTPUT);
  pinMode(A0, INPUT);
  pinMode (relay, OUTPUT);
  pinMode (BUZZER, OUTPUT);
  pinMode(pump, OUTPUT);
  Blynk.begin(auth, ssid, password);
  dht.begin();
  delay(10);
  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(WiFi.localIP());
  ThingSpeak.begin(client);
  timer.setInterval(1000L, sendSensor);
}
```

```
void loop()
{
  Blynk.run();
  timer.run();
  static boolean data_state = false;
  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
  Gas = analogRead(A0);

  if (Gas >= 600){
    digitalWrite(led2, HIGH);
    digitalWrite(BUZZER,HIGH);
    delay(5000);
    digitalWrite (pump, HIGH);
    Serial.print("Gas is :");
    Serial.print(Gas);
    Serial.println(" %");
    Serial.println("WARNING WARNING");
  }
  else (Gas <= 600){
    digitalWrite(led2, LOW);
    digitalWrite(BUZZER,LOW);
    delay(5000);
    digitalWrite (pump, LOW);
    Serial.print("Gas is :");
    Serial.print(Gas);
    Serial.println(" %");
    Serial.println("safe");
  }
  if ( humidity >=75){
  digitalWrite(led, HIGH);
  Serial.print("Humidity is :");
  Serial.print(humidity);
  Serial.println(" %");
  Serial.println("WARNING WARNING");
  }
  else{
  digitalWrite(led, LOW);
  Serial.print("Humidity is :");
  Serial.print(humidity);
  Serial.println(" %");
  Serial.println("safe");
  }

 if ( temperature>=30){
  digitalWrite(led1, HIGH);
  Serial.print("Temp is :");
  Serial.print(temperature);
  Serial.println(" C");
  Serial.println("WARNING WARNING");
  }
  else{
  digitalWrite(led1, LOW);
  Serial.print("Temp is :");
  Serial.print(temperature);
  Serial.println(" C");
  Serial.println("safe");
  }
  if( data_state )
  {
   ThingSpeak.writeField(myChannelNumber, 1, temperature, myWriteAPIKey);
   data_state = false;
  }
  else   {
   ThingSpeak.writeField(myChannelNumber, 2, humidity, myWriteAPIKey);
   data_state = true;
  }

  ThingSpeak.writeField (myChannelNumber, 3, Gas, myWriteAPIKey);

  if (number == 12345) //password
  {
  Serial.println("Correct Password");
  digitalWrite(relay, HIGH);
  correct.on();
  wrong.off();
} else {
  Serial.println("Incorrect Password ... !!!");
  delay(1000);
  digitalWrite(relay, LOW);
  correct.off();
  wrong.on();
}


  delay(1000);
}
```

5.0 Video Link

To see the demonstration of this smart home project, please click the link below:

https://cloudmails.sharepoint.com/:v:/s/kin/EXmJkpVHsmZGsEOEOvnEKnEBorF7NgPJVKNJFGJ_hTFtzw?e=esFkg9

6.0 Conclusion

This project is a simulation on how a smart home will function. All in all, the project was successful as it meets all the project scope that was set earlier. Smart LED, Smart Door, Smart Fan, Smart Air Conditioner and Smart Water Pump are configured, and everything works perfectly. Smart home has a great potential to grow in the future. It is able to provide users with security, convenience and even energy tracking capability. Although there are several concerns such as security and privacy, but with proper and ethical development of IoT Smart Home environment, these concerns will be solved.

6.1 Limitation

There are several limitations in this project that prevent the project from becoming from advanced.

1. Microcontroller

   ESP8266 is inferior compare ESP32. ESP32 has a Bluetooth module and more GPIO pins than ESP8266. Having more GPIO pins will allow more components to be connected. For example, more smart LEDs can be configured to show how the user can control individual lights in the house. Not only that, ESP32 has other features such as touch sensor that can provide a more complex project.

2. Blynk app

   Blynk app provides a free version that is rather limited. It provides the user with 2000 coin that the user can use. Having more coin will allow the user to use more widgets. For example, another button can be configured where when the user presses this button , the water hose will water the plants in the house. Having this functionality will provide users with extra comfort and convenience.

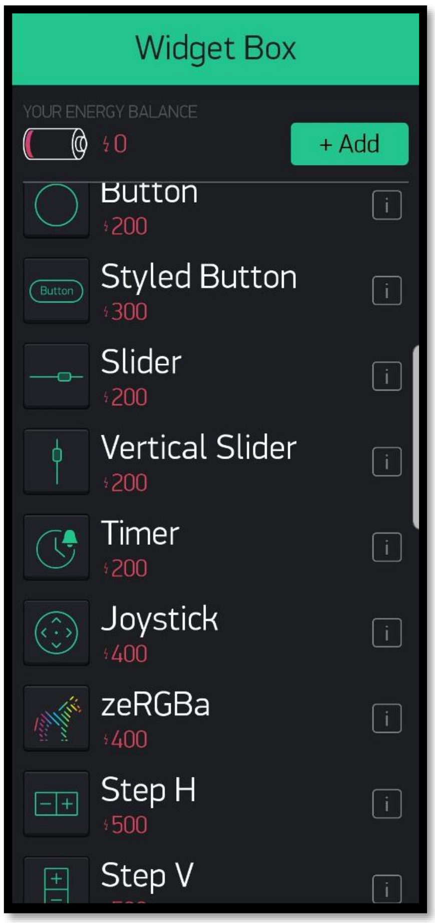


Figure 6.1.1: Widget Boxes

3. Instead of using LED to indicate the fan, a real fan can be used. This will provide a better visualization on how the smart home will operate.
4. When there is a fire in the house, it is not really effective to use a hose. Instead of using a hose to put out the fire, a sprinkler or sprayer might be the better option.
5. Security is not emphasized throughout this project. Although a remote-controlled solenoid door lock has been configured, the password “12345” is easily identified by potential attackers. There is no proper authentication and authorization security mechanism has been applied.
6. Connectivity can be an issue. Connection between 3 parties which are the circuit, the code in Arduino IDE and the Blynk app was proven to be rather unstable despite having a strong internet connection.
7. Data analysis has not been done at all.

6.2 Future Enhancement

These are several future enhancements that can be made into this project.

1. Switch microcontroller ESP8266 to ESP32.
2. To expand this project, more coins should be purchased to control more components remotely.
3. Use a real fan instead of LED to indicate the fan.
4. Switch mini water pump to a water sprinkler.
5. Provide another layer of security mechanism.
6. Having a stronger internet connection can be an ideal solution.
7. Importing the data gathered from the ThingSpeak or Microsoft Excel to a data analysis software would be ideal. An example of a data analysis software can be R-Project where it used R-programming language to provide a thorough analysis.

7.0 Reference

Administrator. (2018). *Role of Wi-Fi in IoT.* [Online]. Available from: https://www.wifiattendance.com/blog/role-of-wifi-in-iot/ [Accessed: 7 May 2021]

Ali, Y. (2020). *What is ThingSpeak?* [Online]. Available from: http://tecmafia.com/what-is-thingspeak/ [Accessed: 7 May 2021]

Aqeel, A. (2018). *Introduction to Arduino IDE.* [Online]. Available from: https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html [Accessed: 3 May 2021]

Babun, L. et al (2021). A survey on IoT platforms: Communication, security, and privacy perspectives. *Computer Networks.* [Online]. 192 Available from: https://www-sciencedirect-com.ezproxy.apiit.edu.my/science/article/pii/S1389128621001444 [Accessed: 13 May 2021]

Bruce, O. (2012). *What are the advantages of using a line graph to represent data?* Available from: https://www.enotes.com/homework-help/what-advantages-using-line-graph-represent-data-514721 [Accessed: 3 May 2021]

Campbell, S. (2015). *HOW TO SET UP THE DHT11 HUMIDITY SENSOR ON AN ARDUINO.* [Online]. Available from: https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/ [Accessed: 5 May 2021]

Excelhelp. (2021). The History of Microsoft Excel. [Online]. Available from: https://www.excelhelp.com/the-history-of-microsoft-excel/ [Accessed: 3 May 2021]

George, L. (2020). *What is a Microcontroller?* [Online]. Available from: https://electrosome.com/microcontroller/ [Accessed: 3 May 2021]

Han, M.J.N., Kim, M.J. & Kim, H.I. (2021). Exploring the user performance of Korean women in smart homes with a focus on user adoption. *Journal of Building Engineering.*[Online]. 39. Available from: https://www-sciencedirect-com.ezproxy.apiit.edu.my/science/article/pii/S2352710221001595 [Accessed: 7 May 2021]

Holmes, K. (2020). *Everything about Blynk (Review, Examples, Tutorials and Alternatives) [2020].* [Online]. Available from: https://outwittrade.com/blynk-alternatives-review-examples-tutorials/ [Accessed: 5 May 2021]

Islam, R. et al (2021). LoRa and server-based home automation using the internet of things (IoT). *Journal of King Saud University- Computer and Information Sciences.* [Online].

Corrected Proof Available from: https://www-sciencedirect-com.ezproxy.apiit.edu.my/science/article/pii/S1319157820306285 [Accessed: 13 May 2021]

Maravedis. (2020). *The Role of Wi-Fi in IoT.* [Online]. Available from: https://www.iotforall.com/wifi-role-iot [Accessed: 7 May 2021]

Morgan, L. (2018). *What Is Cloud Storage & How Does It Work?* [Online]. Available from: https://www.enterprisestorageforum.com/cloud/cloud-storage/ [Accessed: 7 May 2021]

Mukherjee, A. (2016). *Smoke Detection using MQ-2 Gas Sensor.* [Online] Available from: https://create.arduino.cc/projecthub/Aritro/smoke-detection-using-mq-2-gas-sensor-79c54a [Accessed: 7 May 2021]

Naveenraj, R. (2020). *Arduino IDE – Complete guide to setup and get started.* [Online]. Available from: https://technobyte.org/arduino-ide-complete-guide-beginners/ [Accessed: 3 May 2021]

Niv, I. (2019). *ESP8266 - Beginner Tutorial + Project.* [Online]. Available from: https://create.arduino.cc/projecthub/Niv_the_anonymous/esp8266-beginner-tutorial-project-6414c8 [Accessed: 5 May 2021]

Queralta, J.P. et al (2019). Comparative Study of LPWAN Technologies on Unlicensed Bands for M2M Communication in the IoT: beyond LoRa and LoRaWAN. *Procedia Computer Science.* [Online]. 155 p.343-350. Available from: https://www-sciencedirect-com.ezproxy.apiit.edu.my/science/article/pii/S1877050919309639 [Accessed: 13 May 2021]

Ruiz, I.L. & Gomez-Nieto, M.A. (2017). Combining of NFC, BLE and Physical Web Technologies for Objects Authentication on IoT Scenarios. *Procedia Computer Science.* [Online]. 109 p.265-272. Available from: https://www-sciencedirect-com.ezproxy.apiit.edu.my/science/article/pii/S1877050917310190 [Accessed: 13 May 2021]

Sharief, K. (2020). *What is Sensor? – Definition, Functions, Types, Characteristics, and More.* [Online]. Available from: https://www.computertechreviews.com/definition/sensor/ [Accessed: 5 May 2021]

Sovacool, B.K., Martiskainen, M. & Del Rio, D.D.F. (2021). Knowledge, energy sustainability, and vulnerability in the demographics of smart home technology diffusion. *Energy Policy.* [Online]. Available from: https://www-sciencedirect-com.ezproxy.apiit.edu.my/science/article/pii/S0301421521000653 [Accessed: 7 May 2021]

Technical Editor. (2017). *Advantages of disadvantages of microcontroller*. [Online]. Available from: https://www.polytechnichub.com/advantages-disadvantages-microcontroller/ [Accessed: 3 May 2021]

Teja, R. (2021). *Basics of Embedded C Program*. [Online]. Available from: https://www.electronicshub.org/basics-of-embedded-c-program/ [Accessed: 3 May 2021]

Zahmatkesh, H. & Al-Turjman, F. (2020). Fog computing for sustainable smart cities in the IoT era: Caching techniques and enabling technologies - an overview. *Sustainable Cities and Society*. [Online]. 59. Available from: https://www-sciencedirect-com.ezproxy.apiit.edu.my/science/article/pii/S2210670720301268 [Accessed: 13 May 2021]

Zamanloo, S. et al. (2021). Optimal two-level active and reactive energy management of residential appliances in smart homes. *Sustainable Cities and Society.*[Online]. 71. Available from: https://www-sciencedirect-com.ezproxy.apiit.edu.my/science/article/pii/S2210670721002584. [Accessed: 7 May 2021]