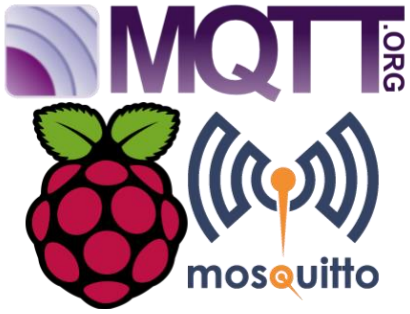
An-Najah National University

Faculty of Engineering and Information Technology

Computer Engineering Department

Graduation Project 2 Report

Home Automation System with MQTT



Supervisor : Dr.Rae'd Al-Qadi

Prepared by :

Azza Abdulghani & Raghad Al-Sayeh

23rd of December , 2018

"Presented in partial fulfillment of the requirements for Bachelor degree in Computer Engineering "

Acknowledgment

Thanks to God first and last. This project could not be achieved without dedicated support from people around us. We would thank our Supervisor Dr.Rae’d Al-Qadi for his continuous support and guidance. And, A lot of thanks to our parents , who keep on supporting us and encouraging us to reach our dreams.

We would thank all teachers in Computer Engineering Department at An-Najah National University for their assistance and unconditional bestowal through the last 4 years. Finally we thank Hani Qaddumi Scholarship Foundation (HQSF) for their financial support.

Disclaimer statement

This report was written by Azza Abdulghani and Raghad Al-Sayeh at the Computer Engineering Department, Faculty of Engineering, An-Najah National University. It has not been altered or corrected, other than editorial corrections, as a result of assessment and it may contain language as well as content errors. The views expressed in it together with any outcomes and recommendations are solely those of the students. An-Najah National University accepts no responsibility or liability for the consequences of this report being used for a purpose other than the purpose for which it was commissioned.

Table of Contents

Table of Figures :

Abstract

Home automation in general designed for controlling devices and appliances in home remotely and automatically from a mobile phone or any PC. Home automation systems become so popular in recent years due to many reasons like indoor and outdoor security, energy efficiency and it is considered as cost effective way.

In this project, the Home Automation System is based on MQTT protocol (IoT protocol) and a Mosquitto based MQTT broker is established for remote monitoring and control. MQTT will be implemented on ESP8266, a Wi-Fi based development board, that controls the major home appliances such as lights, temperature, humidity, gas detection, and garage door, with sensors and actuators.

Reasons behind using this protocol for faster communication, safe, secure, works on top of TCP/IP protocol.

Keywords: Message Queuing Telemetry Transport (MQTT), Internet of Things (IOT), ESP8266, Mosquitto.

Chapter 1:Introduction

1.Statement of the problem

With the rapid development of different areas of life, the need arose to keep abreast of these developments, follow them and take advantage of their advantages and services of all kinds. There is therefore a need to use technology and wireless connections related to these developments. Hence the idea to create or develop a system helps to monitor and control things around us remotely.

Home Automation System is one of the most important technologies, as it needs to keep people safe and comfortable. Therefore, all the comforts of people in their homes should be provided in a safe and cost-effective way with the latest technology.

2. Objectives of the work

The main objectives of this project are firstly to control all home appliances remotely using IoT protocols and to keep up with home environment information such as temperature and humidity. Secondly, to implement this system with low power consumption and remotely in safe way.

3.Scope of the work

Home Automation system deals with wireless connections and boards using MQTT protocol. It designed for people who want to monitor their homes and want to take continuous information about their home environment.

4. Significance or importance of your work

The importance of our project comes from using IoT protocols. Internet of things can be considered as a network of devices that are wirelessly connected so that they communicate and organize themselves based on the predefined rules. Hence light weight protocols such as MQTT will be used for the data transmission over wireless connectivity.

5.Organization of the report:

Organization of the report is as follows:

Chapter one: Introduction

This chapter consists of introduction of our project, purposes and goals of it, also explain who can use and benefit from it and finally the importance of this project.

Chapter two: Standards and earlier course work

This chapter contains some standards, Materials and codes that we use through the development process of the system.

Chapter three: Literature Review

This section shows some previous systems that are a little similar in the idea of our project, some comparisons between their features and services and our system features.

Chapter four: Methodology

This part discuss in details the methods, tools, Materials and Codes that is used to implement the system.

Chapter five: Results and Analysis

This chapter presents the results of the system, and its features and services that are provided to the user.

Chapter six: Discussion

This section summarizes the aims, process of the application from the start point to the end take with considerations limitations and also final results.

Chapter seven: Conclusion and Recommendation

Finally this part concludes the final outputs of the application, future work that can make to build on it to improve application more and more maybe in most cost-effective ways.

Chapter 2: Standards and Earlier course work

2.1 Standards

This project is mainly implemented on top of MQTT protocol :

- **MQTT Protocol**

Message Queuing Telemetry Transport (MQTT) is a light weight transport protocol that efficiently uses the network bandwidth with a 2 byte fixed header. MQTT works on TCP and assures the delivery of messages from node to the server. Being a message oriented information exchange protocol; MQTT is ideally suited for the IoT nodes which have limited capabilities and resources. MQTT was initially developed by IBM in 1999 and recently has been recognized as standard by Organization for the Advancement of Structured Information Standards (OASIS).

MQTT is a publish/subscribe based protocol. Any MQTT connection typically involves two kinds of agents: MQTT clients and MQTT public broker or MQTT server. Data that is being transported by MQTT is referred to as application message. Any device or program that is connected to the network and exchanges application messages through MQTT is called as an MQTT client. MQTT client can be either publisher or subscriber. A publisher publishes application messages and subscriber requests for the application messages. MQTT server is a device or program that interconnects the MQTT clients. It accepts and transmits the application messages among multiple clients connected to it. Devices such as sensors, mobiles etc. are considered as MQTT client. When an MQTT client has certain information to broadcast, it publishes the data to the MQTT broker. MQTT broker is responsible for data collection and organization. The application messages that are published by MQTT client is forwarded to other MQTT clients that subscribe to it. MQTT is designed to simplify the implementation on client by concentrating all the complexities at the broker. Publisher and subscriber are isolated, meaning they need not have to know the existence or application of other. Before transmitting the application messages, control packets are exchanged based on the QoS associated with them. An MQTT control packet consists of a fixed header, a variable header and payload. CONNECT, CONNACK, PUBLISH, PUBACK, PUBREC, PUBREL, SUBSCRIBE, SUBACK, etc. are some of the MQTT control packets [4] exchanged between MQTT clients and MQTT server. ”Topic” in MQTT provides the routing information. Each topic has a topic name and topic levels associated with it. There may be multiple topic levels separated by / in a topic tree. Wildcard characters such as # and + are used to match

multiple levels in a topic. Featuring the queuing system, MQTT server buffers all the messages if client is offline and delivers them to the client when the session is enabled.

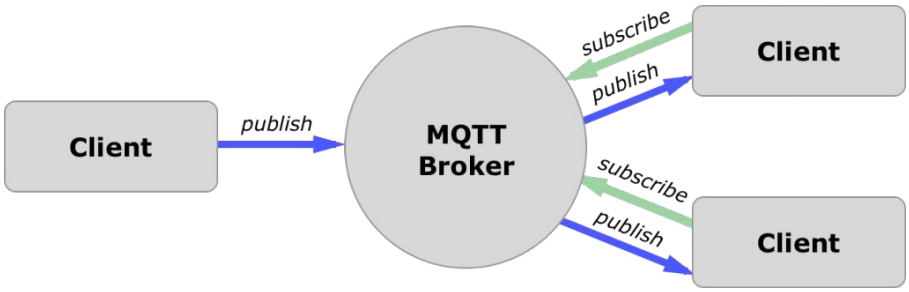


Figure 1 Mqtt protocol

The MQTT protocol is based on TCP/IP. Both the client and the broker need to have a TCP/IP stack.

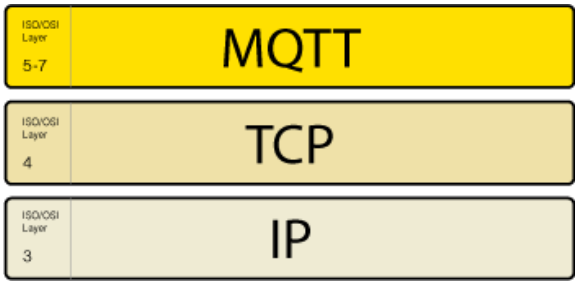


Figure 2 MQTT on TCP/IP

A. Establishing a connection:

Upon the successful establishment of network between the MQTT client and the MQTT server, control packets are exchanged between the client and the server. The client that wishes to connect to the MQTT server sends a CONNECT packet to the server specifying its identifier, flags, protocol level and other fields. The server acknowledges the client with the specified identifier through CONNACK packet with a return code denoting the status of connection.

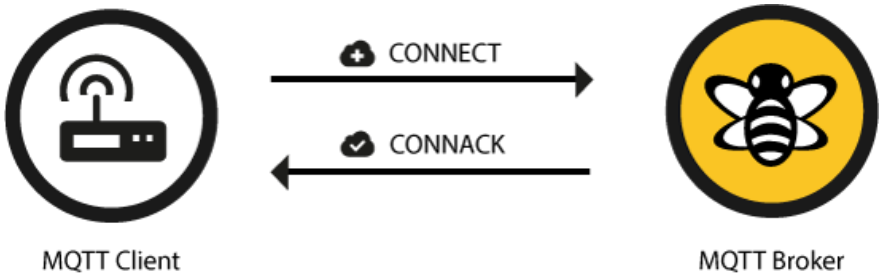


Figure 3 Establish a connection on MQTT

B. Publishing the application messages:

If the client desires to be a publisher, it sends a PUBLISH packet to the server. This packet contains details about the QoS level of transmission, topic name, payload, etc. MQTT supports three levels of Quality of Service (QoS) [5] to the client. If the application messages are transmitted at QoS 0, the client does not receive any acknowledgment for the published packet. For QoS 1, the server acknowledges the published packet with PUBACK including the packet identifier. However in QoS 2, four packets are exchanged. The server acknowledges the receipt of PUBLISH packet with the PUBREC packet. MQTT client then sends a packet to release publish with a PUBREL packet. The server then sends the fourth packet PUBCOMP, indicating the completion of publishing the application message on the given topic.

C. Subscribing to a topic

If the MQTT client want to subscribe to the application messages published on topic, it sends the SUBSCRIBE packet along with the topic name indicated in UTF-8 encoding. The server acknowledges the subscription with SUBACK packet along with a return code denoting the status of request. Once the subscription is successful, the application messages on the specified topic are forwarded to the client with the maximum QoS. To unsubscribe a topic, the client sends an UNSUBSCRIBE packet to the server which acknowledges it with the UNSUBACK packet.

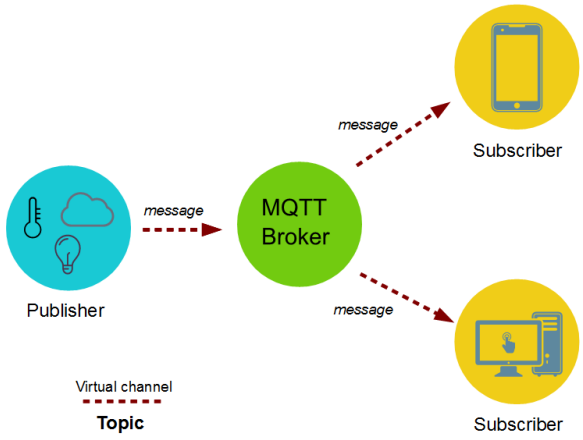


Figure 4 MQTT Publisher and Subscriber

D. MQTT Broker :

There are several brokers can be used. In this home automation project, we use the Mosquitto broker which is installed on the Raspberry Pi.

Figure 5 Mosquitto broker logo

2.2 Earlier Work

Earlier of this project, we start learning about IoT protocols and focusing on MQTT protocol. We understand all parts of the protocol and how it works. Then , we start with Raspberry pi and turn it on. After that, we connect all devices together and write codes to communicate between each other using MQTT.

Chapter 3: Literature Review

Home automation system build using many methods and techniques such as Bluetooth-based home automation, home automation without any protocol(not real IOT), Zigbee-based home automation using cell phones and other types but the mentioned methods were the most common ones.

Bluetooth-based home automation the home devices connected to Arduino BT board at input output ports and it is implemented using C language of microcontrollers, the connection is made through Bluetooth. The connection is established between Arduino BT and phone and it is portable, the circuit is structured for receiving feedback from the phone which gives an indication for the state of the device.

But there are some disadvantages for this system that the user cannot get the required quality devices for setting up the smart phone via Bluetooth, also there is a Bluetooth communication range issue between only 10-20 meters, and if the Bluetooth is disconnected this cause the application also disconnects.

Home automation without real IOT protocol, this is an ordinary automation system without organized protocol provides control to home devices with no management. Some experts don't consider it as IoT due to lack of the real concept of IOT that based on the interaction between the server (controller) that make a distribution and the multiple clients (home devices/appliances).

Zigbee-based home automation, for the purpose of monitoring and controlling home appliances Zigbee is used. The device performance is record by network coordinators, it uses the four switch port standard wireless ADSL router and it passed the network SSID and security Wi-Fi by the virtual home algorithm. This type of home automation provides safety of all messages that received by the algorithm. On the other hand it has some problems like it is not secure, replacement cost very high when some defect happens, also it is prone to attack from any unauthorized people, finally it has low transmission rate which make the operation difficult in case of multiple clients.

Chapter 4 : Methodology:

4.1 Tools

4.1.1. Raspberry pi 3 model B+

The raspberry pi 3 b+[1] is the latest product in the raspberry pi 3 range, boasting an updated 64-bit quad core processor running at 1.4GHz with built-in metal heat sink, dual-band 2.4GHz and 5GHz wireless LAN, faster (300 mbps) Ethernet, and PoE capability through a separate PoE HAT.

It has also on board Bluetooth 4.2 HS low-energy (BLE) (CYW43455), 4xUSB 2.0 ports,40 GPIO pins, full size HDMI 1.3a port, combined 3.5mm analog audio and composite video jack, camera interface (CSI), display interface (DSI) , microSD slot, VideoCore IV multimedia/3D graphics core @ 400MHz/300MHz.

It is faster than previous versions such as Raspberry pi 3 model b and raspberry pi 2 from many aspects. The first one is faster processing improved thermals on the Pi 3 B+ means that the CPU on the BCM2837 SoC can now run at 1.4GHz, a 17% increase on the previous Pi 3 model (which ran at 1.2GHz).

Video performance on Pi 3 B+ is similar to the previous version Pi 3,the VideoCore being clocked at 400MHz for video processing and the 3D graphics processor running at 300MHz.

The second aspect is faster wireless, a significant change on the Pi 3 B+ compared to the Pi 3 is the inclusion of a new faster, dual-band wireless chip (CYW43455) with 802.11 b/g/n/ac wireless LAN and Bluetooth 4.2.The dual-band 2.4GHz and 5GHz wireless LAN enables faster networking with less interference (although the higher bandwidth has less range), and the new PCB antenna technology should allow better reception. Bluetooth allows you to use a wireless keyboard/trackpad without extra devices.

Also there is a faster Ethernet, the Pi 3 B+ has significantly faster wired networking, thanks to an upgraded USB/LAN chip, and you should see speeds that are 3-5x faster than on previous models of the Pi, at least 300Mbit/s. Besides it also has many advantages in its architecture like GPIO and layout.

It is used as a MQTT broker (server) to communicate with multiple clients.

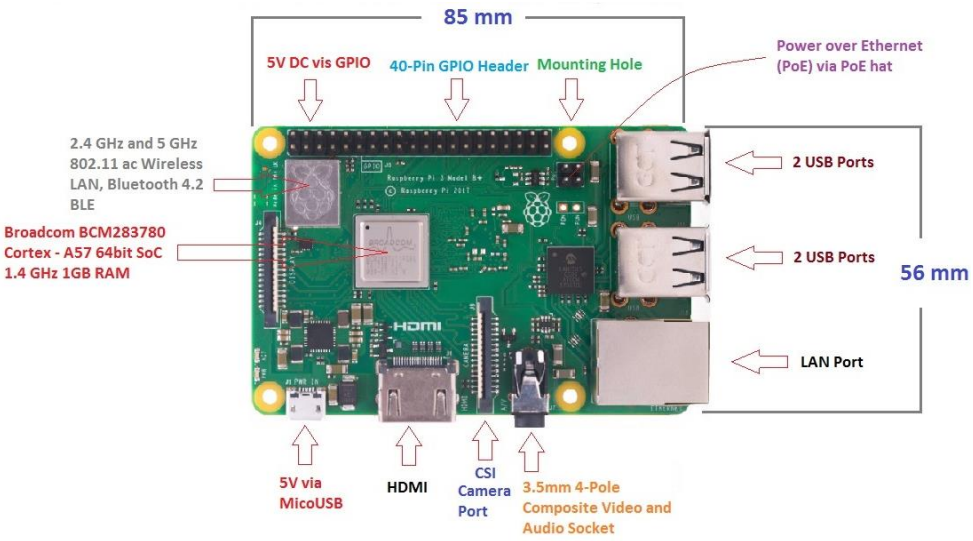


Figure 6 Raspberry pi 3+ structrue

4.1.2. Nodemcu ESP8266

Nodemcu[2] is an open source IoT platform that includes firmware which running on ESP8266 Wi-Fi module. It is developed be ESP8266 Open Source Community, its type single-board microcontroller, it has XTOS operating system and ESP8266

cpu, its storage 4Mbytes and its memory 128Kbytes, uses wireless protocol 802.11 n/b/g , built in PCB antenna on the ESP-12e chip, uses CP2102 USB serial communication interface module.

It supports many network protocols IPv4, TCP, UDP, FTP, and HTTP and has a frequency range 2.412-2.484GHz. It is powered by USB and its power voltage 3.3v, 5v, it is also gas 15 pin header with access to GPIOs, SPI, ADC ,UART and power pins, also it is breadboard friendly, lightweight. It has advantages from the other generations like low cost, reduced size of the board, low energy consumption and integration support for Wi-Fi network, it is programmed using Arduino IDE software. It is used as a client that publishes and subscribes to the server (Raspberry pi).

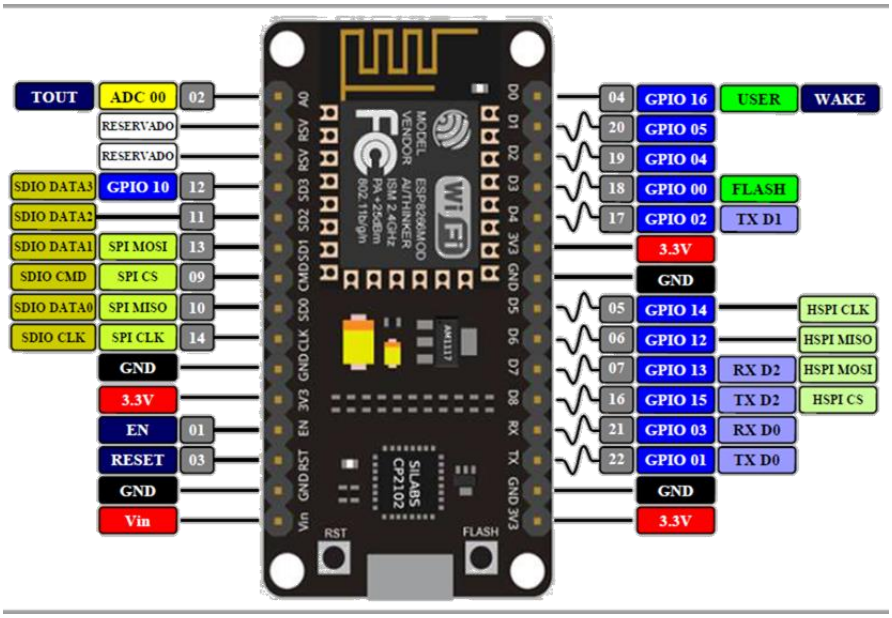


Figure 7 Nodemcu Esp8266

4.1.3. MQ6 module

This module[3] used to detect LPG, Propane or Butane gas, its operating voltage 5v and its analog from 0v-5v and digital output 0v-5v(TTL logic).Its preheat duration 20 seconds, can be used as a digital or analog sensor, and the sensitivity of the digital pin can be varied using a potentiometer.

It has four pins which are VCC, GND, Digital pin and analog pin, it is connected directly with Nodemcu with any extra parts like resistors.

Figure 8 MQ6 Module

4.1.4. DHT11[4]

It is a low cost digital temperature and humidity sensor, it uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It has some features such as operating voltage 3.5v-5.5v, its operating current 0.3mA, its output serial data and the temperature range from 0C-50C, humidity range from 20% to 90% , and its resolution for both temperature and humidity is 16 bit. It has 4 pins VCC, Data(serial data), NC (No connection and not used pin) and GND.

This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness

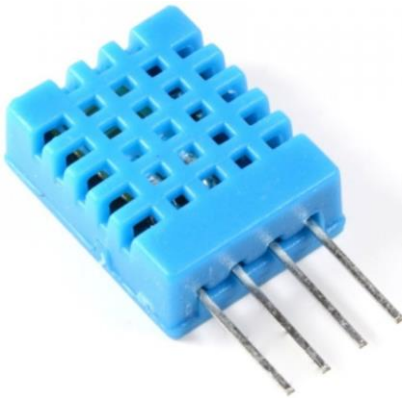


Figure 9 DHT 11

4.1.5. Servo motor[5]

It is tiny and lightweight motor with high output power, can rotate approximately 180 degree(90 in each direction). It can be used by any servo code hardware or library to control these servos.

Its operating voltage 5v typically and its torque 2.5Kg/cm, its gear type plastic and its rotation 0-180 degree, the weight of the motor 9gm and the package includes gear horns and screws. It has three wires brown wire connected to the ground of the system, red wire connected to 5v and the orange wire is PWM signal is given in through this wire to drive the motor.

It is used to control door of the home and its windows.

Figure 10 Servo Motor

4.1.6. Power LEDs

It is unlike the ordinary led, its power 1W and its current 350 mA, 3.2 v , its size 5mm. It has 3 colors red, blue and green. It gives power light equal to 20 leds and it connected through transistor through collector or emitter with resistor. It uses for decorating lighting and fountain lighting.

Figure 11 Power light

4.1.7. Some other small chips

It is also used to build this system some small pieces such as resistors with different values (220k,47k,1k) used to connect power LEDs with Nodemcu and to connect DHT11 with Nodemcu, some transistors(npn), some male-female wires with different lengths and some LEDs and breadboard for testing.

Figure 12 Male-Female wires

Figure 13 Resistors

Figure 14 Transistor

4.2 Materials

We use some applications to see the results and to control the home :

4.2.1 MQTT Client Application

We use this free application that works on MQTT to test our project.

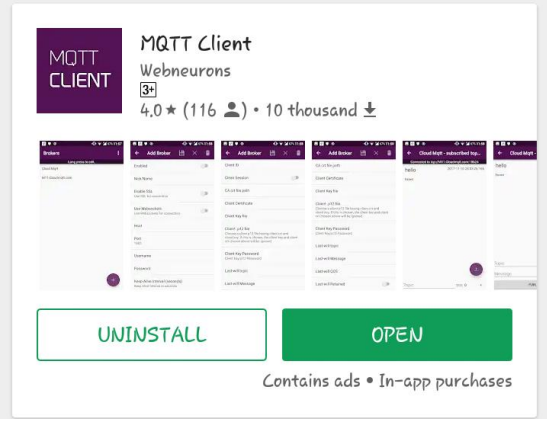


Figure 15 MQTT Client Application

4.2.2 Google Assistant

Google assistant is Google's voice-controlled smart assistant. It's was originally an extension of Google Now- designed to be personal- while expanding on Google's existing "Ok Google" voice control.

The "Ok Google" or "Hey, Google" side covers voice commands, voice searching and voice-activated device control, and so that allow the user to perform things in effective way.

The Google assistant can achieve these things: control the appliances and the smart home, find the information about the status of the home devices such as temperature, humidity and so on.

This will be done with the help of IFTTT and Adafruit platform, the combination of the three applications will provide efficient way to control home devices easily.

IFTTT stands for if this then that, it is a software platform that connects applications, devices and services from different developers in order to perform trigger one or more automations including these appliances and services.

IFTTT consists of some components such as triggers, trigger fields, actions and action fields. First of all, the triggers can be described as events on the user service or signal for the applet to run.

Also there is trigger field let the user to input filter or modifier for a given trigger. In addition, the actions explained as transmitting data or information or creating resources on the service for example "Turn on Light".

Besides action fields can be understood like a form field that will assemble the data sent and analyze it to perform what is required. It is available as a website account and mobile application.

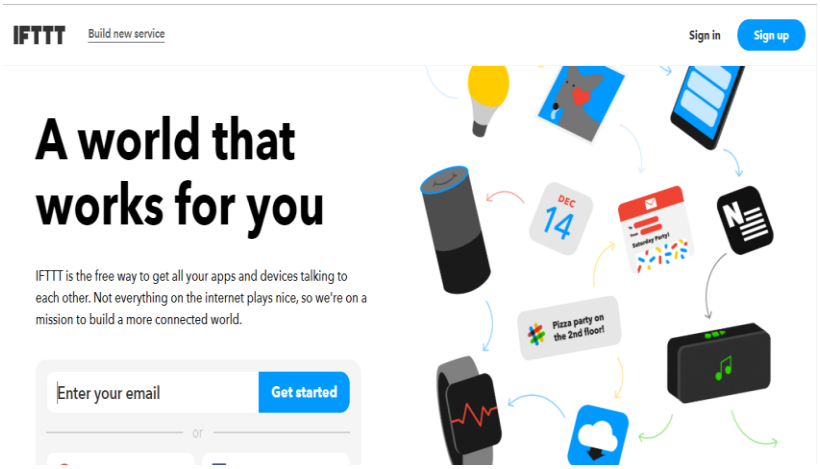


Figure 16 IFTTT

The third part of this combination is **Adafruit IO**, is an IOT platform built around the MQTT protocol. It is userd to provide the mqtt broker service, also it lets the user to setup dashboards that allow the user directly to manipulate or display the current value of each topic, it can be accessed and deal with through a web browser, it makes it the typical center for observing and controlling all of the iot projects. After the creation of Adafruit IO account and design the dashboard of the home automation for example.

Build the dashboard through the left-hand menu, press the Actions drop-down menu, and create a new dashboard. After this step the user clich on the blue button to the right and add new UI components to the page for example toggle buttons, slider and gauge.

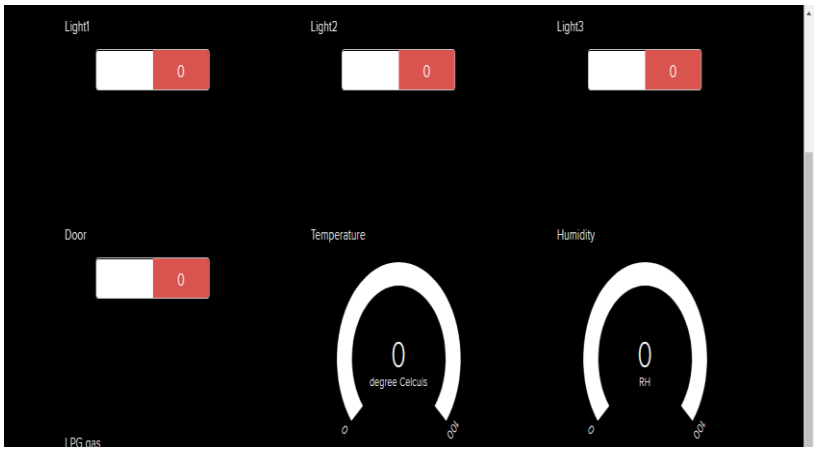


Figure 17 Google Assisatant

And this platform will connect with Nodemcu(esp8266) using Arduino IDE and this include to install the Adafruit MQTT Client library.

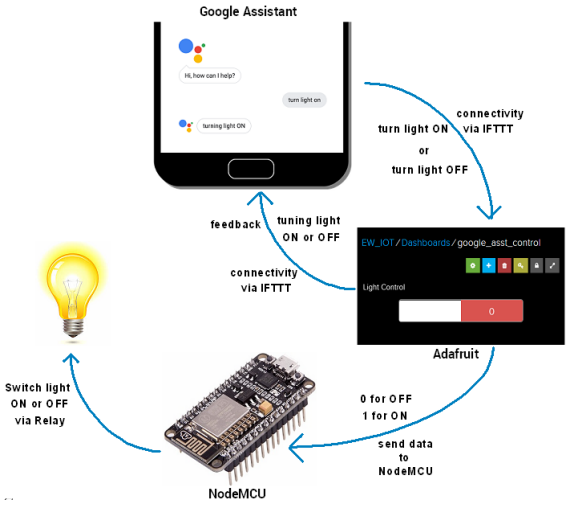


Figure 18 Adafruit with nodemcu and Andro

4.3 Codes

We write codes on both sides server and client :

4.3.1 Server(Broker) Side

MQTT Server is the counterpart of the MQTT client. The broker is responsible for receiving all messages, filtering the messages, determining who is subscribed to each message, and sending the message to these subscribed clients. The broker also holds the sessions of all persisted clients, including subscriptions and missed messages. Another responsibility of the broker is the authentication and authorization of clients.

In this project, MQTT server is set-up using Mosquitto. Any authorized MQTT client can publish or subscribe to the data onto this server with the ID of its host IP and port 1883. The sensor data is aggregated by ESP8266 module and published to the MQTT broker on its topic. Any MQTT client subscribed to that topic can view the sensor readings. LED and Buzzer connected to ESP8266 can be turned ON and OFF by publishing suitable control commands on appropriate topics.

So, Mosquitto has to subscribe to topics that is related to sensors to be able to read their readings. We use MQTT-Paho python to write our code .

First, we install Mosquitto on the raspberry pi using this command :

```
pi@raspberry:~ $ sudo apt install -y mosquitto mosquitto-clients
```

Then Mosquitto was enabled and started :

```
pi@raspberry:~ $ sudo systemctl enable mosquitto.service
pi@raspberry:~ $ mosquitto -d
```

And, paho-MQTT was installed :

```
pi@raspberrypi ~ $ sudo pip install paho-mqtt
```

Then, we write the code :

```
import paho.mqtt.client as mqtt
from flask import Flask, render_template, request
from flask_socketio import SocketIO, emit

app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret!'
socketio = SocketIO(app)

# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))

    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("/esp8266/temperature")
    client.subscribe("/esp8266/humidity")

# The callback for when a PUBLISH message is received from the ESP8266.
def on_message(client, userdata, message):
    #socketio.emit('my variable')
    print("Received message '" + str(message.payload) + "' on topic '"
        + message.topic + "' with QoS " + str(message.qos))
    if message.topic == "/esp8266/temperature":
        print("temperature update")
        socketio.emit('dht_temperature', {'data': message.payload})
    if message.topic == "/esp8266/humidity":
        print("humidity update")
        socketio.emit('dht_humidity', {'data': message.payload})

mqttc=mqtt.Client()
mqttc.on_connect = on_connect
mqttc.on_message = on_message
mqttc.connect("localhost",1883,60)
mqttc.loop_start()
```

```
pins = {
   4 : {'name' : 'GPIO 4', 'board' : 'esp8266', 'topic' : 'esp8266/4', 'state' : 'False'},
   5 : {'name' : 'GPIO 5', 'board' : 'esp8266', 'topic' : 'esp8266/5', 'state' : 'False'}
   }

# Put the pin dictionary into the template data dictionary:
templateData = {
   'pins' : pins
   }

@app.route("/")
def main():
   # Pass the template data into the template main.html and return it to the user
   return render_template('main.html', async_mode=socketio.async_mode, **templateData)

# The function below is executed when someone requests a URL with the pin number and action in it:
@app.route("/<board>/<changePin>/<action>")
def action(board, changePin, action):
   # Convert the pin from the URL into an integer:
   changePin = int(changePin)
   # Get the device name for the pin being changed:
   devicePin = pins[changePin]['name']
   # If the action part of the URL is "1" execute the code indented below:
   if action == "1" and board == 'esp8266':
      mqttc.publish(pins[changePin]['topic'],"1")
      pins[changePin]['state'] = 'True'
   if action == "0" and board == 'esp8266':
      mqttc.publish(pins[changePin]['topic'],"0")
      pins[changePin]['state'] = 'False'
   # Along with the pin dictionary, put the message into the template data dictionary:
   templateData = {
      'pins' : pins
   }
   return render_template('main.html', **templateData)
```

Figure 19 Pyhton code at paho-mqtt

As shown, we make a paho-mqtt-client and this client subscribe to temperature, humidity and gas detection topics. That's will make Mosquitto be able to see the readings of the sensors. Also, Mosquitto will publish the messages from clients that subscribed on LEDs and garage door topics.

4.3.2 Client Side

We have two main clients(two Nodemcu), which subscribe and publish to Mosquitto.

First Nodemcu subscribes on LEDs topics and publishes on temperature and humidity topics. Second Nodemcu subscribes on garage door topic and publishes to Gas detection topic.

mqtt_esp8266 §

```
// Loading the ESP8266WiFi library and the PubSubClient library
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

#define DHTTYPE DHT11   // DHT 11

const char* ssid = "NetworksLab";
const char* password = "Networks123456";
const char* mqtt_server = "172.16.96.205";

WiFiClient espClient;
PubSubClient client(espClient);

const int ledGPIO5 = 5;
const int ledGPIO4 = 4;

const int DHTPin = 14;
DHT dht(DHTPin, DHTTYPE);
long now = millis();
long lastMeasure = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
```

mqtt_esp8266 §

```
String messageTemp;

for (int i = 0; i < length; i++) {
  Serial.print((char)message[i]);
  messageTemp += (char)message[i];
}
Serial.println();
if(topic=="esp8266/4"){
    Serial.print("Changing GPIO 4 to ");
    if(messageTemp == "1"){
      digitalWrite(ledGPIO4, HIGH);
      Serial.print("On");
    }
    else if(messageTemp == "0"){
      digitalWrite(ledGPIO4, LOW);
      Serial.print("Off");
    }
}
if(topic=="esp8266/5"){
    Serial.print("Changing GPIO 5 to ");
    if(messageTemp == "1"){
      digitalWrite(ledGPIO5, HIGH);
      Serial.print("On");
    }
    else if(messageTemp == "0"){
      digitalWrite(ledGPIO5, LOW);
      Serial.print("Off");
    }
}
```

mqtt_esp8266 §

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      client.subscribe("esp8266/4");
      client.subscribe("esp8266/5");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }


    if (client.connect("ESP8266Client2")) {
      Serial.println("connected");
      client.subscribe("esp8266/2");
      //client.subscribe("esp8266/5");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

mqtt_esp8266 §

```
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  if(!client.loop())
    client.connect("ESP8266Client");
    client.connect("ESP8266Client2");

  now = millis();
  if (now - lastMeasure > 10000) {
    lastMeasure = now;
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    float f = dht.readTemperature(true);
    if (isnan(h) || isnan(t) || isnan(f)) {
      Serial.println("Failed to read from DHT sensor!");
      return;
    }
    float hic = dht.computeHeatIndex(t, h, false);
    static char temperatureTemp[7];
    dtostrf(hic, 6, 2, temperatureTemp);
    static char humidityTemp[7];
    dtostrf(h, 6, 2, humidityTemp);
    client.publish("/esp8266/temperature", temperatureTemp);
    client.publish("/esp8266/humidity", humidityTemp);
```

Figure 20 Arduino code for nodemcu

Chapter 5: Results and Analysis

In MQTT Client Application, we enter the broker IP and the client ID to see the results. We test all functionality as shown :

At this page, we can subscribe at any used topic to see the upcoming readings for them.

Here, we subscribed at 3 topics : temperature, humidity and Gas Status. Each topic has the readings that come from the Nodemcu.

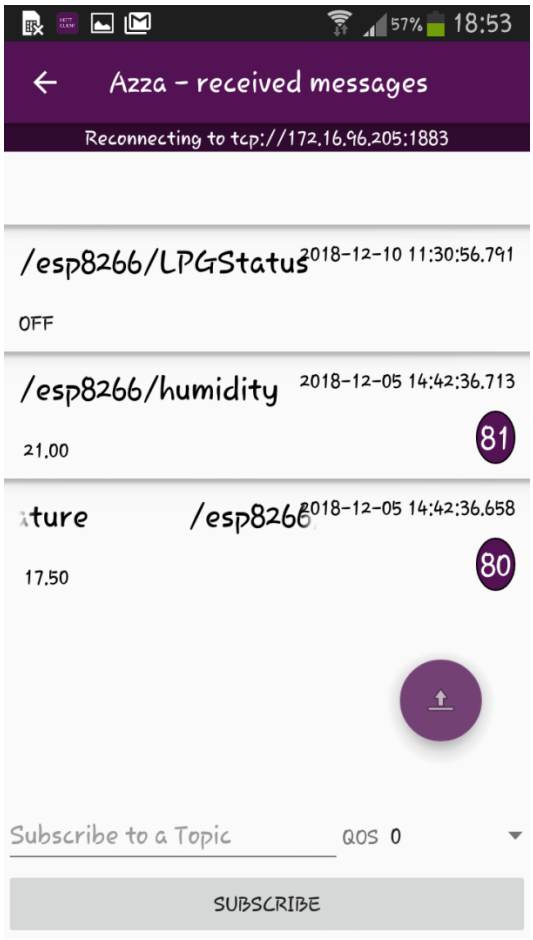


Figure 21 MQTT client - Subscribe page

For example, every 10 seconds new readings are received at temperature and humidity topics. As shown below :

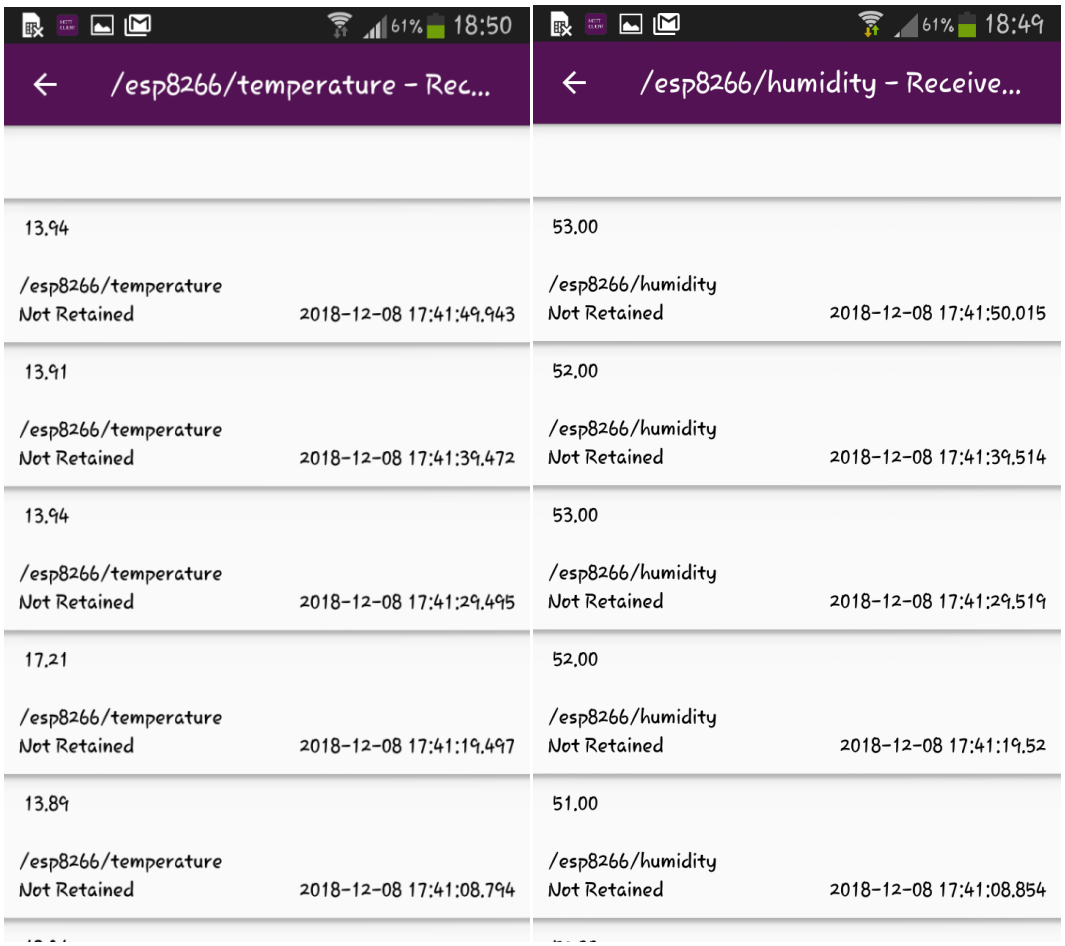


Figure 22 Temperature and humidity Readings

And, the gas detection status. OFF means that there is no gas leakage. ON means there is gas leakage , and a buzzer will turn on at home and a window will be opened.

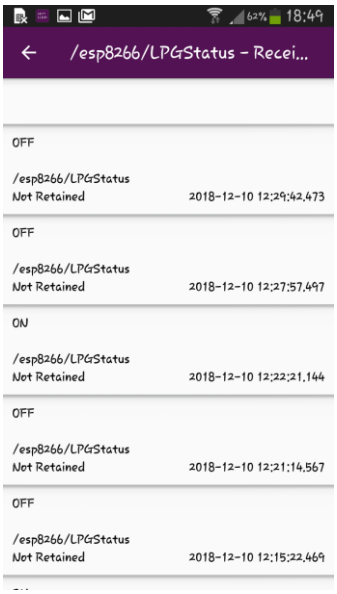


Figure 23 Gas Leakage status page

Figure 24 is the publish page. We write the topic that we want to send data to it and the message.

Figure 24 publisher page

The final result of the home :

Figure 25 Front View of home

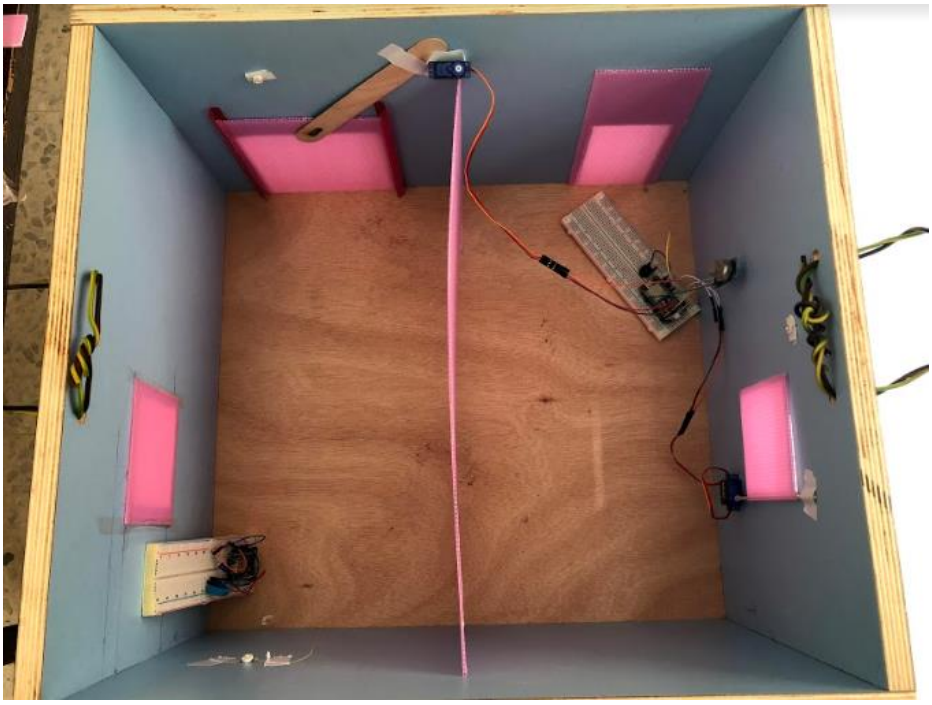


Figure 26 Top View of home

Chapter 6: Discussion

The goal of this project is to implement a home automation system based on MQTT protocol. As IoT is efficient and popular technology, we liked to learn it and try to use it in our real life. Facebook currently uses MQTT for their messenger app, not only because the protocol conserves battery power during mobile phone-to-phone messaging, but also because, in spite of inconsistent internet connections across the globe, the protocol enables messages to be delivered efficiently in milliseconds.

At the beginning, we took time to learn the protocol very well and to deicide the appropriate devices to use. Also, we had difficulty to build the house, so we ask for help from a carpenter. We start from the raspberry pi and MQTT broker. Then, MQTT clients on Nodemcu and connect sensors and devices with Nodemcu. Finally, we connect MQTT client and broker together.

Chapter 7: Conclusion and recommendations

MQTT is thus a light weight protocol that occupies low bandwidth and consumes less power. Considering the ease of wireless internet access through Wi-Fi, MQTT client application is built on ESP8266. A prototype of MQTT based home automation system is implemented on ESP8266. The sensors and actuators connected to ESP8266 are remotely monitored and controlled through a common home gateway. Thus the existing infrastructure can be used to enhance the home appliances and make them smart. This implementation provides an intelligent, comfortable and energy efficient home automation system. It also assists the old and differently able persons to control the appliances in their home in a better and easier way.

References :

[1] Raspberry pi 3 , [online]. Available : https://thepihut.com/products/raspberry-pi-3-model-b-plus.

[2] Nodemcu [online] . Available : https://www.ahirlabs.com/2017/10/21/what-is-nodemcu-esp8266.

[3] MQ6 Module[online] . Available : https://components101.com/sensors/mq-6-gas-sensor-pinout-equivalent-datasheet .

[4] DHT11 [online] . Available : https://www.robot-r-us.com/vmchk/sensor-temp/humid/dht11-temperature-and-humidity-sensor.html.

[5] Servo Motor[online]. Available : https://components101.com/servo-motor-basics-pinout-datasheet.