

**IMPLEMENTATION OF IOT-BASED HOME CONTROL  
AND MONITORING SYSTEM USING RASPBERRY PI  
AND NodeMCUs**

**KAUNG NYUNT SAN**

**M.C.Tech.**

**JANUARY 2019**

**IMPLEMENTATION OF IOT-BASED HOME CONTROL  
AND MONITORING SYSTEM USING RASPBERRY PI  
AND NodeMCUs**

**BY**

**KAUNG NYUNT SAN**

**B.C.Tech.**

**A Dissertation Submitted in Partial Fulfillment of the  
Requirements for the Degree of**

**Master of Computer Technology**

**(M.C.Tech.)**

**University of Computer Studies, Yangon**

**JANUARY 2019**

## ACKNOWLEDGEMENTS

I thank all my teachers and my friends, who taught and helped me during the period of studies at the University of Computer Studies, Yangon.

First of all, I would like to express my gratitude and sincere thanks to **Dr. Mie Mie Thet Thwin**, Rector of the University of Computer Studies, Yangon, for allowing me to develop this thesis.

I would like to thank **Dr. Khin Than Mya**, Professor and Head of Faculty of Computer Systems and Technologies, University of Computer Studies, Yangon, for her guidance and enthusiasm throughout the progress of this work.

I also wish to especially and deeply thank to my supervisor **Daw Hlaing Thida Oo**, Associate Professor of Faculty of Computer Systems and Technologies, University of Computer Studies, Hinthada, for her invaluable suggestion, supervision, guiding and constant encouragement all through the thesis work.

I also thank to my teacher, **Dr. Aung Htein Maw**, Professor at University of Information Technology, Yangon, an external teacher of this thesis committee and I am gratefully indebted to his for his precious comments on this thesis.

I would also like to show my gratitude to my teacher, **Dr. Tweta Oo**, Lecturer and Coordinator of Faculty of Computer Systems and Technologies, University of Computer Studies, Yangon, for her suggestion and help in this work.

Moreover, I would like to thank **Daw Nan Win Yee**, Tutor of Department of Language, University of Computer Studies, Yangon, for editing my thesis from the language point of view.

I really like to express my great thanks to **my parents** for their continuous love and support throughout my studies. Also, thanks to my family for their support during my hard time.

Finally, I also thank **my fellow classmates** who have been with me over the past few years, their support and assistance has been key in helping me.

## **ABSTRACT**

This thesis presents the application of “Implementation of IoT-based Home Control and Monitoring System using Raspberry Pi and NodeMCUs”. The purpose of the presented application is an efficient implementation for IoT (Internet of Things) used for monitoring and controlling the home appliances via World Wide Web. Home automation system uses the portable end devices as a user interface. They can communicate with home automation network through an Internet gateway, by means of low power communication protocols like MQTT, Wi-Fi etc. The system is running with Raspberry Pi (Credit-card size computer) as a server and NodeMCU as a client node. User can easily control many appliances like light switches, electronic devices and door locks, and also can monitor home environment through a friendly web-based user interface through browser (Firefox, Chrome, Internet Explorer, etc.) from end devices. To protect from the aspect of fireplace accidents this system contains flame-detected fire alarm function which detects the flame within the event of fireplace and inform the status of event to the user as messages, e-mails and notifications. The server is directly connected with relay-hardware circuits to control the home appliances and webcam for monitoring the home environment. This system makes easier by implementing automation and security along with the Internet of Things to create a system which will enable someone to remotely monitor and control some areas of a house remotely and securely from anywhere with minimum cost.

# CONTENTS

|   | <b>Page</b> |
|---|-------------|
| <b>ACKNOWLEDGEMENTS</b>   | i           |
| <b>ABSTRACT</b>   | ii          |
| <b>CONTENTS</b>   | iii         |
| <b>LIST OF FIGURES</b>  | vi          |
| <b>LIST OF TABLES</b>   | ix          |
| <b>CHAPTER 1 INTRODUCTION</b>   | 1           |
| 1.1 Objectives of the Thesis  | 2           |
| 1.2 Outline of the Thesis   | 2           |
| 1.3 Overview of the Thesis  | 3           |
| <b>CHAPTER 2 BACKGROUND THEORY</b>  | 6           |
| 2.1 Automatic Control System (Automation)                                 | 7           |
| 2.1.1 Open-loop and closed-loop (feedback) control                        | 7           |
| 2.1.2 Computer Control  | 8           |
| 2.2 Wireless Communication Technologies                                   | 9           |
| 2.3 Lightweight Communication Protocols                                   | 9           |
| 2.3.1 Message Queuing Telemetry Transport                                 | 11          |
| 2.3.2 Constrained Application Protocol (CoAP)                             | 13          |
| <b>CHAPTER 3 ANALYSIS AND REVIEW OF IoT-BASED HOME AUTOMATION SYSTEMS</b> | 15          |
| 3.1 Internet of Things  | 15          |
| 3.2 Raspberry Pi  | 16          |

|                  |  |           |
|------------------|--|-----------|
| 3.2.1            | Software of Raspberry Pi   | 18        |
| 3.3              | NodeMCU  | 18        |
| 3.3.1            | Specification of NodeMCU   | 19        |
| 3.3.2            | Code examples  | 21        |
| 3.4              | Wireless Router  | 24        |
| 3.5              | Sensors and Components of the System                                     | 25        |
| 3.5.1            | Magnetic Door Sensor   | 25        |
| 3.5.2            | DTH 11 Sensor  | 26        |
| 3.5.3            | Flame Sensor Module  | 26        |
| 3.5.4            | RFID Module  | 27        |
| 3.5.5            | Relay  | 28        |
| 3.5.6            | Buzzer   | 29        |
| 3.5.7            | Keypad   | 29        |
| 3.5.8            | Solenoid Electronic lock   | 30        |
| 3.5.9            | Webcam   | 30        |
| 3.6              | Design Implementation of IoT-based Home<br>Control and Monitoring system | 31        |
| <b>CHAPTER 4</b> | <b>HARDWARE AND SOFTWARE IMPLEMENTATION</b>                              | <b>35</b> |
| 4.1              | Sensors and Camera Interfacing   | 35        |
| 4.2              | Raspberry Pi and NodeMCU   | 36        |
| 4.3              | Wi-Fi Router Configuration   | 38        |

|   |           |
|---|-----------|
| 4.4 MQTT Protocol                                 | 39        |
| 4.4.1 MQTT Method and Testing                     | 40        |
| 4.5 IFTTT, Remot3.it, Weather Underground         | 42        |
| 4.5.1 IFTTT                                       | 42        |
| 4.5.2 Remot3.it                                   | 44        |
| 4.5.3 Weather Underground                         | 46        |
| 4.6 Home Assistant                                | 47        |
| 4.7 Software Implementation of the System         | 57        |
| 4.8 Operation of the System                       | 59        |
| 4.9 Comparison from other Home Automation System  | 62        |
| 4.10 Evaluation of the System                     | 64        |
| <b>CHAPTER 5 CONCLUSION AND FURTHER EXTENSION</b> | <b>66</b> |
| 5.1 Conclusion                                    | 66        |
| 5.2 Advantages of the Thesis                      | 67        |
| 5.3 Limitations of the Thesis                     | 67        |
| 5.4 Further Extension                             | 68        |
| <b>REFERENCES</b>                                 | <b>69</b> |
| <b>PUBLICATION</b>                                | <b>71</b> |

## LIST OF FIGURES

| <b>Figure</b>  | <b>Page</b> |
|--|-------------|
| Figure 1.1 Block Diagram of Proposed System                              | 4           |
| Figure 1.2 Architecture of Proposed System                               | 4           |
| Figure 2.1 Block Diagram of the Open Loop and Closed Loop Control System | 7           |
| Figure 2.2 State Diagram of Door System                                  | 8           |
| Figure 2.3 M2M Architecture  | 10          |
| Figure 2.4 Levels of QoS   | 13          |
| Figure 2.5 Comparison of MQTT and CoAP                                   | 14          |
| Figure 3.1 Layers of Internet of Things (IoT)                            | 15          |
| Figure 3.2 Series of Raspberry pi  | 16          |
| Figure 3.3 Raspberry Pi 3 Model B with GPIO Pinout Diagram               | 17          |
| Figure 3.4 NodeMCU   | 19          |
| Figure 3.5 Pin Definition of NodeMCU                                     | 20          |
| Figure 3.6 Sim Wireless Router   | 25          |
| Figure 3.7 Magnetic Door Sensor  | 25          |
| Figure 3.8 DTH11 Sensor  | 26          |
| Figure 3.9 Flame Sensor Module   | 27          |
| Figure 3.10 RFID Reader Module and Tag                                   | 27          |
| Figure 3.11 Relay Module   | 28          |
| Figure 3.12 Buzzers  | 29          |



|  |    |
|--|----|
| Figure 3.13 Keypads  | 30 |
| Figure 3.14 Solenoid Electronic lock   | 30 |
| Figure 3.15 Webcams  | 31 |
| Figure 3.16 Design Layout of Front Door Part                                     | 32 |
| Figure 3.17 Design Layout of Back Door Part                                      | 33 |
| Figure 3.18 Design Layout of Main Part   | 34 |
| Figure 4.1 Preparation result of Raspberry Pi                                    | 37 |
| Figure 4.2 Preparation result of NodeMCU   | 38 |
| Figure 4.3 Wi-Fi configuration setting   | 39 |
| Figure 4.4 MQTT Process  | 40 |
| Figure 4.5 MQTT Testing  | 41 |
| Figure 4.6 IFTTT Applets   | 43 |
| Figure 4.7 IFTTT API key   | 43 |
| Figure 4.8 Configured Services and Connections of Remot3.it                      | 45 |
| Figure 4.9 WUI and API key of Weather Underground Service                        | 47 |
| Figure 4.10 Home Assistant Platform and Some Support Components                  | 48 |
| Figure 4.11 Text Output of home-assistant@pi.service File                        | 51 |
| Figure 4.12 Output of configuration.yml File                                     | 52 |
| Figure 4.13 Result of checking conFigure files with “hass --script check_config” | 53 |
| Figure 4.14 Result of Testing Home Assistant Webpage                             | 54 |
| Figure 4.15 List of Home Assistant Configuration Files for the System            | 55 |

|   |    |
|---|----|
| Figure 4.16 Complete Web-UI of the Whole System           | 55 |
| Figure 4.17 Hardware Implementation of the System         | 56 |
| Figure 4.18 Prototype of the System                       | 57 |
| Figure 4.19 Flow Chart of the System                      | 58 |
| Figure 4.20 Login Page and Complete WUI of Home Assistant | 60 |
| Figure 4.21 Notifications receive from user side          | 61 |

## LIST OF TABLES

| <b>Table</b>  | <b>Page</b> |
|---|-------------|
| Table 2.1 Wi-Fi Protocols and their Data rates                            | 9           |
| Table 2.2 Quick Comparison of MQTT and CoAP                               | 11          |
| Table 3.1 Pin Mapping Table of NodeMCU                                    | 20          |
| Table 4.1 Comparison with other Home Automation System                    | 62          |
| Table 4.2 Comparison of the Features of Commercial Home Automation System | 65          |

## CHAPTER 1

# INTRODUCTION

Today, technology play very important role in our daily lives. We cannot live without technologies such as televisions, smart phones, computers, Internet and others. Because these technologies influence as an essential part in our day-to-day lives. Communication is thus enhanced, and people can communicate more easily with each other. With the help of technologies, we can easily communicate with our friend and store personal data such as photo, documents, music and movies. To simplify daily life of people, we connect many devices to Internet. The main advantage of the Internet is its ability to connect billions of end devices from different places to create a link between them. So, we can connect from any places to get access and information of devices we need. During this progress we can saving time, energy and money.

Home automation (also known as domotics) refers to control appliances, activity, and features of home. By using end devices from anywhere in the world Home automation can give you access to control devices at your home [1]. In this system all the main parts are connected wirelessly and Message Queuing Telemetry Transport (MQTT) is used to communicate between them. MQTT is mainly designed for automation, it has a low footprint to send and receive data. So, very less amount of data is used between MQTT server and clients [2]. It also uses three Cloud services IFTTT, Remot3.it and Weather Underground. IFTTT service is used for send notifications, G-mails and messages to user. Remot3.it service supports user remotely accesses to the system from anywhere. Getting weather information around home environment from Weather Underground service.

## 1.1 Objectives of the Thesis

The main objective of the thesis is to control and monitor our home easily using web-user interface(browser) from End-*devices* which are connected to the Home Automation system running on Raspberry pi particularly for monitoring and controlling electronic door locks, door sensors, surveillance cameras, and flame detectors, which helps ensuring and maximizing safety and security of homes. This system addresses an IoT-based approach on the field of Home Automation. Common use-cases include checking the status of home, controlling appliances at home and controlling the access of home through keypad and RFID cards as an example and doors through electronic locks. However, the main focus of this thesis is to control and monitor of homes applying of Computer Control theory and Wireless communication. Other objectives of the thesis are as follows;

- To introduce about the IoT (Internet of Things)
- To study how the Open-source OS (Raspbian) runs on Raspberry Pi
- To learn about how Raspberry Pi, NodeMCU & Sensors work together
- To study Home Automation Platform (Home Assistant) and how MQTT protocol (mosquitto) works

## 1.2 Outline of the Thesis

In this thesis, the detailed descriptions of related information and function involved in developing the **IoT-based Home Control and Monitoring System** are explained in each chapter. The chapters and sub-titles are concentrated on giving detailed description of the equipment.

There are 5 chapters in this thesis. The introduction of the system, objectives of the thesis and outline of the thesis as well as summary of work are introduced in Chapter 1.

The background theory, Computer Control, and Wireless Communication are presented in Chapter 2. In Chapter 3, the methodology hardware, detailed design of controllers, Raspberry Pi and NodeMCUs are described. The functions and requirement of controllers, interface circuits for other peripheral components, DHT11(Temperature and

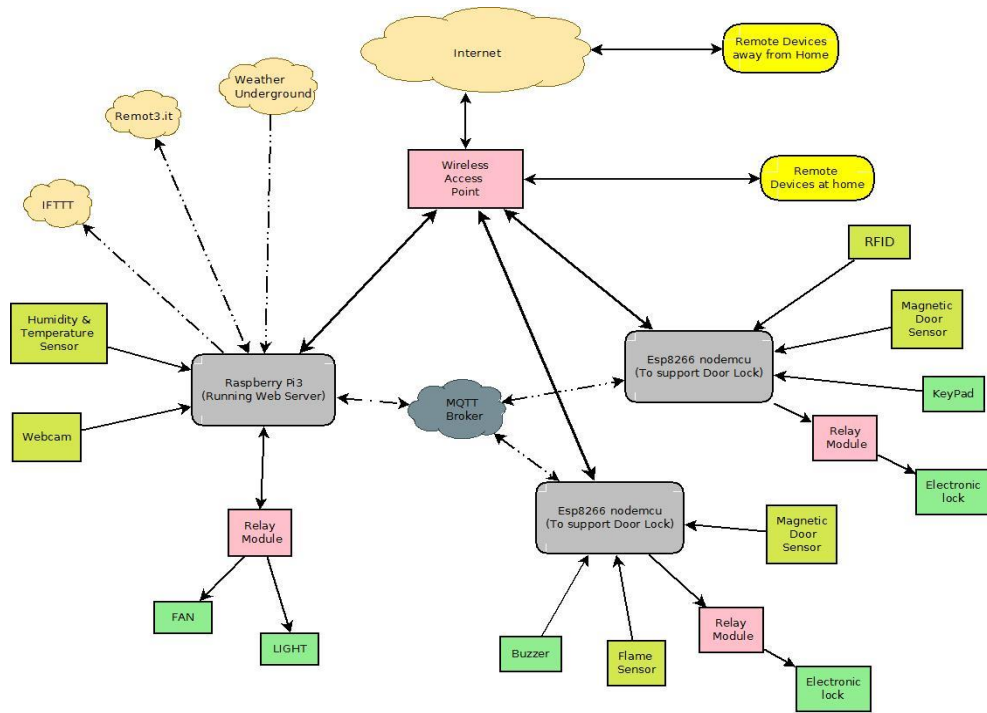
Humidity Sensor), Magnetic Door/Window Sensor, Relay Module, Keypad, Electronic lock, Mosquitto MQTT, IFTTT, Remot3.it, Weather Underground, Home Assistant Open-Source Platform and other components of the system are also stated in this chapter.

The software implementation and system results discussion are presented in Chapter 4. In Chapter 5, the conclusion, advantages and disadvantages, limitation of the thesis and future work that are likely to be carried out are discussed.

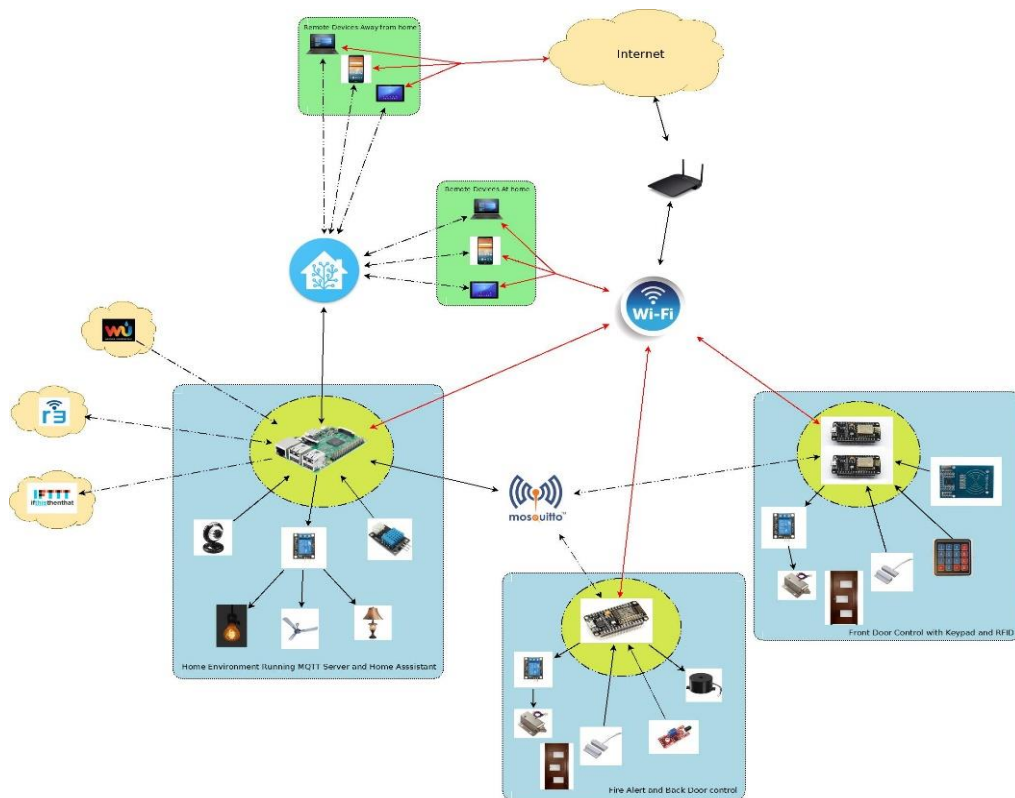
### **1.3 Overview of the Thesis**

In this thesis, the IoT-based Home Control and Monitoring system is developed by implementing Raspberry Pi, NodeMCUs. The main objectives are to design and to make an effective for low-cost and open source home control and monitoring system that might lead most of the homes and sustain the home automation system. The great efficiency of this system is the use of reliable wireless technology to make connections between various modules and server of this automation system. This proposed system can reduce the deployment cost; complexity of the wires and reconfiguration of the whole system. The wireless connection between sensors nodes, server, clients and various communication protocols between server and users are mainly used in this system.

The block diagram and architecture of proposed system is shown in Figure 1.1 and Figure 1.2. The system can be delivered into three main parts Front Door, Back Door and Main Server. All main parts are connected to Wireless Access Point and they communicate each other wirelessly through MQTT Protocol to exchange information.



**Figure 1.1 Block Diagram of Proposed System**



**Figure 1.2 Architecture of Proposed System**

Front Door part is running with two NodeMCUs. One NodeMCU is connect to the MQTT Broker as MQTT Client to exchange data. The other NodeMCU is to support user multi-access to control the door lock. This part contains Magnetic-Door sensor sensing the status of door OPEN/CLOSED, Electronic door lock to LOCK/UNLOCK the door and Relay Module to control Electronic door lock, which can be controlled with RFID or Keypad to open the door. Back Door part is running with one NodeMCU as MQTT Client connect to MQTT broker and exchange data like Front Door part. Also adding Fire Alert in this part to sense the flame of fire with flame sensor and triggered on buzzer if there is any fire accident detected. Main Server part is running with Raspberry Pi that support User Interface to control home appliances, monitor the situation of home through Home Assistant Platform. Home appliances, Humidity & Temperature Sensor, Webcam and Cloud Services are connected to Raspberry Pi. Alert critical status of home to user using IFTTT Cloud service and also run MQTT broker to support communication channel within main parts [3].



## CHAPTER 2

### **BACKGROUND THEORY**

During the past few years, internet was known as a big mass that we can acquire data from. Embedding mobile transceivers to everyday items and gadgets enabled new forms of bi-directional communication between people and other people, as well as people and things.

While that paradigm is growing and it has high positive impact on many aspects of our lives, challenging issues arise, that should be considered and addressed. The central issues are guaranteeing security and privacy of users and their data. Another issue is fully achieving smartness of interconnected devices by enabling their interaction. Exchanging data and autonomous behavior is the key to achieving the latter.

IoT has different definitions from different perspectives, however, they all revolve around "things" generally, collecting, exchanging and communicating data with each other's and with people through the "internet". IoT helps in decision making and automating almost everything around us. The smarter life IoT vision promises in the near future through various applications, makes smart Home Automation actually possible, starting from basically monitoring different parts of home, to actually controlling them. Integration of IoT and Home Automation, makes it possible to monitor and control homes from different parts of the world. Some examples of applications to this are: controlling and setting the desired temperature of the house before arriving home, turning on/off the lights of a room and setting its intensity, running washing machine while the person is at work, leakage or flame detection and notification, monitoring home through surveillance camera or car inside the house while the person is away, or remote central locking, and many other applications.

As the field of Home Automation through IoT is chosen to work a wide application in a very wide and challenging field due to the reasons mentioned in the previous

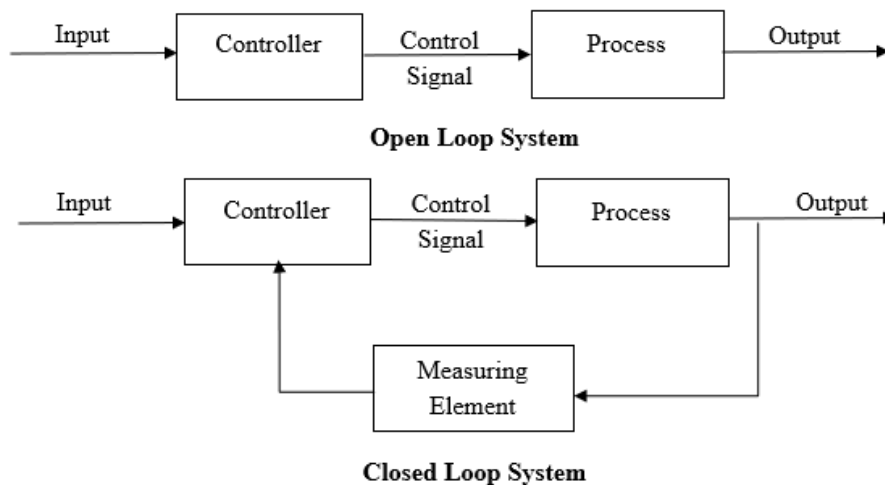
paragraphs. I have chosen to work on the field of Home Automation through IoT as part of this thesis, specifically in Computer Control and Wireless Communication technology.

## 2.1 Automatic Control System (Automation)

**Automation** can be defined as a series of process or procedure is performed without any human help. There are various control systems used automation to control many processes without any human assistance.

### 2.1.1 Open-loop and closed-loop (feedback) control

There are two types of control loop; closed loop (feedback) control and open loop control. The control action of open loop control is that the controller is independent of the "process output". And the control action of closed loop control, the controller is dependent on the process output. The decency of the closed loop system can be a Feedback Control loop that tends to maintain a prescribed relationship of variable from one process to another by comparing functions of these variables and using as meaning of the difference of control to get the require output. Figure 2.1 represents Block Diagram of open-loop and closed-loop System.



**Figure 2.1 Block Diagram of the Open Loop and Closed Loop Control System**

### 2.1.2 Computer Control

Computers can perform both feedback control and sequential control, and typically a single computer will do both sequential and feedback control in an industry. Sequential control may be either predefined sequence or logical one that will perform many actions depending on various states of system. Figure 2.2 represent the state diagram of the door system. Programmable logic controllers (PLCs) are a type of special purpose microprocessor that replaced many hardware components such as timers and drum sequencers used in relay logic type systems.

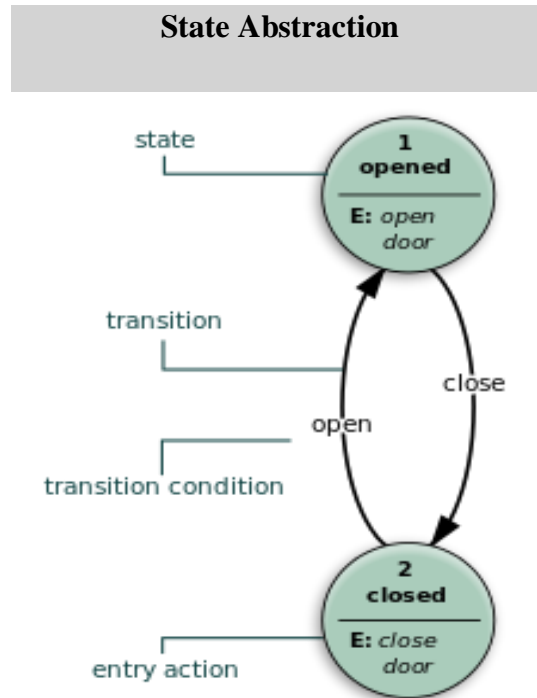


Figure 2.2 State Diagram of Door System

## 2.2 Wireless Communication Technologies

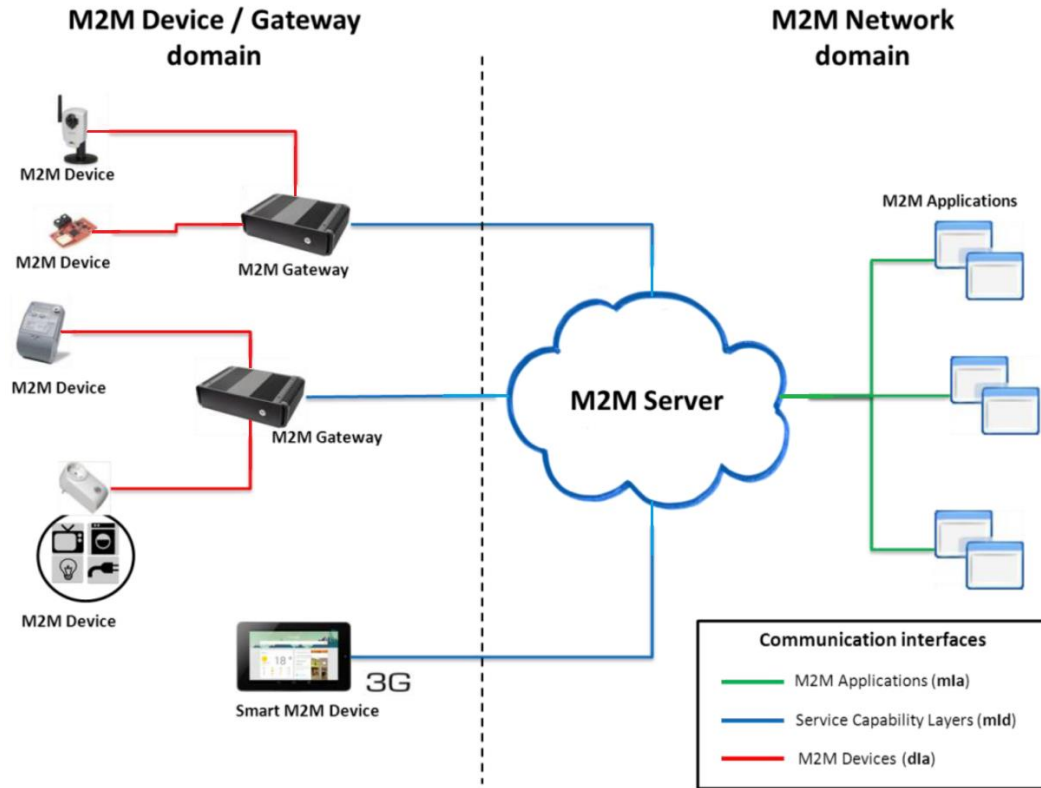
Wireless technologies help us to transfer of information between different many places that are not wirely connected to each other. Also, it is depending on the distances between places that we want to transfer data. Some types of Wireless Communication Technologies are: Bluetooth, Wi-Fi, M2M communications. Wireless fidelity or Wi-Fi is based on the standards of IEEE 802.11a/b/g/n that make for wireless local area networks. The difference of Wi-Fi protocol and their data rates are shown in Table 2.1.

**Table 2.1 Wi-Fi Protocols and their Data rates**

| <b>Protocol</b>      | <b>Frequency</b> | <b>Signal</b> | <b>Maximum data rate</b> |
|----------------------|------------------|---------------|--------------------------|
| <b>Legacy 802.11</b> | 2.4 GHz          | FHSS or DSSS  | 2 Mbps                   |
| <b>802.11a</b>       | 5 GHz            | OFDM          | 54 Mbps                  |
| <b>802.11b</b>       | 2.4 GHz          | HR-DSSS       | 11 Mbps                  |
| <b>802.11g</b>       | 2.4 GHz          | OFDM          | 54 Mbps                  |
| <b>802.11n</b>       | 2.4 GHz or 5 GHz | OFDM          | 600 Mbps (theoretical)   |
| <b>802.11ac</b>      | 5 GHz            | 256-QAM       | 1.3 Gbps                 |

## 2.3 Lightweight Communication Protocols

There are many customized protocols that exist for the IoT to be implemented. But only a handful of them has been standardized for all the devices which work on IoT. Machine to Machine (M2M) is presently a major subset of the Internet of Things (IoT) concept. Wireless Sensor Networks are being implemented in large scale to monitor the status of different places and things like temperature and humidity of rooms, open/closed state of door and home appliances etc. There is no need of any UI(user-interface) for sensor node, so monitoring and controlling if the things and places only from the master side. The architecture of M2M protocols and applications can be seen in Figure 2.3.



**Figure 2.3 M2M Architecture**

The opportunity of M2M is dynamic and growing very fast. It is currently providing connectivity for variety of devices like sensors, home automation system, smart devices, health monitoring systems etc. Because of the things and objects that we are using every day are getting updated and can be addressed, controlled and monitored via networks. The two most evident protocols are:

1. MQTT (Message Queuing Telemetry Transport)
2. CoAP (Constrained Application Protocol)

The above two protocols are better suited to control monitor environments than HTTP because they are open standards. These protocols have very large range of implementations for different requirements using in the IoT field. And both protocols provide asynchronous communication mechanisms and running over IP. There are other major protocols that are used in a smaller environment or as a base protocol like Wi-Fi, Bluetooth, Infra-Red, Zigbee etc. But among these MQTT and CoAP are widely used because of their lightweight design and advantages.

**Table 2.2 Quick Comparison of MQTT and CoAP**

|                         | MQTT                        | CoAP   |
|-------------------------|-----------------------------|--|
| Application             | Completely Single Layered   | Single layered with two sub layers (Message and Request Response Layer)              |
| Transport Layer         | Runs on TCP                 | Runs on UDP  |
| Reliability Mechanism   | 3 Quality of Service levels | Confirmable Messages, Non-Confirmable Messages, Acknowledgements and retransmissions |
| Supported Architectures | Publish-Subscribe           | Request-Response, Resource observe/Publish-Subscribe                                 |

Comparison of MQTT and CoAP protocol can be seen in Table2.2. Transfer Control Protocol is used in MQTT and User Datagram Protocol is used in CoAP. MQTT can be defined as multiple node communication protocol for communication between different nodes through a single node. In MQTT the broker (server) the route of the incoming message. CoAP is one to one protocol to send the state information between server and clients. There are other protocols like Bluetooth and Zigbee that can be used to smaller networks less than 50 nodes.

### **2.3.1 Message Queuing Telemetry Transport**

MQTT is Publish/Subscribe messaging protocol designed for lightweight M2M communications between multiple nodes. It has a very small overhead to transport and the protocol exchanges are minimized to reduce the traffic in the network.

The benefits of publish-subscribe models for MQTT are mentioned below:

1. Information Independent Nodes
2. Time Decoupling
3. Synchronization decoupling
4. Security Analysis
5. MQTT Quality of Service levels

In MQTT Quality of Service levels, each level would indicate the responsibility of a message being delivered. All the three levels are described below:

i. Level 0:

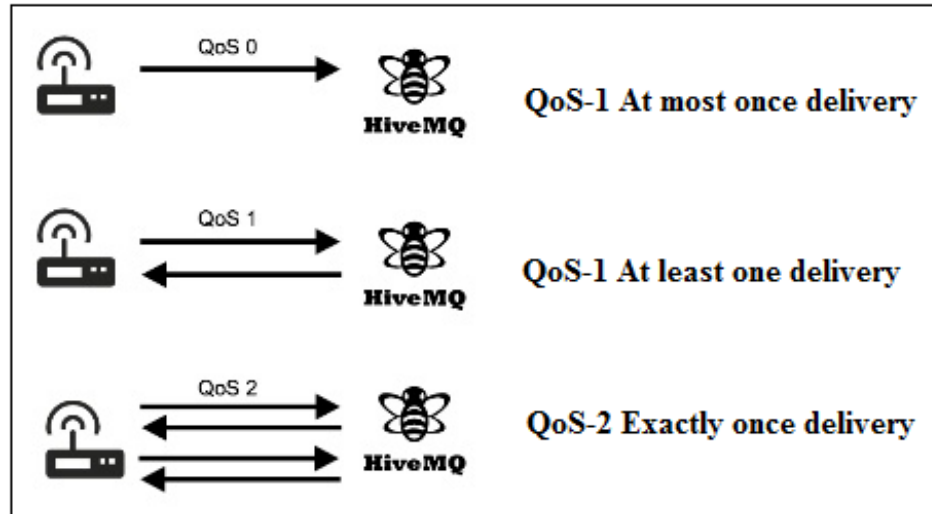
This would send the message and also is called as send it and never acknowledge. It is just a one-time send without confirmation about the message reaching the destination. It is more suited to situations where the importance is low.

ii. Level 1:

This would make sure the data reaches one-time minimum. It guarantees that the data reaches the destination at least once. It gives an acknowledgment about the message arrival at the intended destination. When the data being sent is received and confirmed by the other node, it will send a response/acknowledgment to the sending node. The sender waits for acknowledgment, and it saves the data to be sent and keeps resending it at regular intervals.

iii. Level 2:

This rule will make sure that the data is received and properly fetched by the intended node. This is reliable level among the mentioned levels. When the sender has QoS level 2 messages it sends a message announcement. The proposed receiver gets prompt and then would decrypt it. It shows that it is ready to receive data. The publishing node sends the data, and when the receiver decrypts and gets the data, the whole process is done with an ack/talk back. It is mainly useful for turning things on or off at home. The Quality of Service levels of MQTT can be seen in Figure 2.4.



**Figure 2.4 Levels of QoS**

There are some disadvantages of using MQTT which are: Require Central Broker, Using TCP and Need Wake-up Time. Because of central broker might be a disadvantage for distributed system when the single point of failure is happened. TCP was designed to handle complex handshakes and higher data loads, In IoT this increases waking a node will talk times and brings down battery life, if portable.

### **2.3.2 Constrained Application Protocol (CoAP)**

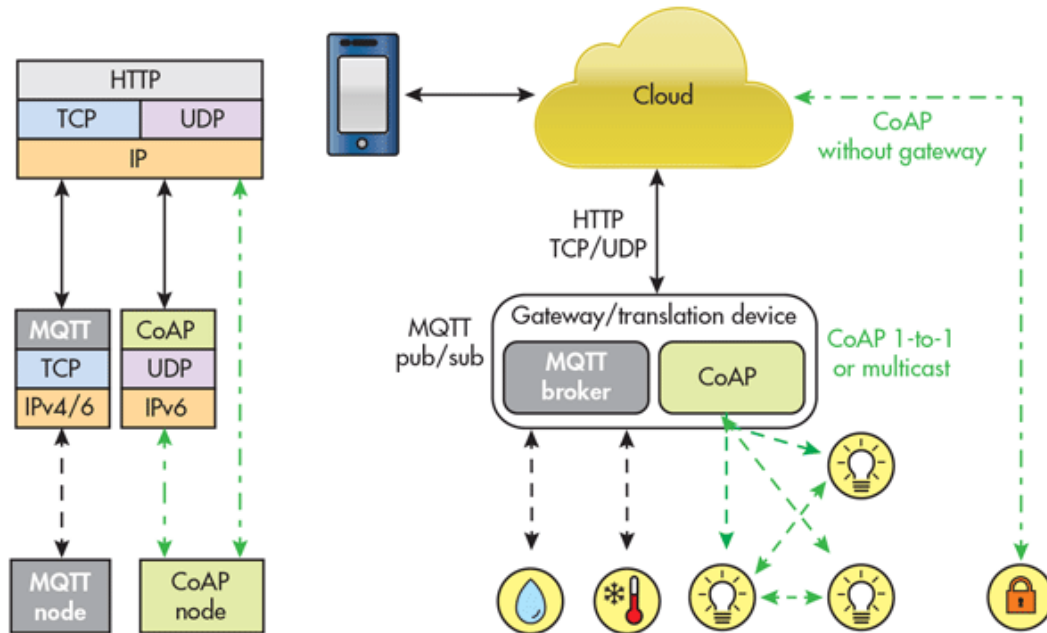
This is another popular IoT protocol which gives one on one response/request with IETF standardization. This protocol is a transfer protocol used for the documents. It works mostly for the constrained devices. CoAP is work on connectionless protocol (User Datagram Protocol). Some advantages of CoAP are as follows:

- i. User Datagram Protocol (UDP)
- ii. Send multiple nodes
- iii. Resource/Service Discovery
- iv. Async-Data Transfer

Considering an example, the observing mode for node-1 can manage node-2 for a given data transfer cycle. Whenever a second node sends any relevant data, primary node will receive it after its sleep cycle. Any one of the node saves data for its observers. This might be similar to working mechanism when compared to MQTT. But it doesn't have any particular need of a broker and need not hold any messages for other observers.



There have some issues with the CoAP. In current market, CoAP is less mature than MQTT. Also the reliability of CoAP is comparatively less with what MQTT provides in terms of levels of QoS. The quick comparison of MQTT and CoAP described in Figure 2.6.



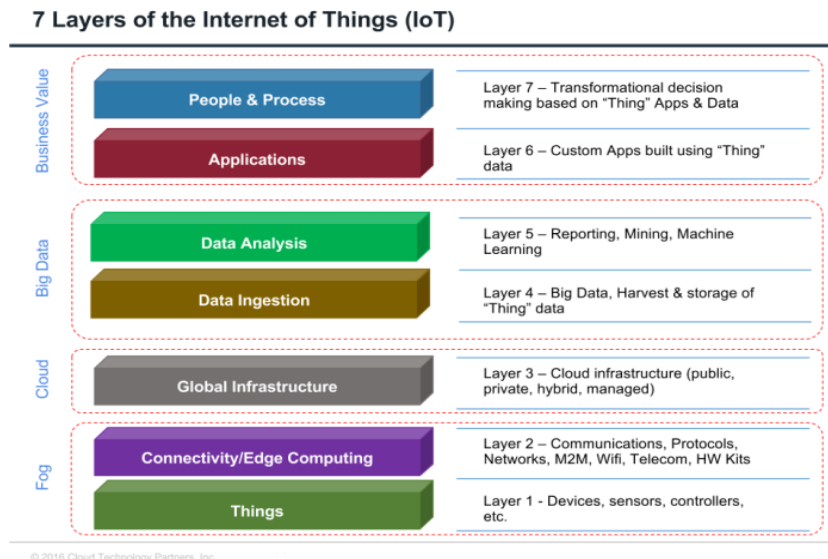
**Figure 2.5 Comparison of MQTT and CoAP**

## CHAPTER 3

# ANALYSIS AND REVIEW OF IoT-BASED HOME AUTOMATION SYSTEMS

### 3.1 Internet of Things

The Internet of things (IoT) easily can be defined as various devices like smart watch, Tablets, Computer, smart light, and CCTV that are connected to the internet. M2M (Machines-to-Machines) communication is the previous level of IoT that connect a device to the cloud, managing it and collecting data. With the help of internet very useful and powerful communication links can be created between things and people and also between things. The emergence of IoT technology, Home automation topic attract people attention. For the purpose of helping human being many devices are connected to internet wirely or wirelessly. And these devices also can remotely control and monitor from any places to improve the intelligence of home environment. By using IoT technology we can easily connect with other things to innovate new idea about smart home and also can improve the living standard of life [4]. Figure 3.1 represent the layers of IoT.



**Figure 3.1 Layers of Internet of Things (IoT)**

## 3.2 Raspberry Pi

A Raspberry Pi is a credit card-sized computer originally designed for education purpose. Compare with modern desktop or laptop, Raspberry Pi is slower but is still a complete portable Linux computer and can provide all the expected features, at a low-power consumption level. Several generations of Raspberry Pis have been released. All models feature a Broadcom system on a chip (SoC) with an integrated ARM compatible central processing unit (CPU) and on-chip graphics processing unit (GPU). Series of Raspberry Pi can be seen in Figure 3.2. Raspberry Pi 3 Model B is used in this system.

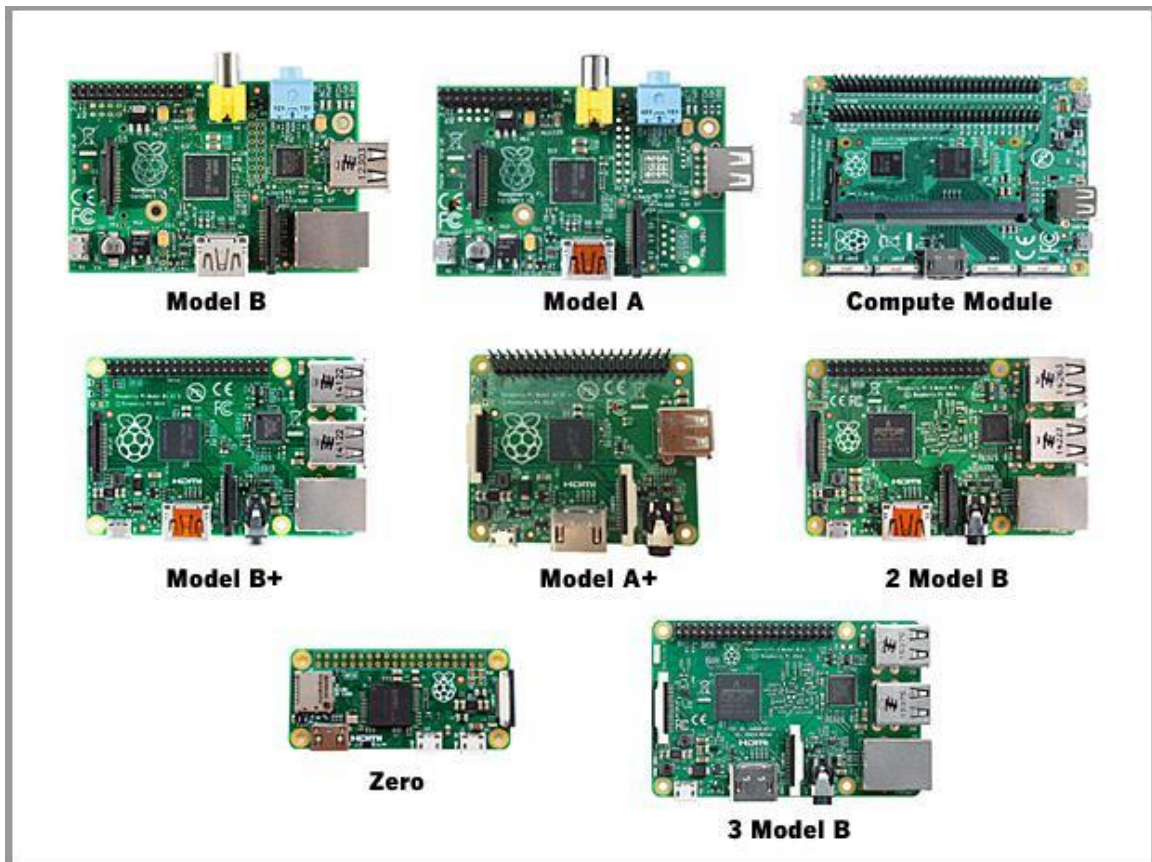
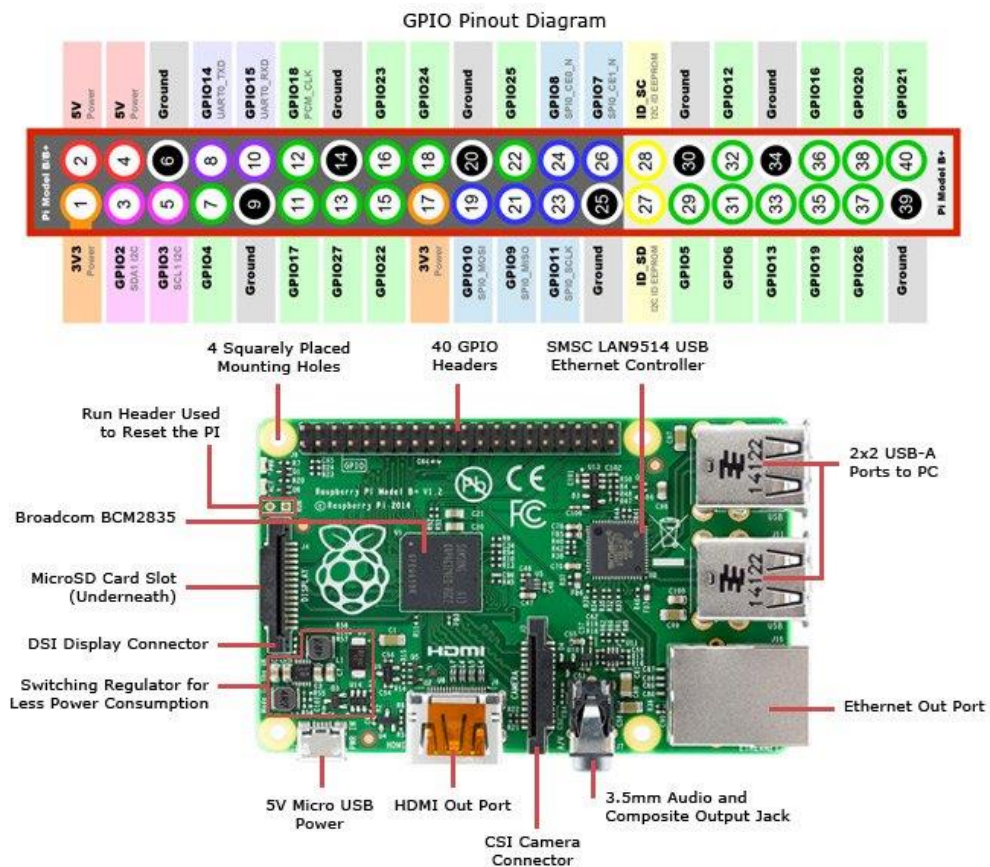


Figure 3.2 Series of Raspberry pi

Specifications of Raspberry Pi 3 Model B are as follow:

- SoC: Broadcom BCM2837 (roughly 50% faster than the Pi 2)
- CPU: 1.2 GHZ quad-core ARM Cortex A53 (ARMv8 Instruction Set)
- GPU: Broadcom Video Core IV @ 400 MHz
- Memory: 1 GB LPDDR2-900 SDRAM
- Network: 10/100 MBPS Ethernet, 802.11n Wireless LAN, Bluetooth 4.0
- Storage: microSD
- GPIO: 40-pin header, populated (Shown in Figure 3.3)
- Ports: HDMI, 3.5mm analogue audio jack, 4 USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)



**Figure 3.3 Raspberry Pi 3 Model B with GPIO Pinout Diagram**

### 3.2.1 Software of Raspberry Pi

The Raspberry Pi Foundation recommends to use Debian-based Linux operating system (Raspbian). Other third-party operating systems also available via the official website include Windows 10 IoT Core, RISC OS, Ubuntu MATE and specialized distributions for the Kodi media centre and classroom management.

Other operating systems (not Linux-based)

- RISC OS Pi, FreeBSD, NetBSD , OpenBSD Plan 9 from Bell Labs and Inferno (in beta), Windows 10 IoT Core , xv6, Haiku, etc.

Other operating systems (Linux-based)

- Android Things, Arch Linux ARM, openSUSE, SUSE Linux Enterprise Server, Raspberry Pi, Fedora Remix, Gentoo Linux, CentOS for Raspberry Pi 2 and later, Devuan, Maemo Leste, RedSleeve (a RHEL port), Slackware ARM, etc.

Third party application software

- Minecraft, RealVNC, UserGate Web Filter, AstroPrint, C/C++ Interpreter Ch, Mathematica & the Wolfram Language

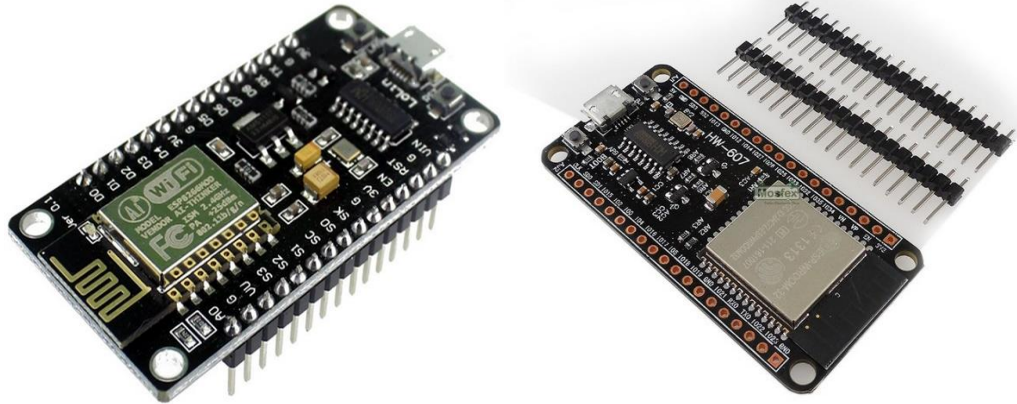
Software development tools

- Ninja-IDE, Object Pascal, Processing, Scratch, Squeak Smalltalk, V-Play Game Engine, Xojo, C-STEM Studio, Arduino IDE, Algoid, BlueJ, Greenfoot, Julia, Lazarus, LiveCode

## 3.3 NodeMCU

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and the hardware is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the

development kits. The firmware uses the Lua scripting language. To upload code and compile on NodeMCU can be used by modifying the Arduino IDE.



**Figure 3.4 NodeMCU**

### **3.3.1 Specification of NodeMCU**

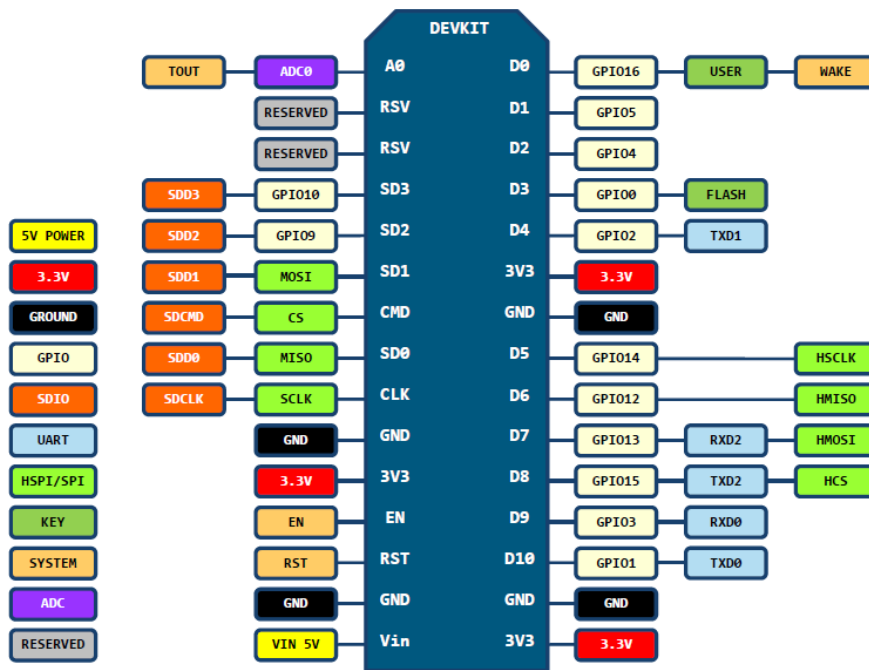
- Voltage:3.3V
- Wi-Fi Direct (P2P), soft-AP
- Current consumption: 10uA~170mA
- Flash memory attachable: 16M B max (512K normal).
- Integrated TCP/IP protocol stack
- Processor: Tensilica L106 32-bit
- Processor speed: 80~160MHz
- RAM: 32K + 80K
- GPIOs: 17 (multiplexed with other functions)
- Analog to Digital: 1 input with 1024 step resolution
- +19.5dBm output power in 802.11b mode
- 802.11 support: b/g/n
- Maximum concurrent TCP connections: 5

**Table 3.1 Pin Mapping Table of NodeMCU**

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

Pin Mapping and Definition of NodeMCU is shown in Table 3.1 and Figure 3.5.

**PIN DEFINITION**



*D0(GPIO16) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.*

**Figure 3.5 Pin Definition of NodeMCU**

### 3.3.2 Code examples

The NodeMCU repository contains its own collection of elaborate code examples. Besides that, small code examples of NodeMCU for most functions and modules are as follows.

**a) Connect to an AP**

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

**b) Control GPIO**

```
////////////////////////////////////////////////////////////////  
ledPin = 1  
swPin = 2  
gpio.mode(ledPin,gpio.OUTPUT)  
gpio.write(ledPin,gpio.HIGH)  
gpio.mode(swPin,gpio.INPUT)  
print(gpio.read(swPin))  
////////////////////////////////////////////////////////////////
```



c) **HTTP request**

```
//////////////////////////////////////////////////////////////////  
-- A simple HTTP client  
conn = net.createConnection(net.TCP, 0)  
conn:on("receive", function(sck, payload) print(payload) end)  
conn:on("connection", function(sck)  
  sck:send("GET / HTTP/1.1\r\nHost: nodemcu.com\r\n"  
    .. "Connection: keep-alive\r\nAccept: */*\r\n\r\n")  
end)  
conn:connect(80, "nodemcu.com")  
//////////////////////////////////////////////////////////////////  
//////////////////////////////////////////////////////////////////  
http.get("http://nodemcu.com", nil, function(code, data)  
  if (code < 0) then  
    print("HTTP request failed")  
  else  
    print(code, data)  
  end  
end)  
//////////////////////////////////////////////////////////////////
```

d) **HTTP server**

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

e) **Connect to MQTT Broker**

```
////////////////////////////////////  
-- init mqtt client with keepalive timer 120sec  
m = mqtt.Client("clientid", 120, "user", "password")  
-- setup Last Will and Testament (optional)  
-- Broker will publish a message with qos = 0, retain = 0, data = "offline"  
-- to topic "/lwt" if client don't send keepalive packet  
m:lwt("/lwt", "offline", 0, 0)  
  
m:on("connect", function(con) print ("connected") end)  
m:on("offline", function(con) print ("offline") end)  
  
-- on publish message receive event  
m:on("message", function(conn, topic, data)  
  print(topic .. ":" )  
  if data ~= nil then  
    print(data)  
  end  
end)  
  
-- for secure: m:connect("192.168.11.118", 1880, 1)  
m:connect("192.168.11.118", 1880, 0, function(conn) print("connected") end)  
  
-- subscribe topic with qos = 0  
m:subscribe("/topic",0, function(conn) print("subscribe success") end)  
-- or subscribe multiple topic (topic/0, qos = 0; topic/1, qos = 1; topic2 , qos = 2)  
-- m:subscribe({"topic/0"]=0,["topic/1"]=1,topic2=2}, function(conn) print("subscribe  
success") end)  
-- publish a message with data = hello, QoS = 0, retain = 0  
m:publish("/topic", "hello",0,0, function(conn) print("sent") end)  
  
m:close();  
-- you can call m:connect again  
////////////////////////////////////
```

#### f) UDP client and server

```
/////////////////////////////////////////////////////////////////  
-- a udp server  
s=net.createServer(net.UDP)  
s:on("receive",function(s,c) print(c) end)  
s:listen(5683)  
  
-- a udp client  
cu=net.createConnection(net.UDP)  
cu:on("receive",function(cu,c) print(c) end)  
cu:connect(5683,"192.168.18.101")  
cu:send("hello")  
/////////////////////////////////////////////////////////////////
```

### 3.4 Wireless Router

A wireless router is a device that performs the functions of a router and also includes the functions of a wireless access point. It is used to provide access to public network (Internet) or a local area computer network. A portable Wi-Fi router is just like the internet box at home, but instead of being attached to a phone cable, they have a SIM card inside instead. If you get an 'unlocked' portable Wi-Fi device, this means you can use any SIM card, from anywhere in the world, inside it. The benefit of this is that you can therefore always get the lowest rates as you can use a worldwide data SIM or a local SIM in it. A portable Wi-Fi device will enable you to set up your own private internet connection, practically anywhere in the world, on at least 10 devices simultaneously, including phones, laptops, tablets and iPads, games consoles and cameras etc. Figure 3.6 represents the photo of some wireless routers.

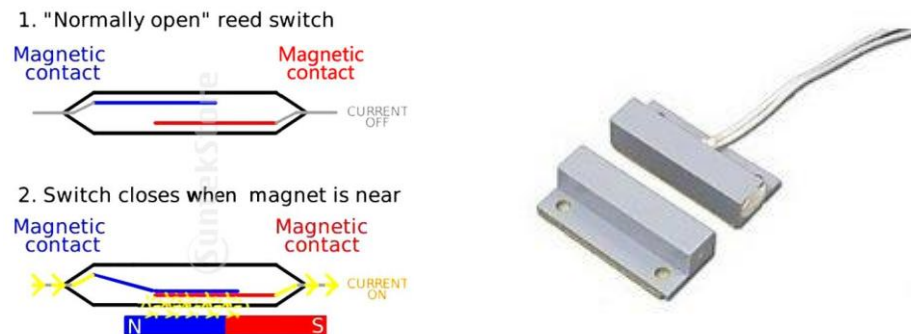


**Figure 3.6 Sim Wireless Router**

### 3.5 Sensors and Components of the System

#### 3.5.1 Magnetic Door Sensor

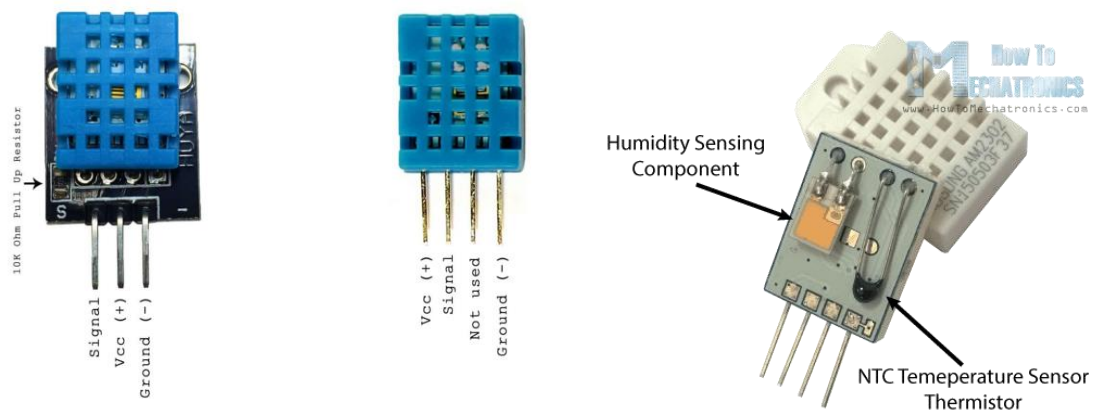
Magnetic Door Sensor is essentially a reed switch, encased in an ABS plastic shell. Normally the reed is 'open' and there is no connection between the two wires. The other half is a magnet. If the magnet is less than 13mm (0.5") away, the reed switch closes. They're often used to detect a door or drawer is open or closed, that why they have mounting tabs and screws. Magnetic Door Sensor can be seen in Figure 3.7.



**Figure 3.7 Magnetic Door Sensor**

### 3.5.2 DTH 11 Sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. 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. The structure of DTH11 is shown in Figure 3.8.



**Figure 3.8 DTH11 Sensor**

Specifications of the DTH11 sensor:

- Supply Voltage: +5
- Temperature range: 0-50 °C error  $\pm 2$  °C
- Humidity: 20-90% RH  $\pm 5$ % RH error
- Interface: Digital

### 3.5.3 Flame Sensor Module

A flame detector is a sensor designed to detect and respond to the presence of a flame or fire, allowing flame detection. Responses to a detected flame depend on the installation, but can include sounding an alarm, deactivating a fuel line (such as a propane

or a natural gas line), and activating a fire suppression system. A flame sensor module that consists of a flame sensor (IR receiver), resistor, capacitor, potentiometer, and comparator LM393 in an integrated circuit. It can detect infrared light with a wavelength ranging from 700nm to 1000nm. Figure 3.9 represent the flame sensor and it's pins.



**Figure 3.9 Flame Sensor Module**

### 3.5.4 RFID Module

Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to objects. The tags contain electronically-stored information. Passive tags collect energy from a nearby RFID reader's interrogating radio waves. Active tags have a local power source (such as a battery) and may operate hundreds of meters from the RFID reader. RFID sensor module and RFID card are shown in Figure 3.10.

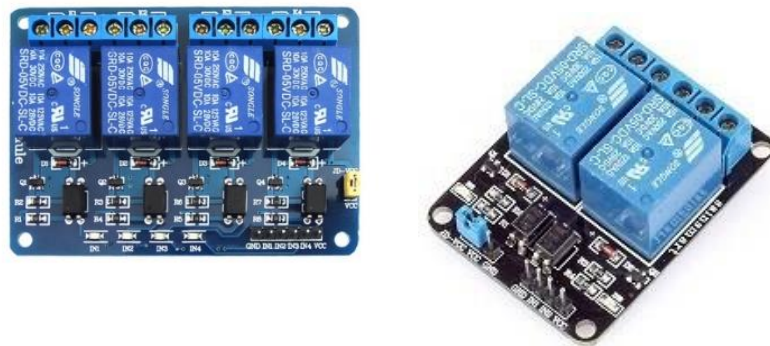


**Figure 3.10 RFID Reader Module and Tag**

RFID tags are used in many industries, for example, an RFID tag attached to an automobile during production can be used to track its progress through the assembly line; RFID-tagged pharmaceuticals can be tracked through warehouses; and implanting RFID microchips in livestock and pets allows for positive identification of animals. However, the RFID tag does not have to be scanned directly, nor does it require line-of-sight to a reader. The RFID tag it must be within the range of an RFID reader, which ranges from 3 to 300 feet, in order to be read. RFID technology allows several items to be quickly scanned and enables fast identification. RFID technology may be used in a variety of applications including: Passports, Smart cards, Airplane luggage, Toll booth passes, Home appliances, Merchandise tags, Animal and pet tags, Automobile key-and-lock, Monitoring heart patients, Pallet tracking for inventory, Telephone and computer networks, Operation of spacecraft and satellites.

### 3.5.5 Relay

A relay is an electromagnetic switch; hence, its heart is the electromagnet, which is powered by a small current that acts as a lever or as the switch itself. This makes it possible to allow relatively small electric currents to leverage and control much larger electrical currents.



**Figure 3.11 Relay Module**

Sensors are sensitive devices, and they only produce small amounts of electric currents, but in order for a sensor to drive larger pieces of equipment it needs something

that would switch on this equipment by allowing larger currents to flow. The small control current is used to energize the electromagnet, which pulls the armature toward it. The armature makes contact with the other end of the circuit, which completes the circuit and allows current to flow. There are many types of relays Electromagnetic Relays, Attraction Type Electromagnetic Relays, Induction Type Relays, Hybrid Relay, Thermal Relay, Reed Relay. Four-channel and Two-channel Relay modules are shown in Figure 3.11.

### 3.5.6 Buzzer

A buzzer or beeper is an audio signaling device, which may be electromechanical, mechanical, or piezoelectric (piezo for short). Typical uses of beepers and buzzers include timers, alarm devices and confirmation of user input such as keystroke or a mouse click. An example of a purely mechanical buzzer is a joy buzzer and require drivers to run it. Doorbells are other examples of them. Figure 3.12 represents buzzer that used in this system.

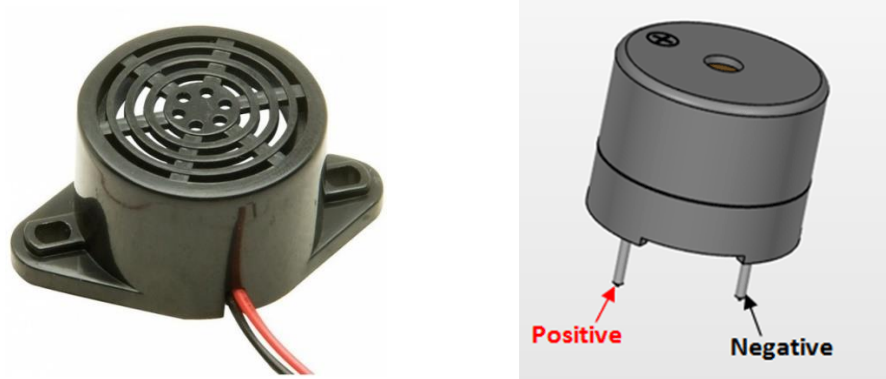


Figure 3.12 Buzzers

### 3.5.7 Keypad

A keypad is a set of keys or buttons bearing symbols, digits and/or alphabetical letters placed in order on a pad, which can be used as an efficient input device. A keypad may be purely numeric, as that found on a digital door lock or a calculator, or alphanumeric as those used on phones. Aside from the row of number keys found on the upper portion



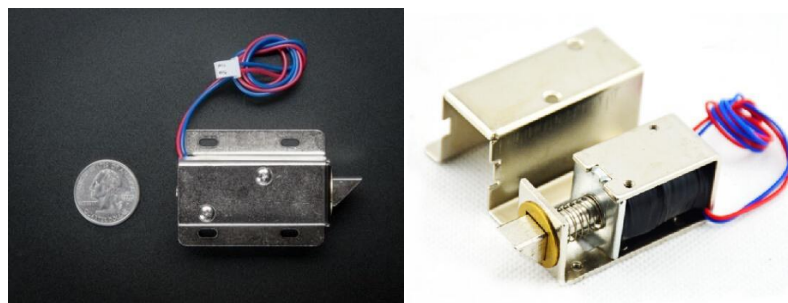
of a computer keyboard, a separate numerical pad is also located on the right side for efficient data entry. Figure 3.13 shows some type of keypads.



**Figure 3.13 Keypads**

### **3.5.8 Solenoid Electronic lock**

An electronic lock (or electric lock) is a locking device which operates by means of electric current. Electric locks are sometimes stand-alone with an electronic control assembly mounted directly to the lock. Electric locks may be connected to an access control system. Electric locks are working by using magnets, solenoids, or motors to actuate the lock by either removing or supplying power. It operates when 9-12VDC is applied. Figure 3.14 shows how the solenoid electronic lock look like.



**Figure 3.14 Solenoid Electronic lock**

### **3.5.9 Webcam**

A webcam is a video camera that feeds or streams its image in real time to or through a computer to a computer network. When "captured" by the computer, the video

stream may be saved, live viewed or sent on to other networks travelling through systems (end-devices) such as the public network (Internet) and e-mailed as an attachment. Difference from IP camera a webcam is generally connected by a USB cable, or similar cable, or built into computer hardware, such as laptops. Figure 3.15 shows some type of Webcams.

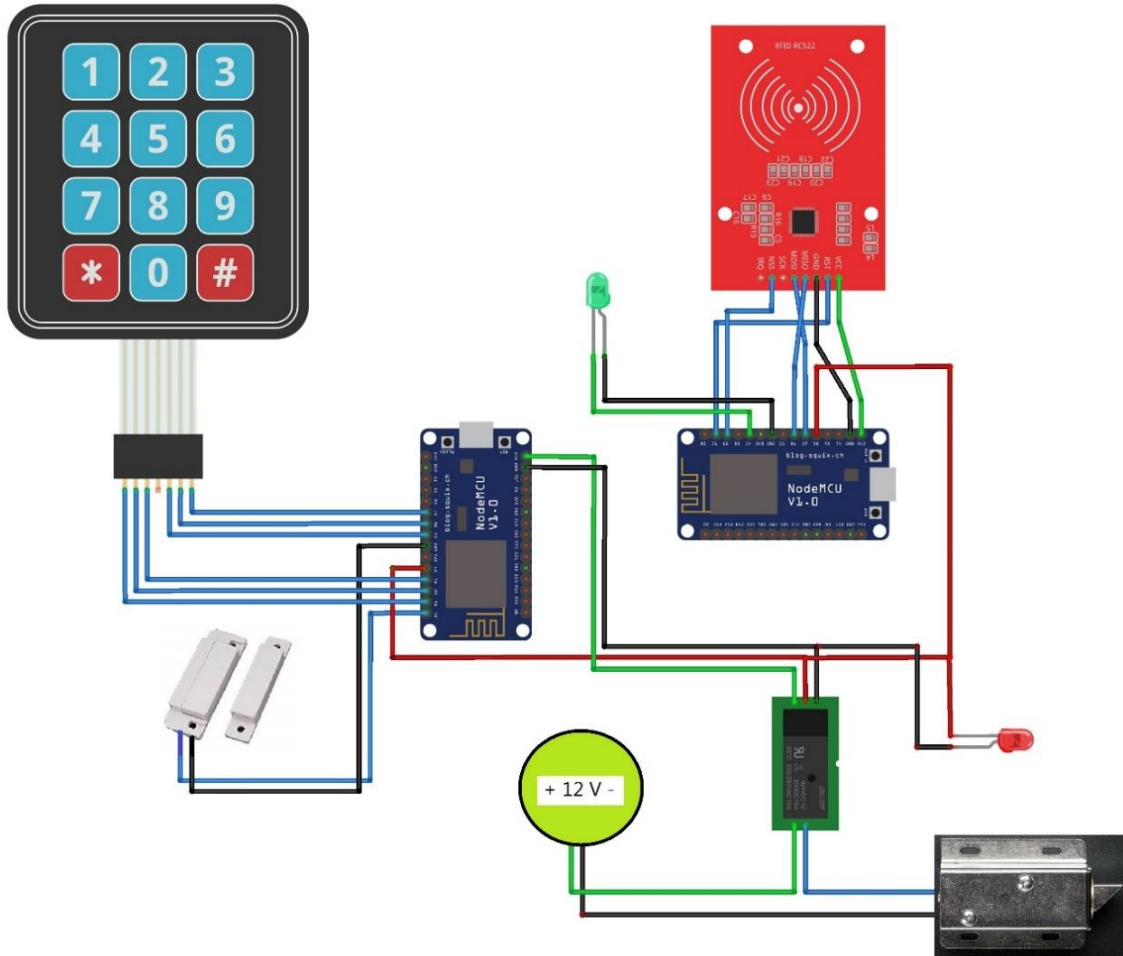


**Figure 3.15 Webcams**

### **3.6 Design Implementation of IoT-based Home Control and Monitoring System**

This system consists of Raspberry Pi 3 Model B (RPi), three NodeMCUs, Sim Router, two Magnetic Door sensors, Temperature and humanity sensor (DTH11), IR Flame sensor, Buzzer, Keypad, RFID Card Sensor Module, two Solenoid Electronic lock, Webcam, three 4 channel Relays Module, two Fans and five LED lights. This system have three main parts, so the circuit design of the system is connected as three main parts separately. In Front door part Keypad is connected at 6 pins (D1,D2,D3,D5,D6,D7), Magnetic Door sensor is connected at 2 pins (D0, GND) and Relay Module is connected to three pin (Vin,D4,GND) of first NodeMCU. For the second one, RFID Module is connected to the 6 pins (D1,D2,D6,D7,GND,3v3), one Led is connected to 2 pins (D4,GND) and D8 pin is connected with Relay Input pin. One Led is connected to the relay input pin and GND that can confirm relay is working or not. One pin of Electronic lock is

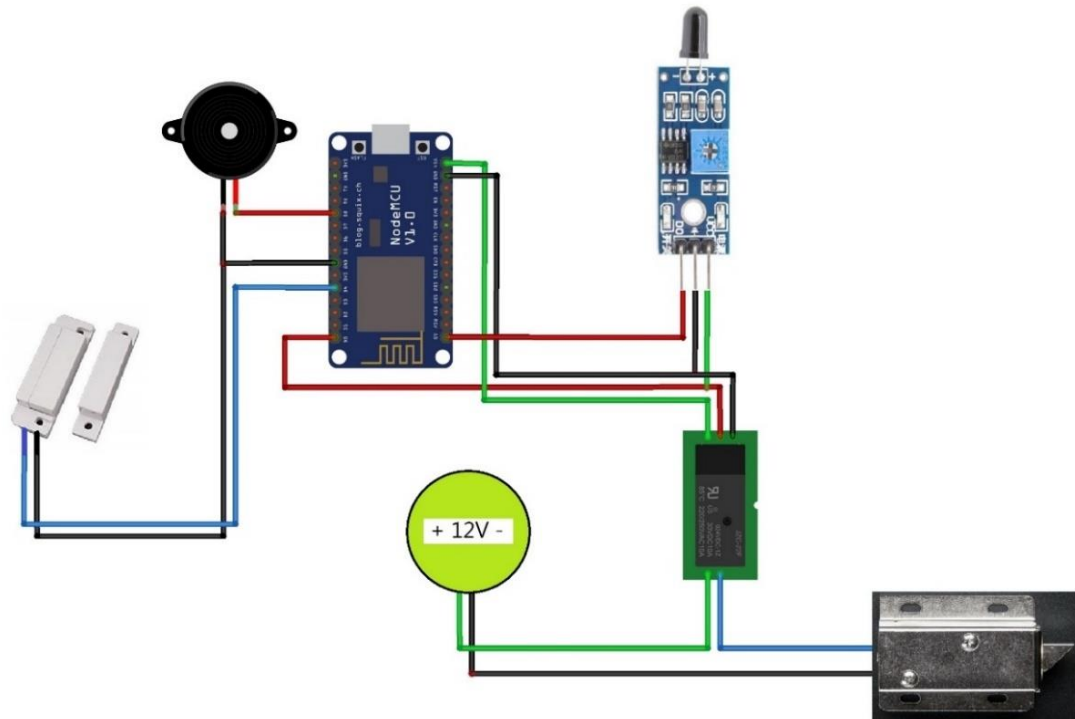
connected to the relay and another pin is connected to the ground pin of Power Supply (12V). The Design Layout of Front door part is shown in Figure 3.16.



**Figure 3.16 Design Layout of Front Door Part**

The Back door part is running like Front door part but there is no Keypad and RFID module for multi-access to door lock and fire alert system is extended in this part. Magnetic Door sensor is connected at 2 pins (D4, GND), IR Flame sensor is connected to three pins (A0, GND, Vin), Buzzer is connected to 2 pins (D8, GND), and Relay Module is connected to three pins (Vin, D0, GND) of NodeMCU. One pin of Electronic lock is connected to the relay and another pin is connected to the ground pin of Power Supply (12V). By using Magnetic Door sensor to know the status of door that is OPEN/CLOSE. And with the help

of IR Flame sensor can sense the palace is On Fire or not. So, we can easily know the status of doors and home environment. The Design Layout of Back door part is shown in Figure 3.17.

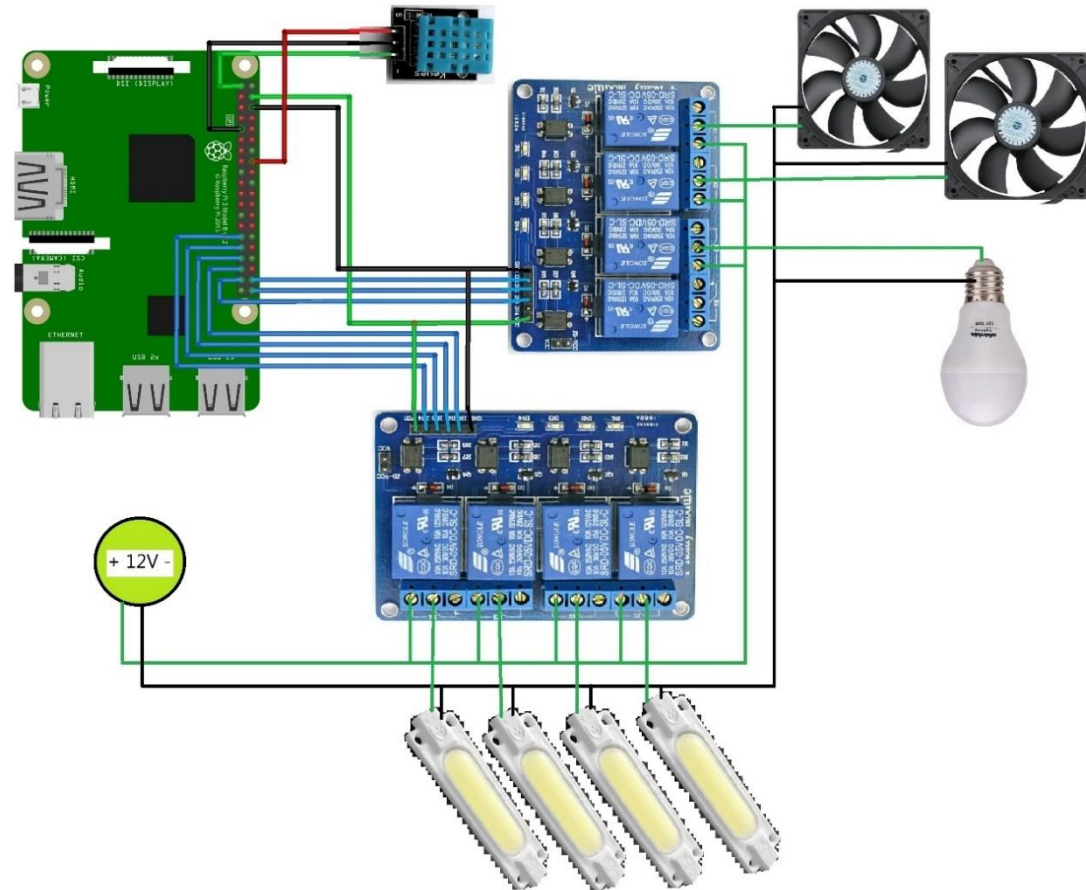


**Figure 3.17 Design Layout of Back Door Part**

Finally, in the Main part, Temperature and Humanity sensor is connected with 3 pins (3v3, GPIO23, Ground) and Relay modules are connected with 7 pins (GPIO6, GPIO13, GPIO19, GPIO26, GPIO20, GPIO21) of Raspberry Pi board. All the lights and fans are connected to the relay and the ground pin of Power Supply (12V). The Schematic and Breadboard circuit diagrams of Main part is shown in Figure 3.18. All the main boards of this system (Rpi, NodeMCUs) are connected with Power Supply(5V).

There are many components used in this system. These components are Raspberry Pi 3 Model B (Rpi), three NodeMCUs, SD card, Sim Router, two Magnetic Door sensors, Temperature and humanity sensor (DTH11), IR Flame sensor, Buzzer, Keypad, RFID Card Sensor Module, two Solenoid Electronic lock, Webcam, three 4 channel Relays Module,

Webcam, small access (led, jumper), Power Supply, two Fans and five LED lights. So, this system is very low cost than other home automation system with similar features.



(a) Breadboard Design

Figure 3.18 Circuit Diagrams of Main Part

## CHAPTER 4

# HARDWARE AND SOFTWARE IMPLEMENTATION

### 4.1 Sensors and Camera Interfacing

In this IoT-based Home Control and Monitoring system, the state of the room and home appliances sensing through sensors and sending the collected information or sensing data to the central unit. The different sensors and components being used are:

- i. **DHT11 digital temperature and humidity sensor** is a composite Sensor contains a calibrated digital signal output of the humidity and temperature. This sensor contains a high-performance 8-bit microcontroller connected with resistive for sensing wet and NTC for sensing temperature. So, it is used for sensing the temperature and humidity of Room.
- ii. **Magnetic-Door sensor** is essentially a reed switch, cover with ABS plastic shell. Normally the reed is 'open' (no connection between the two wires). The other half is a magnet. The reed switch closes if the magnet is less than 0.5" away. They're often used to detect when a door or drawer is open or closed, that is why they have mounting tabs and screws.
- iii. **An electronic lock (or electric lock)** is which operates by means of electric current. Operating the lock can be as simple as using a switch. In this system using RFID and Keypad to access the Electric lock to lock or unlock.
- iv. **RFID** stands for Radio-Frequency Identification. The acronym refers to small electronic devices that consist of a small chip and an antenna. The RFID device must be scanned to retrieve the identifying information. The electronic lock unlocks if identifying information for RFID card is correct with predefined information.
- v. **Keypad** is a set of keys or buttons bearing digits, symbols and/or alphabetical letters placed in order on a pad, which can be used as an efficient input device. The

electronic lock also can unlock electronic lock by typing correct key string (password) on keypad.

**vi.** The **fire detection sensor** consists of flame sensor which is designed to detect and respond to the presence of a flame or fire. If there is any fire accident this will help to take immediate actions like activating the alarm (buzzer) and sending the fire alarm status to MQTT broker.

**vii.** **Webcam** is attached to run Live Streaming Video server on **RPi** to view and check the real-time status of home environment like CCTV. If user get any notifications about home situation user can immediate action remotely using smart phone through Internet [5].

## 4.2 Raspberry Pi and NodeMCU

The core of this home control and monitoring system is minicomputer (RPi 3 model B). The setting up of RPi consists of writing raspbian OS image to SD card. After the OS preparation need to configure RPi using Raspi-conFigure command. For configure RPi by using **Putty** can enter to the RPi terminal and can also can enter into GUI mode using **Remote Desktop Connection** as shown in Figure 4.1.

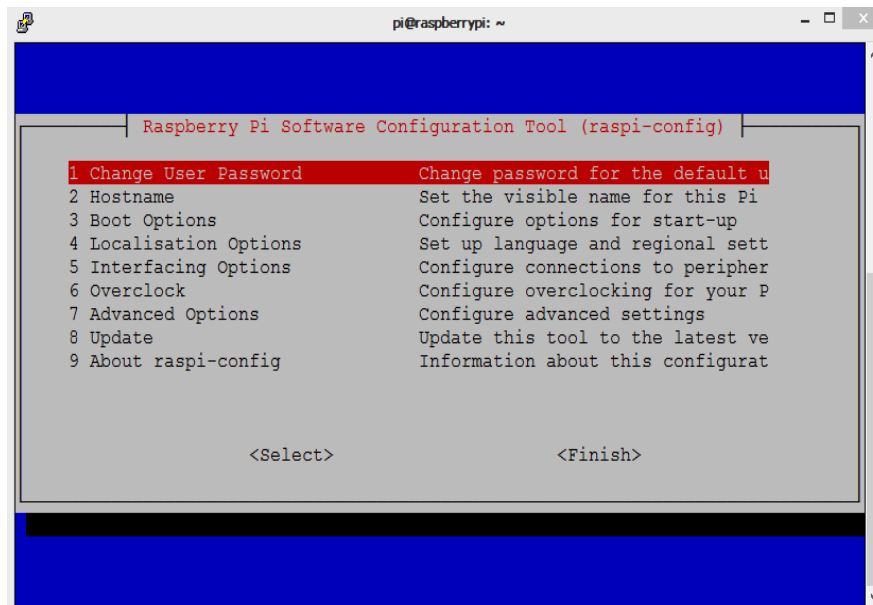
To start RPi is needed to prepare the OS on the SD Card, setup the hardware and connect everything for initial setup.

1. Format the micro SD Card (*use FAT32 file system*)
2. First download Raspbian image, an OS based on Debian optimized for RPi hardware. Download the image from:<https://www.raspberrypi.org/downloads>
3. Install Raspbian image on the SD Card using Win32 image writer. Follow the official guide:<https://www.raspberrypi.org/>
4. Insert the micro SD card into the RPi and attach the keyboard, mouse, monitor and network cables.

Finally attach the power cable and turn on the power to start RPi.



(a) Prepared RPi for initial setup



(b) Output of raspi-conFigure command

**Figure 4.1 Preparation result of Raspberry Pi**

And prepare to run NodeMCUs for Front door part and Back door part of the system by following step:

To get started we need to download the arduino IDE and some necessary drivers.

- The arduino IDE can be downloaded via this link - [www.arduino.cc](http://www.arduino.cc)
- After downloading the arduino IDE navigate to File => Preferences and in the additional boards URLs add the below URL



[http://arduino.esp8266.com/versions/2.3.0/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/versions/2.3.0/package_esp8266com_index.json)

- After that navigate to the Tools>Boards>Board Manager and scroll down the list till you can find ESP8266 click on it and then hit install. Now you have setup the arduino IDE to work along with the node MCU.
- Additionally, you need to install the drivers if you are on a windows-based computer. The drivers can be downloaded and install from the below link like Figure 4.2.

[http://arduino.esp8266.com/versions/2.3.0/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/versions/2.3.0/package_esp8266com_index.json)

- Once the driver is installed it is time to program the NodeMCU, for this select the NodeMCU 1.0 board from the Tools => Boards menu of the arduino IDE.
- Now need to find the specific port to which the NodeMCU is connected to the computer. This can be found in device manager. In Tools => Ports make sure you select the right port as seen before.
- Then put the specific code in the arduino IDE and hit upload.
- The code will take a few minutes to upload and then you should see the LED blink at the interval set in the code.

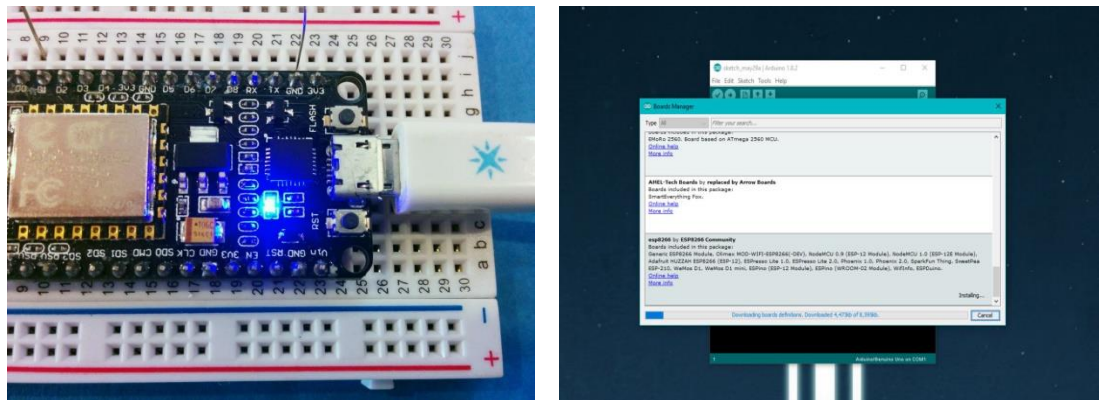
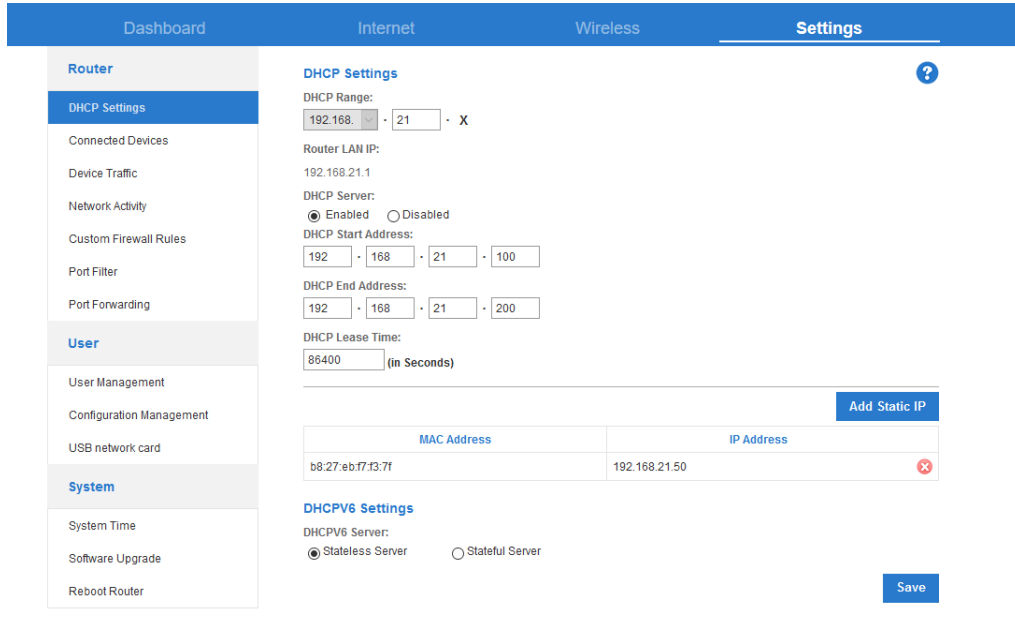


Figure 4.2 Preparation result of NodeMCU

### 4.3 Wi-Fi Router Configuration

Make sure the Wi-Fi is running with certain address and enable the DHCP option . For getting static ip address for RPi configured with MAC address and automatically giving DHCP ip addresses to NodeMCUs. The ip address of the Raspberry pi is “192.168.21.50”

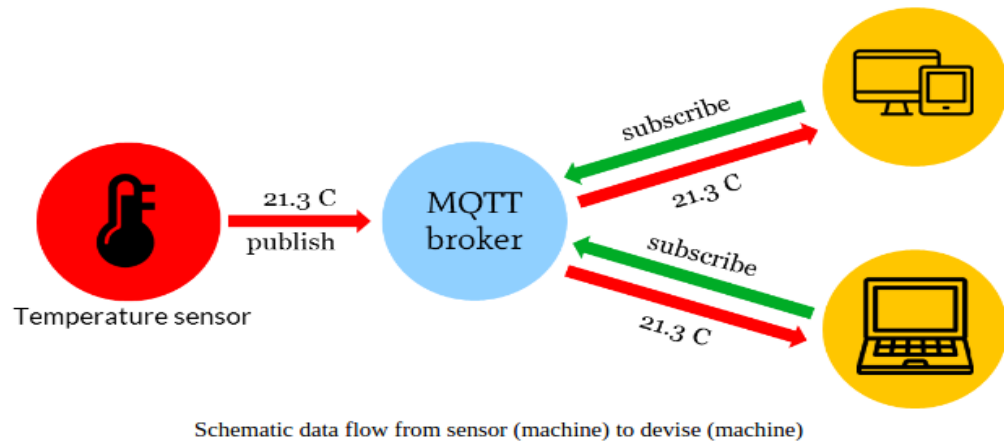
and the DHCP IP address range is “192.168.21.100 – 200”. The setting of Wi-Fi configuration is shown in Figure 4.3.



**Figure 4.3 Wi-Fi configuration setting**

## 4.4 MQTT Protocol

There are three components in MQTT broker, publisher and subscriber (shown in Figure 4.4). It has very much secure and accurate the process of receiving and publishing the data. Whenever the user wants to check or go through any data it sends the request to broker and upon receiving the request it sends to the publisher, it responds to the requests and sends the data that is requested by the subscriber and hence publishes the data, in overall process the communication is secure and up to the topic of interest. MQTT broker acting like a guard which permitted allowing only the data which are requested thereby saving the flow of ambiguous data. Mosquitto broker supports this ACL feature through auth plugging. An access control list (ACL), is a list of permissions attached to an object [6].



**Figure 4.4 MQTT Process**

#### 4.4.1 MQTT Method and Testing

To continue MQTT Method and Testing install Mosquitto proper as follow: Install mosquitto server or broker on the RPi, prepare mosquitto-clients are the command-line clients and python-mosquitto are for the Python bindings [7][8].

The next step is to test the mosquitto and the publisher

In separate terminal windows do the following:

Start the broker: mosquitto

i > Start the command line subscriber:

- **mosquitto\_sub -h 192.168.21.50 -u musert -P mpwd -t 'Door/+'**

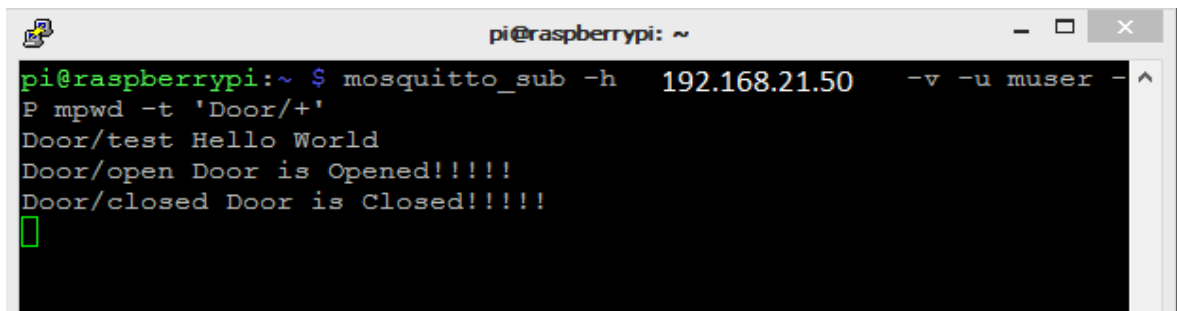
ii > Publish test messages with the command line publisher:

- **mosquitto\_pub -h 192.168.21.50 -u muser -P mpwd -t Door/test -m "Hello World"**

- **mosquitto\_pub -h 192.168.21.50 -u muser -P mpwd -t Door/open -m "Door is Opened!!!!!"**

- `mosquitto_pub -h 192.168.21.50 -u muser -P mpwd -t Door/closed -m "Door is Closed!!!!!"`
- `mosquitto_pub -h 192.168.21.50 -u wuser -P wpwd -t Door/test -m "Wrong user and pwd"`
- `mosquitto_pub -h 192.168.21.50 -u muser -P mpwd -t Fire/ert -m "Wrong Topic"`

iii > So Figure 4.5 (a) and (b) show the MQTT testing output of the Mosquitto subscriber and publisher.

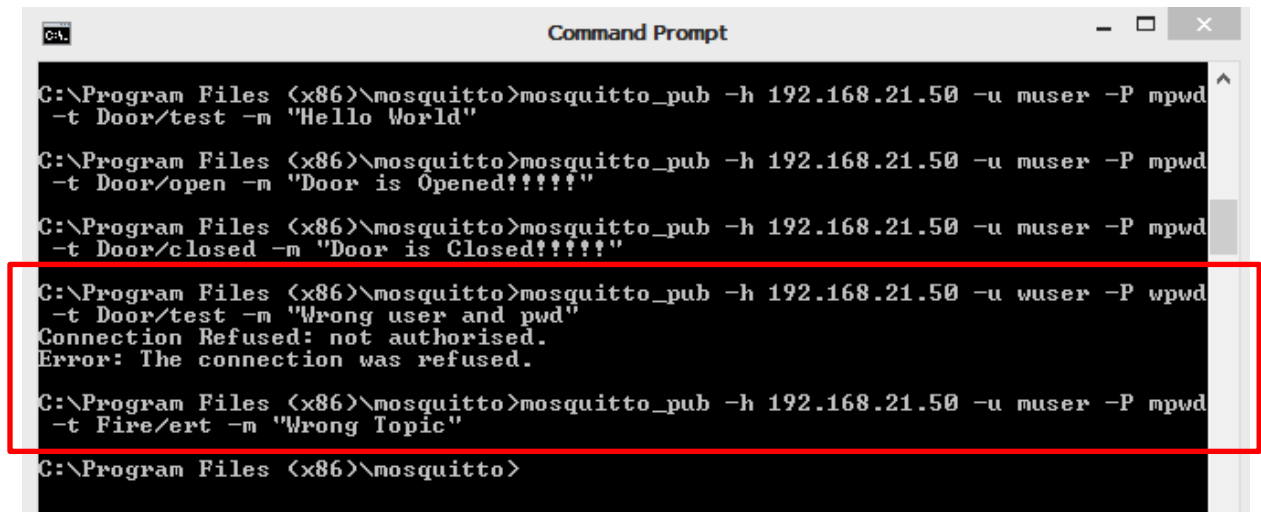


```

pi@raspberrypi: ~
pi@raspberrypi:~ $ mosquitto_sub -h 192.168.21.50 -v -u muser -P mpwd -t 'Door/+'
Door/test Hello World
Door/open Door is Opened!!!!
Door/closed Door is Closed!!!!

```

(a) Mosquitto subscriber



```

C:\Program Files (x86)\mosquitto>mosquitto_pub -h 192.168.21.50 -u muser -P mpwd -t Door/test -m "Hello World"
C:\Program Files (x86)\mosquitto>mosquitto_pub -h 192.168.21.50 -u muser -P mpwd -t Door/open -m "Door is Opened!!!!!"
C:\Program Files (x86)\mosquitto>mosquitto_pub -h 192.168.21.50 -u muser -P mpwd -t Door/closed -m "Door is Closed!!!!!"
C:\Program Files (x86)\mosquitto>mosquitto_pub -h 192.168.21.50 -u wuser -P wpwd -t Door/test -m "Wrong user and pwd"
Connection Refused: not authorised.
Error: The connection was refused.
C:\Program Files (x86)\mosquitto>mosquitto_pub -h 192.168.21.50 -u muser -P mpwd -t Fire/ert -m "Wrong Topic"
C:\Program Files (x86)\mosquitto>

```

(b) Mosquitto publisher

Figure 4.5 MQTT Testing

Total five publisher messages are published from publisher side but only first three published messages accepted by subscriber side. Because of wrong user and password at fourth published message and different publishing topic at fifth published message .

Depending on the result of mqtt testing used four topics to control and monitor the Front door part and back door part from RPi. The topics are:

- Sensor/garage/state1
- Sensor/garage/action1
- Sensor/garage/state2
- Sensor/garage/action2

These above four topics are configured in the arduino-code written in the NodeMCUs of Front door and Back door parts. The static IP address for the Mqtt server is “192.168.21.50” and username and password connect to mqtt server is “muser” and “mpwd”.

## **4.5 IFTTT, Remot3.it, Weather Underground**

### **4.5.1 IFTTT**

IFTTT stands for If This, Then That. IFTTT is both a website and a mobile app. With the help of IFTTT we can connect all of our "services" together so that tasks are automatically completed. There are numerous ways to connect all of services - and the resulting combinations are called "Applets". Applets essentially automate daily workflow, whether it's managing smart home devices or apps and websites

To get started is needed to create IFTTT account. Open IFTTT and create new free account. And then create some applets to inform user the conditions of home, environments by using G-mails, SMS messages and IFTTT notifications. Applets make for this home automation system is described in Figure 4.6. Also, can see the API key to connect with Home Assistant platform in Figure 4.7.

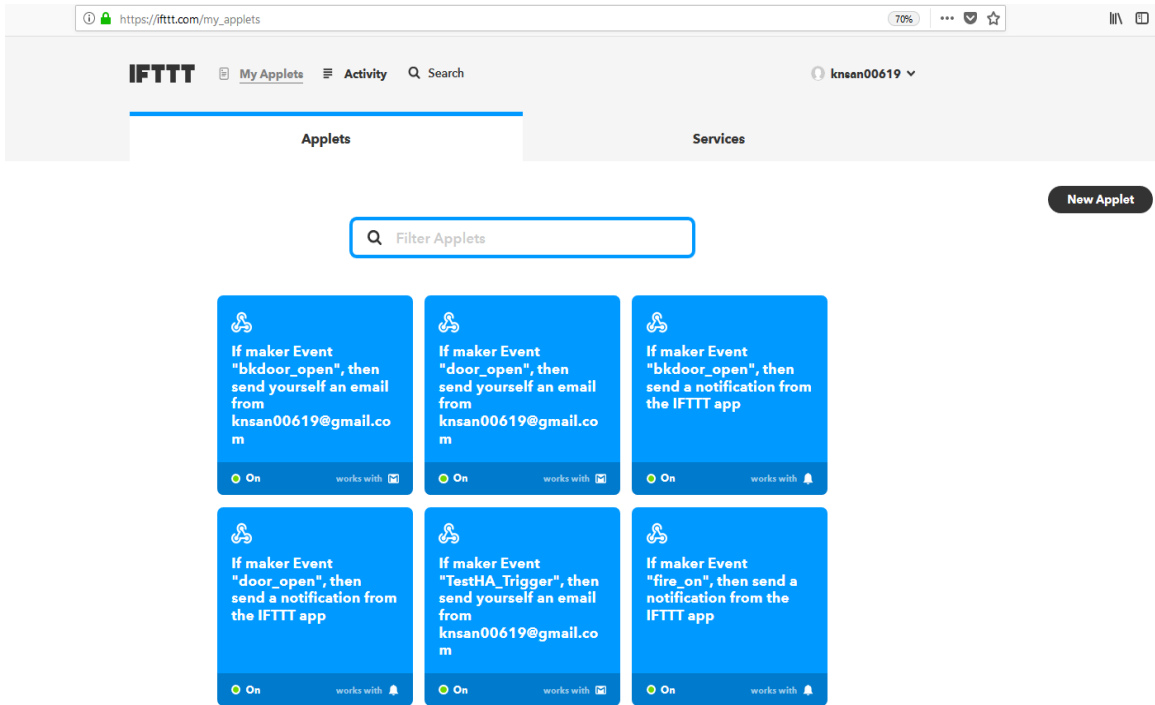


Figure 4.6 IFTTT Applets

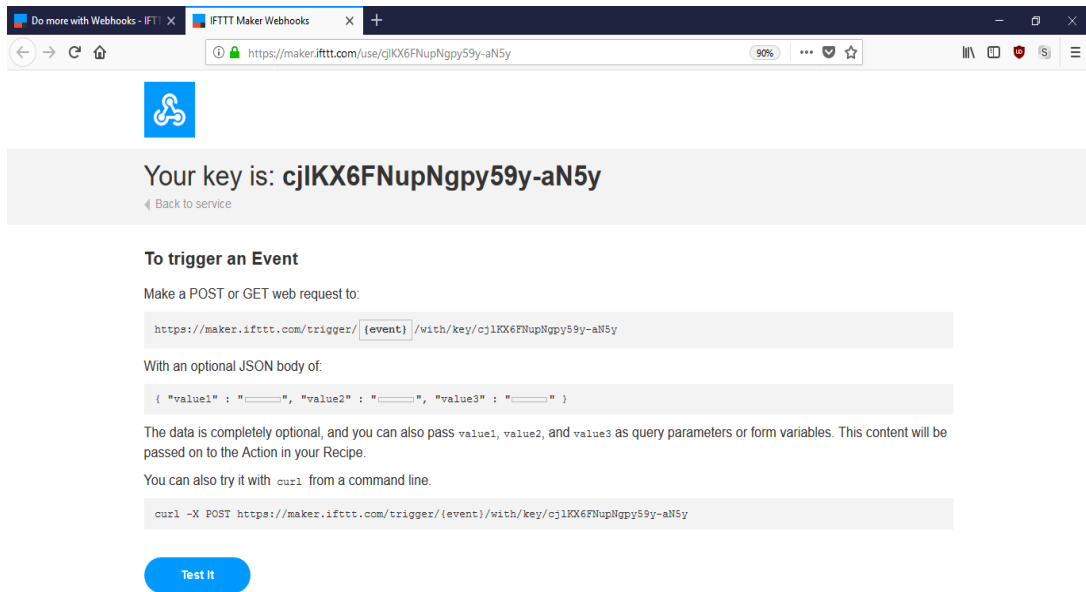


Figure 4.7 IFTTT API key

### 4.5.2 Remot3.it

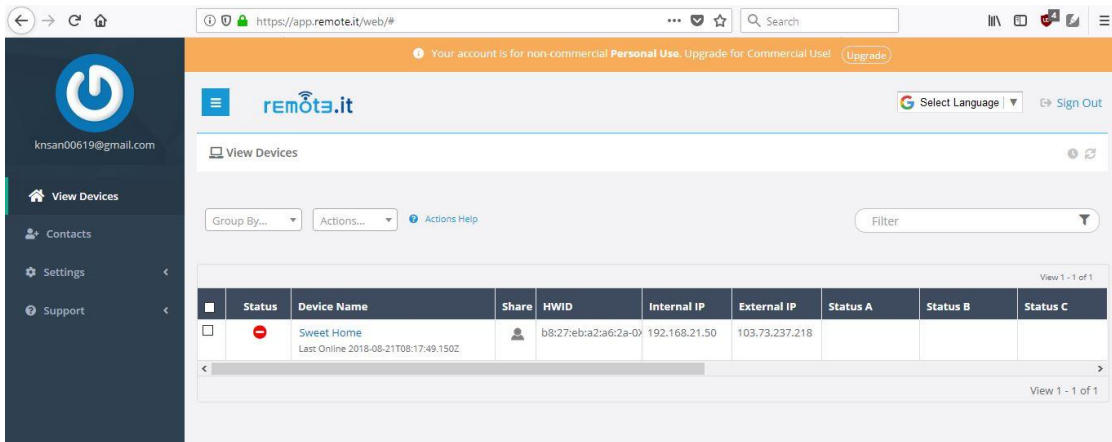
For different Raspberry Pi projects related to IoT or home automation we need device to be accessible from Internet to get more control over it and general flexibility. This can be achieved in different ways, the most common (I believe) is port forwarding when you setup your home network router to open custom port and map it to your device IP address. This is common and popular solution however it has some security concerns. That is why in this article I want to share alternative option. This option is called remot3.it, it provides remote access to your Raspberry Pi and creates network tunnels to services running on your Raspberry Pi, such as HTTP, VNC, SSH and accesses them over the Internet. Remot3.it is a successor of Weaved (now leads to remot3.it web site).

Remot3.it features:

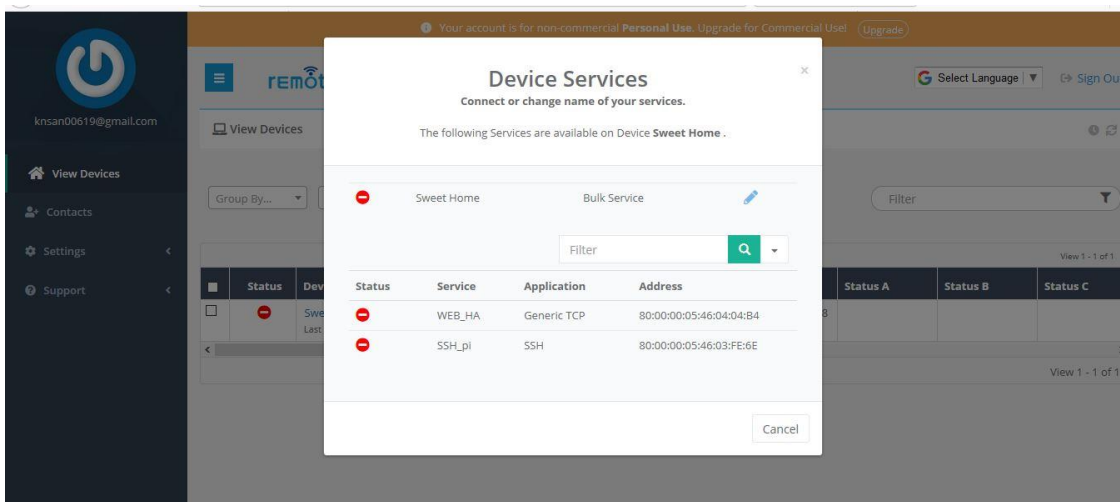
- No Port Forwarding
- No Static IP Needed
- Secure Connections
- Simple Setup
- Bulk Manage

Let's do a quick setup and installation to see how remot3.it actually works.

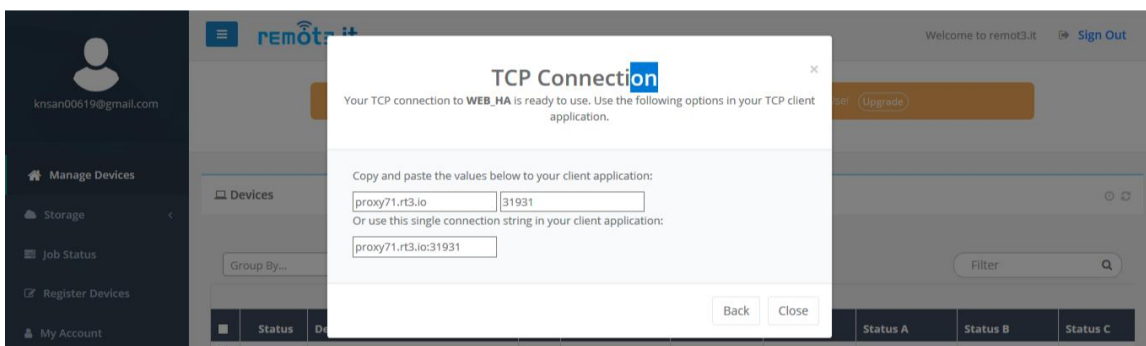
To get started we need to create remot3.it account. Open remot3.it and create new free account, if you do not have it already. The procedure is straightforward and after email verification you can proceed to installation. Firstly, install the weavedconnectd package with “`sudo apt-get install weavedconnectd`” command. After that setting up with “`sudo weavedinstaller`” command. And making SSH and Web services access from any places trough Internet. Remot3.it configured services and supported URL link for the service can be seen in Figure 4.8.



(a)



(b)



(c)

Figure 4.8 Configured Services and Connections of Remot3.it



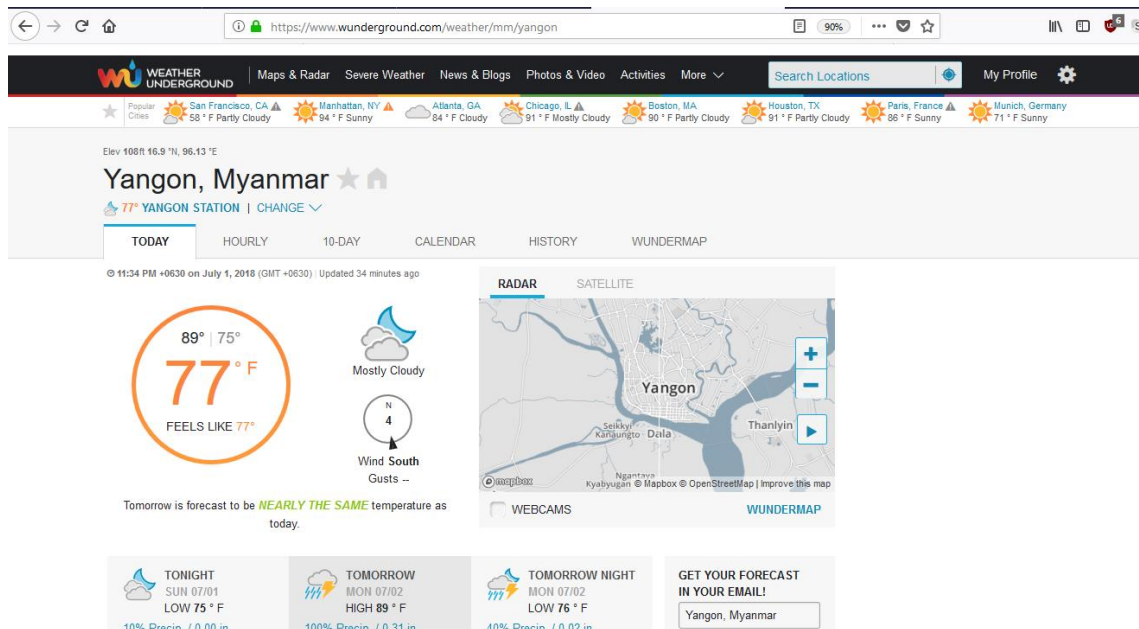
### 4.5.3 Weather Underground

Weather Underground is a commercial weather service providing real-time weather information via the Internet. Weather Underground provides weather reports for all of the major cities around the world on its website, as well as local weather reports for websites and newspapers. Its information comes from the National Weather Service (NWS), and over 250,000 of personal weather stations (PWS).

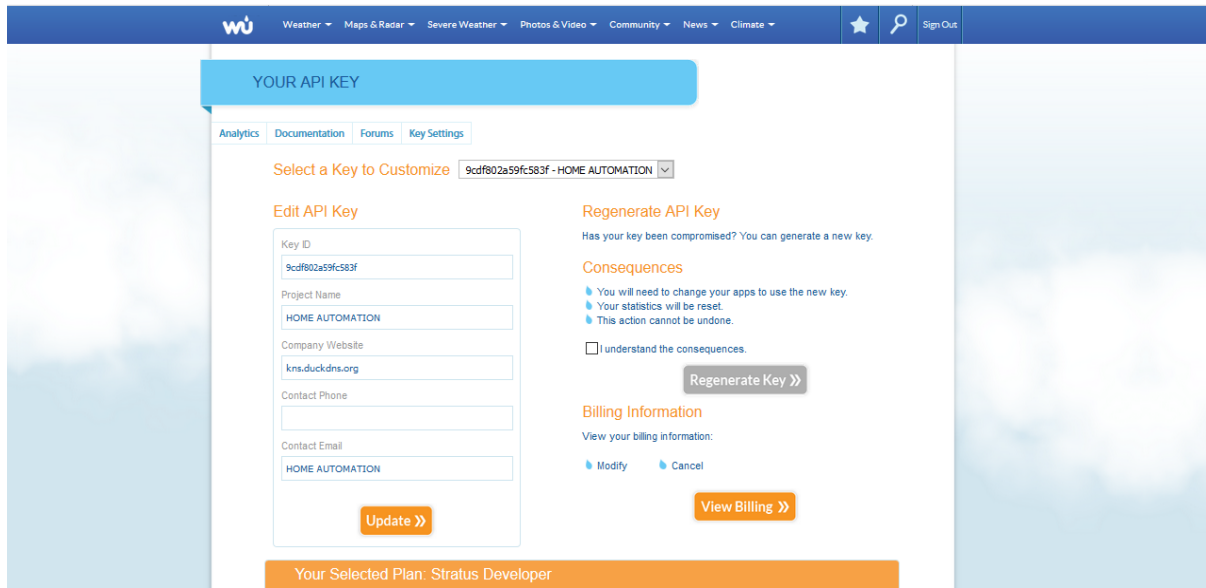
To use Weather Underground Service, follow the given steps:

- Go to <http://www.wunderground.com/weather/api/?MR=1>.
- Click “Sign Up for FREE!”.
- Create an account and click the link sent to you in a validation email to activate your account and Sign in
- Go to Pricing and select the free Stratus Plan (default selection). You get 500 API calls per day for \$0. There is no credit card required to get the Developer level API.
- Click “Purchase Key”
- Fill out the form and submit it to get your API key

After that using API key from Figure 4.9 connect with the Home Assistant platform.



(a)

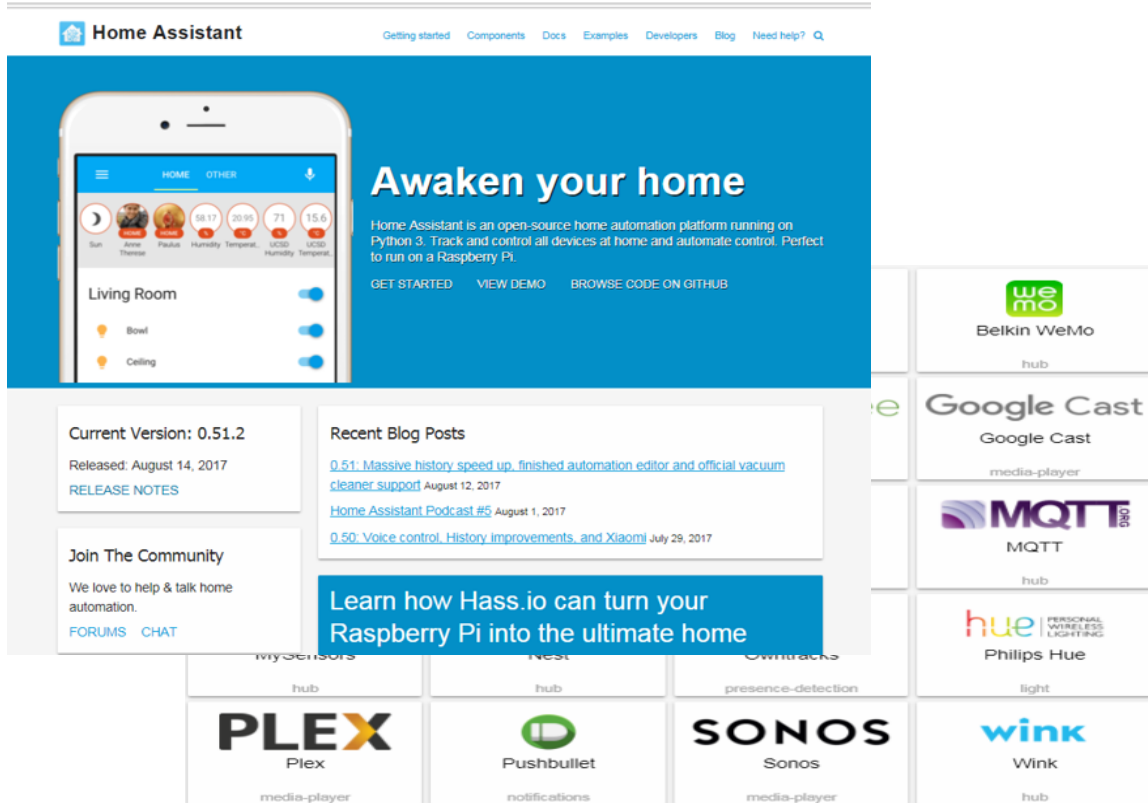


(b)

**Figure 4.9 WUI and API key of Weather Underground Service**

## 4.6 Home Assistant

Home Assistant is a free and open-source home automation platform running with Python. It can help you to control all of your devices from a single, user interface. Furthermore, you can control and track all your data without having to store them on the cloud. In my system Home Assistant is install on raspberry pi and configure with some components (IFTTT, MQTT, Weather Underground and Remot3.it). User can easily can control and monitor from Web User Interface(WUI) through end-devices (Smart phones, Tablets, PCs, Laptops) all the appliances and can get information about home that are connected with home assistant platform. Figure 4.10 represents the official page of Home Assistant and other installable plug-in available for Home Assistant Platform.



**Figure 4.10 Home Assistant Platform and Some Support Components**

### Step1: Install the Home Assistant software

Firstly, need to install the software. This information is taken from the site: <https://www.home-assistant.io/docs/installation/raspberry-pi/>

And connect to raspberry pi and enter the following to get the latest rasbian image:

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade -y
```

Install the dependencies that are needed for homeassistant:

```
$ sudo apt-get install python3 python3-venv python3-pip
```

Once done you will need to create the home assistant account and the virtual environment:

```
$ cd /srv
$ sudo mkdir homeassistant
$ sudo chown homeassistant:homeassistant homeassistant
$ sudo su -s /bin/bash homeassistant
$ cd /srv/homeassistant
$ python3 -m venv homeassistant_venv
$ source /srv/homeassistant/homeassistant_venv/bin/activate
$ exit
```

## **Step 2: Configure .bashrc for the Virtual Environment**

For ease of switching to the virtual environment, which is where run testing for configuring the home assistant, I put the source command in my .bashrc of my homeassistant user to make it easier.

```
$ vi /home/homeassistant/.bashrc
```

Copy and paste the following at the bottom of the file

```
source /srv/homeassistant/homeassistant_venv/bin/activate
```

Now save the file and to test it type the following:

```
$ exit
$ sudo su -s /bin/bash homeassistant
```

Should see the following:

```
(homeassistant_venv) homeassistant@raspberrypi:/home/pi $
```

## **Step 3: Install the Home Assistant Program**

Now that you are in the home assistant virtual environment you will now install the program with the following commands:

```
(homeassistant_venv) homeassistant@raspberrypi:/home/pi $ cd /srv/homeassistant
```

```
(homeassistant_venv) homeassistant@raspberrypi:/srv/homeassistant/ $ pip3 install homeassistant
```

This will install the program. Be patient for it will take a while depending on the version of raspberry pi and the speed of the microSD card used in this system. Once finished you can manually run the program from the virtual environment by typing the following command:

```
(homeassistant_venv) homeassistant@raspberrypi:/home/pi $ hass
```

However, next step we will look at making it start when the pi starts up.

#### **Step 4: Setup Home Assistant to Auto Start**

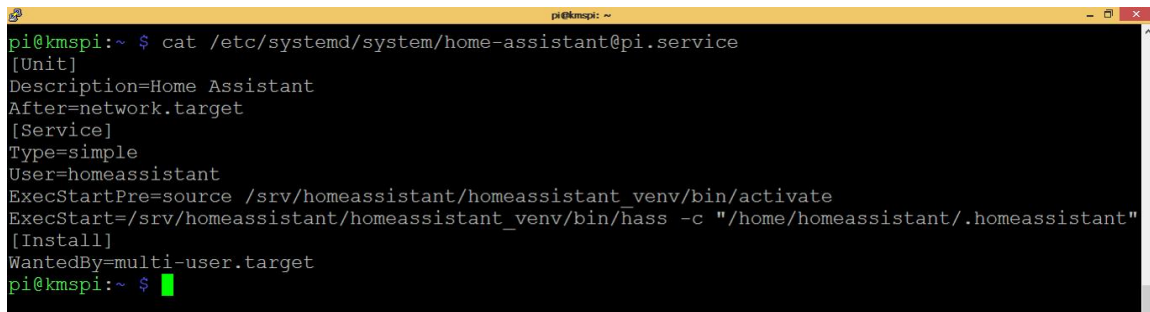
Now we will need to setup the home assistant program to auto start on boot through the systemctl. This information is taken from the following address: <https://www.home-assistant.io/docs/autostart/systemd/>

```
$ sudo su root
$ cd /etc/systemd/system/
$ vi home-assistant@pi.service
```

Now cut and paste the following:

```
#####
[Unit]
Description=Home Assistant
After=network.target
[Service]
Type=simple
User=homeassistant
ExecStartPre=source /srv/homeassistant/homeassistant_venv/bin/activate
ExecStart=/srv/homeassistant/homeassistant_venv/bin/hass -c
"/home/homeassistant/.homeassistant"
[Install]
WantedBy=multi-user.target
#####
```

Save this and exit out of editing the file and exit root to return to the pi user. This auto start file(*home-assistant@pi.service*) must be same as describe in Figure 4.11.



```
pi@kmspi: ~ $ cat /etc/systemd/system/home-assistant@pi.service
[Unit]
Description=Home Assistant
After=network.target
[Service]
Type=simple
User=homeassistant
ExecStartPre=source /srv/homeassistant/homeassistant_venv/bin/activate
ExecStart=/srv/homeassistant/homeassistant_venv/bin/hass -c "/home/homeassistant/.homeassistant"
[Install]
WantedBy=multi-user.target
pi@kmspi: ~ $ █
```

**Figure 4.11 Text Output of *home-assistant@pi.service* File**

Now you will need to restart the systemctl and read the file with the following commands

```
$ sudo systemctl --system daemon-reload
$ sudo systemctl enable home-assistant@pi
$ sudo systemctl start home-assistant@pi
```

Now you should be able to start the service with the following command:

```
$ sudo systemctl start home-assistant@pi
```

You can view the log to see if it is starting properly with the command:

```
$ sudo systemctl status home-assistant@pi -l
```

Or if you would like to view a scrolling log you can issue the following command:

```
$ sudo journalctl -f -u home-assistant@pi
```

### **Step 5: Configuration File Setup GPIO**

Now we have gotten it started we need to configure it for the gpio.

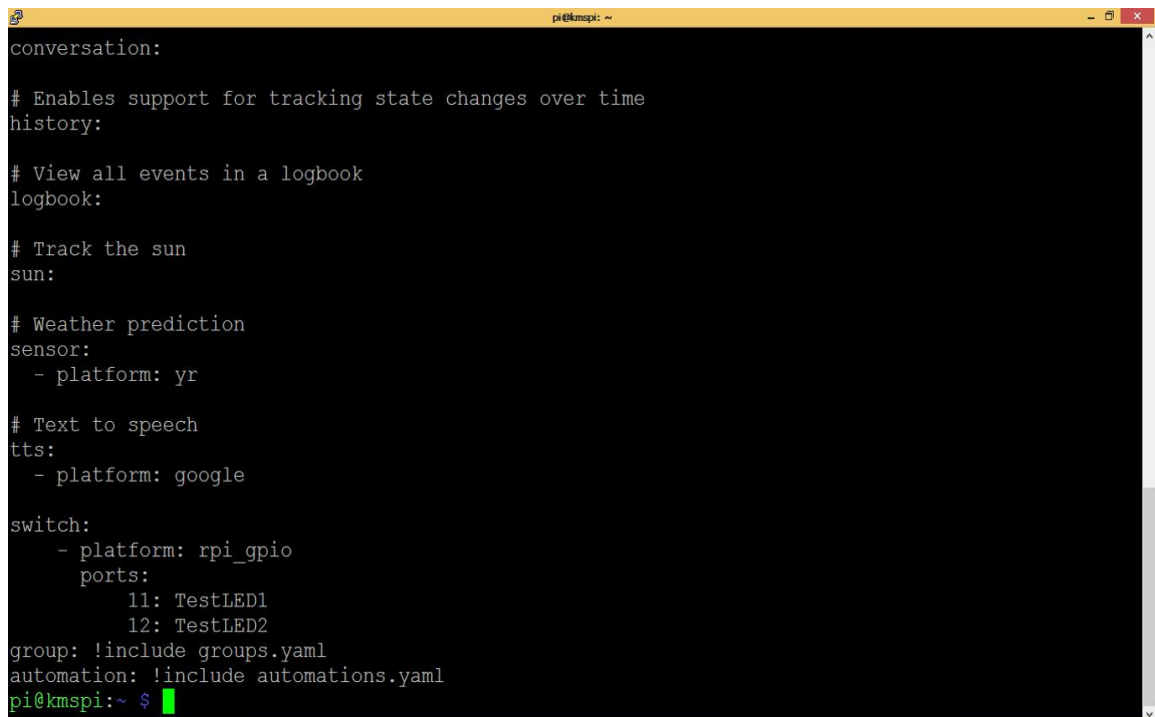
As the user pi you need to open the configuration file as follows:

```
$ cd /home/homeassistant/.homeassistant
$ vi configuration.yaml
```

Once in the configuration file you will disable the introduction component by commenting out the "introduction" line and Now you will add the following to the file to activate the GPIO for the raspberry pi like Figure 4.12. (in this example I am using GPIO 11 and 12 for this)

```
#####  
# Show links to resources in log and frontend  
#introduction:  
switch:  
- platform: rpi_gpio  
ports:  
11: TestLED1  
12: TestLED2  
#####
```

Save this to the configuration file.



```
conversation:  
# Enables support for tracking state changes over time  
history:  
# View all events in a logbook  
logbook:  
# Track the sun  
sun:  
# Weather prediction  
sensor:  
- platform: yr  
# Text to speech  
tts:  
- platform: google  
switch:  
- platform: rpi_gpio  
  ports:  
    11: TestLED1  
    12: TestLED2  
group: !include groups.yaml  
automation: !include automations.yaml  
pi@kmspi:~ $ █
```

Figure 4.12 Output of configuration.yaml File

## Step 6: Testing Your Changes and Restart Home Assistant

Now need to test the changes.

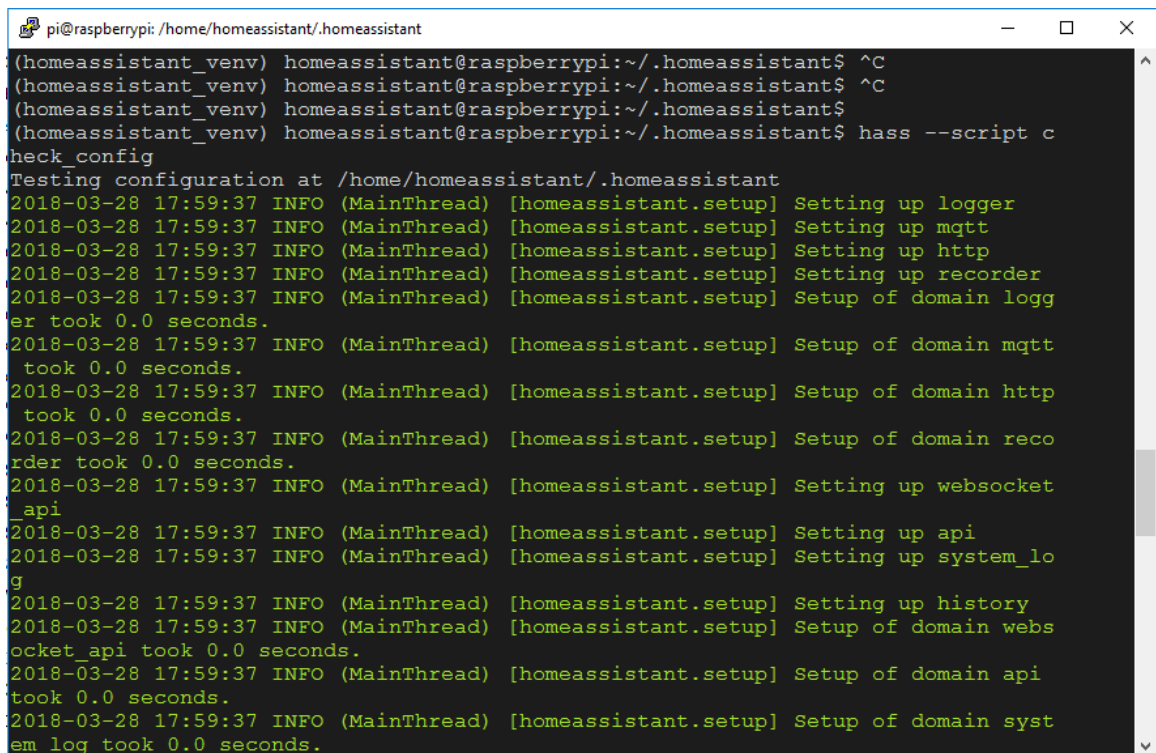
To test for errors in configuration need to run the `check_conFigure` command as follows:

from pi user:

```
$ sudo su -s /bin/bash homeassistant
```

```
(homeassistant_venv) homeassistant@raspberrypi:/home/pi $ hass --script  
check_config
```

If all goes well you should get no errors like Figure 4.13.



```
pi@raspberrypi: /home/homeassistant/.homeassistant
(homeassistant_venv) homeassistant@raspberrypi:~/homeassistant$ ^C
(homeassistant_venv) homeassistant@raspberrypi:~/homeassistant$ ^C
(homeassistant_venv) homeassistant@raspberrypi:~/homeassistant$
(homeassistant_venv) homeassistant@raspberrypi:~/homeassistant$ hass --script c
check_config
Testing configuration at /home/homeassistant/.homeassistant
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setting up logger
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setting up mqtt
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setting up http
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setting up recorder
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setup of domain logg
er took 0.0 seconds.
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setup of domain mqtt
took 0.0 seconds.
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setup of domain http
took 0.0 seconds.
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setup of domain reco
rder took 0.0 seconds.
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setting up websocket
_api
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setting up api
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setting up system_lo
g
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setting up history
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setup of domain webs
ocket_api took 0.0 seconds.
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setup of domain api
took 0.0 seconds.
2018-03-28 17:59:37 INFO (MainThread) [homeassistant.setup] Setup of domain syst
em_log took 0.0 seconds.
```

Figure 4.13 Result of checking conFigure files with “`hass --script check_config`”

To restart with `systemctl` type the following command as the pi user:

```
$ sudo systemctl stop home-assistant@pi
```



```
$ sudo systemctl start home-assistant@pi
```

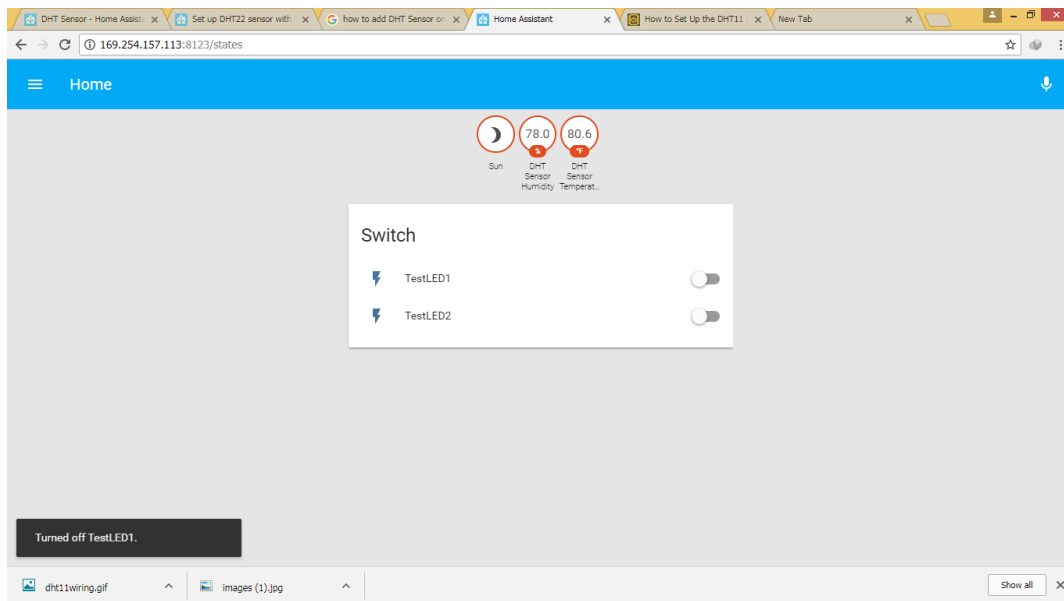
Remember you can watch the start up with the following two commands:

```
$ sudo systemctl status home-assistant@pi -l
```

Or

```
$ sudo journalctl -f -u home_assistant@pi
```

Once restarted go to home assistant web page <http://192.168.21.50:8123>. Now notice two LED's are now accessible as describe in Figure 4.14.



**Figure 4.14 Result of Testing Home Assistant Webpage**

All the complete configuration file of the home assistant that connect with MQTT client, IFTTT, Remot3.it and sensors are store in the “.yaml” files. In Figure 4.15 can see the list of home assistant conFigure files for my system. And the complete Web-UI for my system is describe in Figure 4.16.

```

pi@raspberrypi: /home/homeassistant/.homeassistant
pi@raspberrypi: /home/homeassistant/.homeassistant $ ls -al
total 75004
drwxr-xr-x 6 homeassistant homeassistant 4096 Mar 28 18:02 .
drwxr-xr-x 4 homeassistant homeassistant 4096 Nov 6 15:49 ..
drwxrwxrwx 2 root root 4096 Feb 9 12:09 automation
-rwxrwxrwx 1 homeassistant homeassistant 735 Feb 9 12:29 automations.yaml
drwxrwxrwx 3 root root 4096 Feb 8 00:16 autorun.inf
-rwxrwxrwx 1 root root 544 Jan 7 20:34 binary_sensors.yaml
-rwxrwxrwx 1 homeassistant homeassistant 3319 Feb 6 00:05 configuration.yaml
-rwxrwxrwx 1 root root 4578 Dec 24 23:55 configuration.yaml.bak
-rwxrwxrwx 1 homeassistant homeassistant 1074 Feb 9 08:32 customize.yaml
drwxr-xr-x 2 homeassistant homeassistant 4096 Nov 4 15:51 deps
-rwxrwxrwx 1 root root 3736 Feb 5 15:45 door_back_node2.ino
-rwxrwxrwx 1 homeassistant homeassistant 1177 Feb 9 11:48 groups.yaml
-rwxrwxrwx 1 homeassistant homeassistant 6 Dec 24 22:40 .HA_VERSION
-rw-r--r-- 1 homeassistant homeassistant 88823 Mar 28 18:02 home-assistant.log
-rw-r--r-- 1 homeassistant homeassistant 76617728 Mar 28 18:02 home-assistant_v2.db
-rwxrwxrwx 1 homeassistant homeassistant 120 Dec 24 20:45 IFTTT.yaml
-rw-r--r-- 1 homeassistant homeassistant 0 Nov 4 15:51 scripts.yaml
-rwxrwxrwx 1 homeassistant homeassistant 284 Dec 25 10:53 secrets.yaml
-rwxrwxrwx 1 root root 1254 Feb 8 13:52 sensors.yaml
-rwxrwxrwx 1 root root 726 Feb 9 07:57 switches.yaml
drwxr-xr-x 2 homeassistant homeassistant 4096 Nov 4 15:54 tts
-rwxrwxrwx 1 homeassistant homeassistant 44 Nov 4 15:53 .uuid
pi@raspberrypi: /home/homeassistant/.homeassistant $

```

Figure 4.15 List of Home Assistant Configuration Files for the System

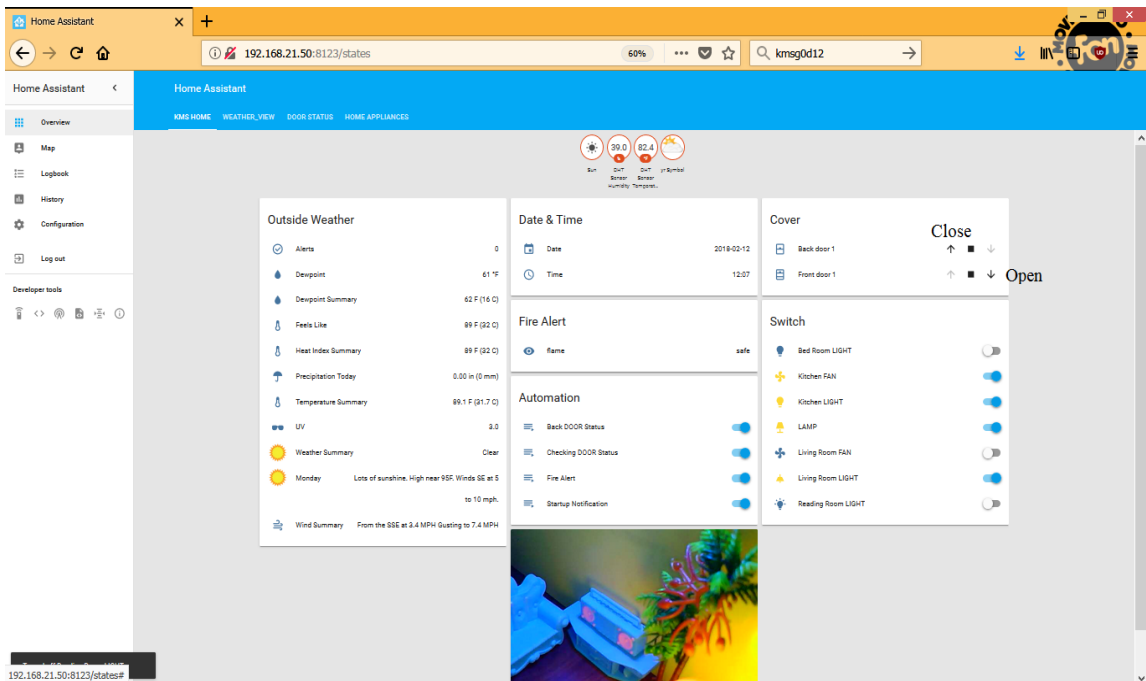
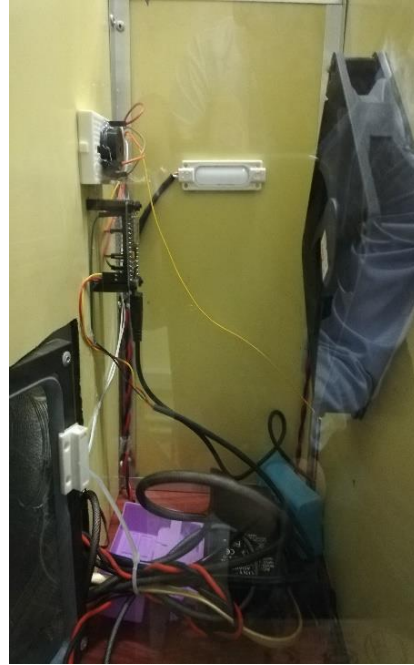


Figure 4.16 Complete Web-UI of the Whole System

A photo of hardware Implementation of the Front Door Part, Back Door Part and Main Part of the system are as shown in Figure 4.17. The complete prototype of the whole system is described in Figure 4.18.



**(a) Front Door Part**



**(b) Back Door Part**



**(c) Main Part**

**Figure 4.17 Hardware Implementation of the System**



**Figure 4.18 Prototype of the System**

## **4.7 Software Implementation of the System**

The following flow chart of the system shows the complete of the IoT-based Home Control and Monitoring System using computer control and MQTT protocol. The required mode is initialized according to the appropriated process. Firstly, board from the main part (RPi) and two door parts (NodeMCUs) are started and connected to the Wi-Fi. From the main part side, starting the MQTT mosquitto server, connecting with Cloud services (Weather Underground, Remot3.it, IFTTT). From the two door parts, collect the data from the sensors (magnetic and flame sensors) and these NodeMCUs are connecting to the MQTT server as clients to send the collected data to the server. To support the multi-ways to open (unlock) the door lock, RFID sensor and Keypad are installed in the front door part.

The fire alarm system is installed in the back door part to alert the accident of fire at home [9][10].

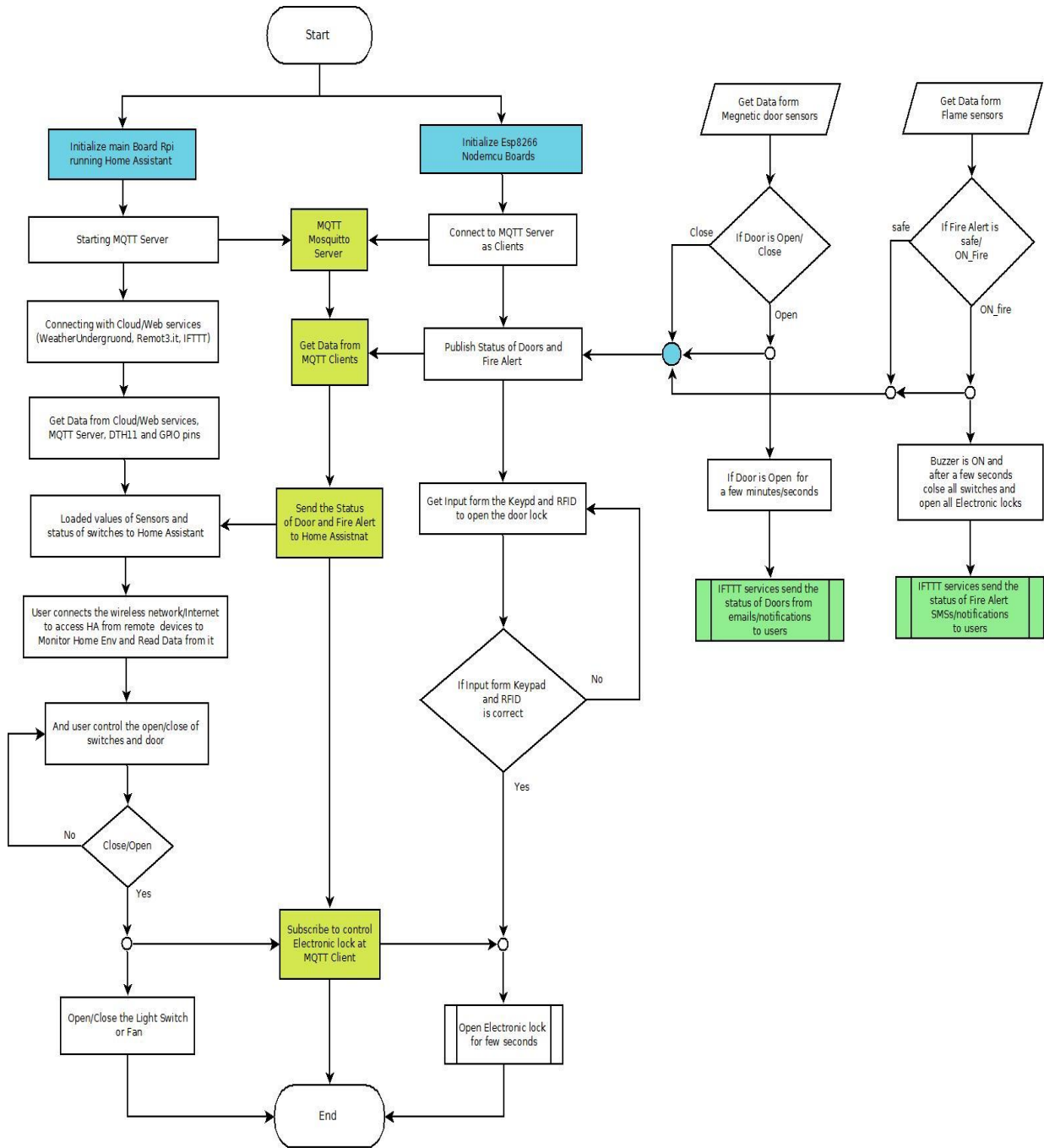
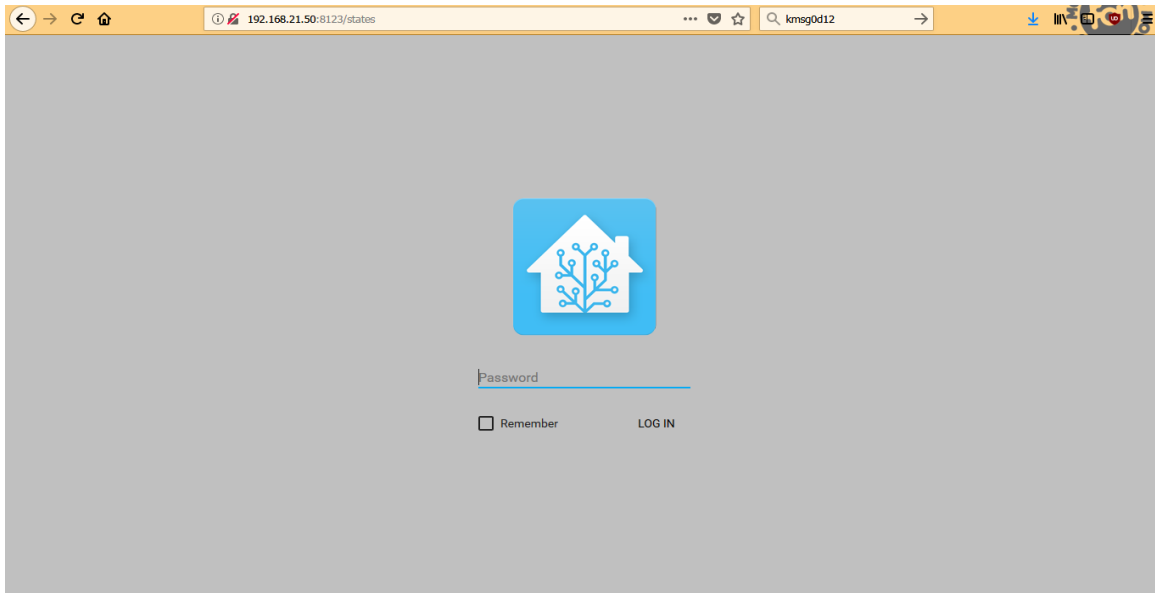


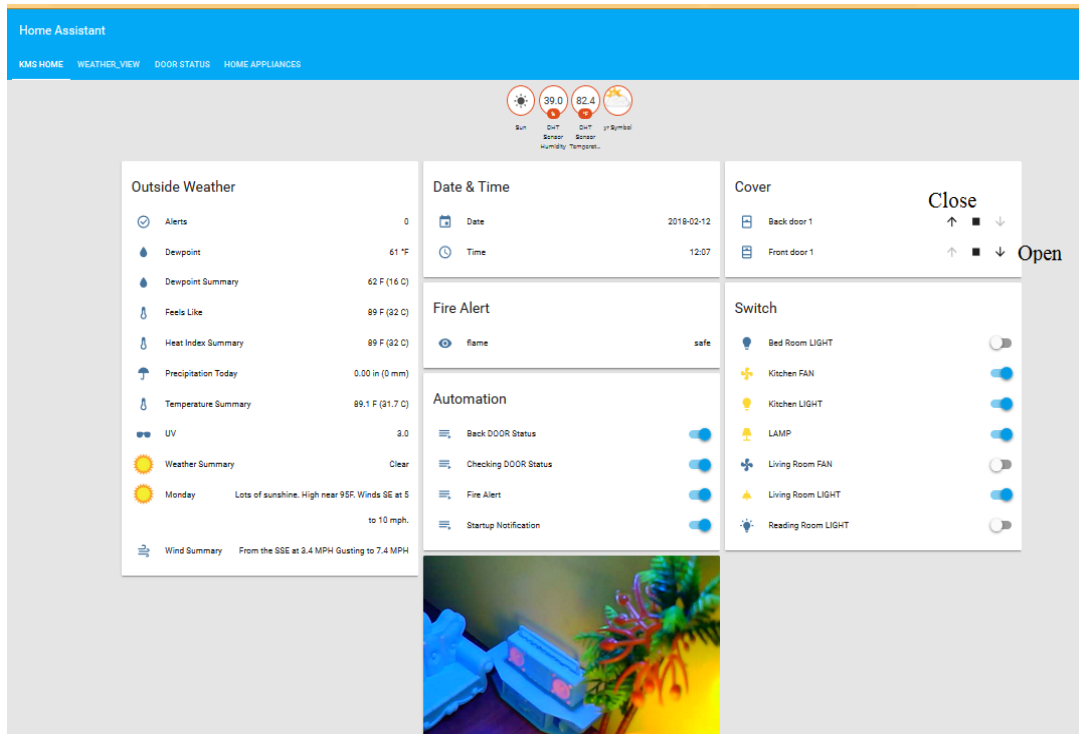
Figure 4.19 Flow Chart of the System

## 4.8 Operation of the System

The sensors are distributed in different parts of the house, from these sensors sending real time continuous data to the main central board RPi wirely or wirelessly. A set of functions continuously executes depending on the condition of the statement. These functions include operations like automatic switching on/off of certain selected lights, performing requested actions like switching on/off of a home appliance, performing emergency actions like notifying the user and sounding the alarm. A Web User Interface (WUI) is being created where many home appliances and switches can be controlled and monitored. And also implemented for the user can easily to control and continuous monitoring of the home environment using end devices. If any fire accident is occurred at home, user can check the situation of home environment through webcam, simultaneously sending alert notifications and messages to user is implemented [11].



(a)



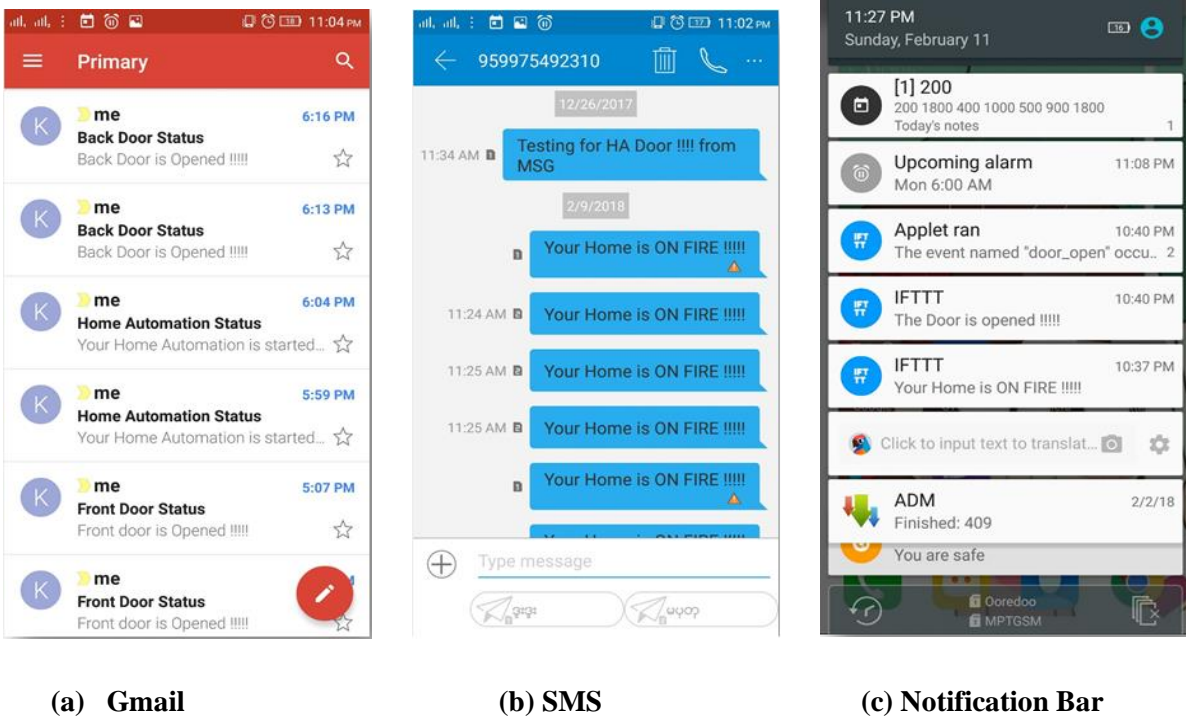
(b)

**Figure 4.20 Login Page and Complete WUI of Home Assistant**

Single WUI is designed for us to monitor and control home very easily. If there is any emergency case is happened we can control and monitor the situation from any remote palaces, that service is implemented using cloud service. And the WUI login page is configured by using password authentication method. In Figure shows the login page of this home control and monitoring system. Anyone can login into this system with the correct password and can see various information of the system collected in one place. We can check all the status of all appliances at home whether that appliances are turned OFF/ON and continuous monitoring of our home environment. We can also view all the room in home with various appliances, we can choose any devices and door locks which we want to control.

The above Figure 20 (b) shows Outside Weather Information, Living Room Temperature, Fire Alert, Date & Time, Home Monitoring Output and controllable Switches, Door Locks and Automations are described in one Web User Interface (WUI). Through this WUI we can easily turn ON/OFF any appliances that we want. Sensors, the controllable devices and switches are directly connected with GPIO (General Purpose Input

Output) pins of server (RPI) and client node (NodeMCU). Server and client nodes are communicated each other to exchange information using MQTT protocol. As a result, we can monitor the status of all the devices by mean of end devices through WUI.



**Figure 4.21 Notifications receive from user side**

One of the great benefits of the system is that the devices and appliances at home can be controlled and monitored from any end devices that can access web-service. And user also get notification with real-time information about home when sensor is detected fire or someone forget to close the door. Figure 4.21 describe the notifications (Gmail, SMS, Notification Bar) received by user. If house is really on fire system will automatically turn-off switches (shutdown electricity) and unlock all door lock at home. The control of various home appliances, door lock and monitor of the devices can be done by using end devices from Home Assistant UI through network (Internet).



## 4.9 Comparison from other Home Automation System

The benefits of home automation can be a separate into few categories, including savings, control, safety and convenience. To support all of the above benefits in home automation system will cost a lot. But my system will give consumers all of above benefits with very low cost. The main building block of Home Automation System consisted of four basic elements: communication, the sensor unit, processing unit and power units. Some automation system sensor unit is running with Arduino UNO and ESP8266 module to sense data through connected sensor, sometime used Raspberry Pi as a sensor node just collecting sensor data and send to the main server. And the processing unit is running with high end PC to communicate with sensor units connected wired or wirelessly. In my system sensor unit is running only with NodeMCU to reduce cost and complexity of hardware implementation because it contains built-in ESP8266 WiFi module. The processing unit in my system is operating with Raspberry Pi because it is cheap, flexible, fully customizable, low power and programmable small computer board with built-in GPIO pins. Also, very powerful and easy to implement with many sensors and Compatible Boards. For end user control part other systems are platform dependency, the control and monitoring home only work on Android or IOS App. One advantage of my system is end user can easily control and monitor through **browser** that install on any device and platform like Android, IOS, Linux, Mac and etc. And my system can easily be extended by adding many sensor units running with NodeMCU [12][13]. Table 4.1 shows Comparison of Home Automation Systems.

**Table 4.1 Comparison with other Home Automation System**

| Serial no. | System  | Controller | Communication on Interface | User Interface                  | Applications  | Merits                                |
|------------|---|------------|----------------------------|---------------------------------|---|---------------------------------------|
| 1          | Wi-Fi based using Arduino microcontroller through IOT | Arduino    | Wi-Fi                      | Web Application and android App | Temperature and motion detection, monitoring and controlling appliances | Low cost, Secure, Remotely controlled |

|   |   |   |  |                         |  |  |
|---|---|---|--|-------------------------|--|--|
| 2 | Smart Task Scheduling Based using Arduino and Android | Arduino   | Wired X10 and Wireless Zig bee                 | Android Application     | Energy Management and task scheduling with power and cost                            | Energy-efficient and Highly scalable   |
| 3 | Web service and android app Based using Raspberry pi  | Raspberry pi                                      | Web server and interface card                  | Android application     | Controlling shutter of window  | Autonomous, and Quite scalable   |
| 4 | Cloud Based Using Hadoop System                       | Home gateway and router                           | Cloud based data server uses Hadoop technology | Smart device            | Monitoring and Controlling Home Appliances   | Effectively manage Semi structured and unstructured data, Reduce computational burden of smart devices |
| 5 | Android based using Arduino                           | Arduino Mega 2560 and the Arduino Ethernet shield | Micro Web Server                               | Android App             | Light switches, Temperature, Humidity sensors, Intrusion detection, Smoke/Gas sensor | Feasibility and Effectiveness  |
| 6 | Konnex-Bus based using raspberry pi                   | Raspberry pi and Konnex Bus                       | SIP Provider                                   | Moblie App              | Lights Control, Temperature Monitoring   | Performance improved, energy consumption could be Reduced  |
| 7 | Bluetooth Based using Arduino                         | Arduino   | Bluetooth                                      | Python supported mobile | controlling  | Secured and Low cost   |
| 8 | GSM Based Using Arduino                               | Arduino   | SMS  | Smartphone App          | Control appliances   | Simplicity   |

|   |  |                           |  |  |   |  |
|---|--|---------------------------|--|--|---|--|
| 9 | IoT-based Home Control and Monitoring System using Raspberry Pi and NodeMCUs | Raspberry pi and NodeMCUs | Wireless Network, MQTT Protocol, Cloud services and Local server | Browser of any end-devices running on Android, IOS, Window, Linux etc. | Home appliances control, Live Streaming Monitoring of home environment, Notify the status of home to user through SMS and Email | Very low cost and power consumption, Can extend many sensor units, Automatic response function depend on the condition |
|---|--|---------------------------|--|--|---|--|

#### 4.10 Evaluation of the System

In this system, the main sever part of the system is running with Raspberry Pi and two client's parts are running with NodeMCUs. These parts are connected to the WiFi network that support by the sim router. MQTT server is running on Raspberry Pi and support communication between server and client's parts wirelessly. The major parts are not connected each other wirely. The client's node is easily adding to the system to monitor and control our home environment so this system is very good at the design point of view. And the user interface of this system is very easy to access and control using any networked devices that can use bowser. One of the special feature of this system is notifying user about the status of home real time getting data got form sensor. With the help of IFTTT cloud service sending the alert notification to user using Gmail, SMS when user is forgot to close the doors and the house is facing any fire accident. When the house is on fire enable the buzzer for a few seconds and switch off all the appliances at home and open all door locks. In this system user can also live streaming monitoring of home environment through webcam any time anywhere. Compare from the other commercial home automation system this system is very cost effective and can do almost the same features like high cost commercial one. In Table 4.2 features comparison of this system with other commercial Home Automation systems.

**Table 4.2 Comparison of the Features of Commercial Home Automation System**

| <b>Solution</b>        | <b>HomeSeer</b>                                   | <b>Qivicon</b> | <b>Domintell</b> | <b>IoT-based Home Control and Monitoring System using Raspberry Pi and NodeMCUs</b> |
|------------------------|---|----------------|------------------|---|
| Protocols              | Insteon, UPB, Wi-Fi, X10, PLC-BUS, Modbus, Z-Wave | Wi-Fi, ZigBee  | S-Bus            | Wi-Fi, MQTT, Cloud service  |
| Transmission           | Wired and wireless                                | Wireless       | Wired            | Wireless  |
| Locking system         | Yes   | Yes            | Yes              | Yes   |
| Temperature            | Yes   | Yes            | Yes              | Yes   |
| Media center           | Yes   | Yes            | Yes              | No  |
| Lighting               | Yes   | Yes            | Yes              | Yes   |
| Environmental control  | Yes   | Yes            | Yes              | Yes   |
| Video surveillance     | Yes   | Yes            | No               | Yes   |
| User experience        | Acceptable  | Acceptable     | Good             | Acceptable  |
| Variety of peripherals | High  | Very high      | Medium           | High  |
| Technical security     | Yes   | Yes            | Yes              | Yes   |
| (2018) Price \$        | 1000 – 1200                                       | 1300           | 900              | 300   |

## CHAPTER 5

# CONCLUSION AND FURTHER EXTENSION

### 5.1 Conclusion

In this system, the circuit design and prototype structure of the IoT-based Home control and monitoring system has been successfully constructed and experimented with many end-devices to control. This system is implemented by using Raspberry Pi (RPI), NodeMCUs, Magnetic Door sensors, Temperature and humidity sensor (DTH11), IR Flame sensor, Buzzer, Keypad, RFID Card Sensor module, Solenoid electronic lock, Webcam, Relays, Fans and Power supply. And the system runs with three main parts: Front Door, Back Door and Main Server parts. Front Door part supports to control the door lock, sensing the status of door and also provides multi-ways to control door lock from Keypad and RFID module. From Back Door part view it supports to control door lock and additionally includes fire alarm function. If users want to control and monitor the devices and appliances at home, they have to get access to the main server part from local network or Internet. All main parts run separately to do their tasks, these parts are connected to Wireless Access Point and they communicate each other wirelessly through MQTT Protocol to exchange information.

The focus is on reliable delivery of data from sensors to central main board using broker-based publish/subscribe messaging protocol MQTT, which main characteristics are low overhead, asynchronous communication, low complexity and low power. The system is very suitable for monitoring real-time situation of home environment and remotely control all the appliances connected to system using any end devices that can accept WUI. There is also simultaneously sending alert notifications and messages to user if anyone forgets to close door and detecting any fire accident. The system can be used in many places like schools, offices, shops, departments etc. and anywhere that users want to control and monitor of their environments.

## **5.2 Advantages of the System**

The advantages of the system presented in this thesis are as follows:

- Being very low cost and low power consumption
- Using Computer Control theory concept and Wireless Communication Technology in this IoT-based Home control and monitoring system
- Using Web User Interface (browser running on any end-devices) to control and monitor the entire system easily
- Being able to turn on or off switches that connected with home appliances and unlock the doors lock using end-devices
- Supporting with Keypad and RFID module as multi-ways access to the door lock for the condition of broken user's end-devices
- Capability of automatically cutting off the power of house when the fire accident occurs at home
- Reporting the abnormal condition of house to user through SMS and Email
- Being able to control the appliances at home and monitor the home environment through Webcam by using Internet
- Being extendible system by adding new sensor nodes running with NodeMCU

## **5.3 Limitations of the System**

In this thesis, the backbone of the system is the Wireless Network that support Internet because all of the parts of the system are connected to that Wireless Network. If the Internet Connection loses from it, the system will go offline and users cannot access the system from any places. This home control and monitoring system is just small size prototype of house. Instead of using low voltage DC output lights and fans, the higher voltage DC or AC lights, fans and real home appliances can be used to control at home where we live.

## **5.4 Further Extension**

IoT is having tremendous attention recently and its various applications are growing, changing the way we live and work. As there are various appliances that can be controlled and automated while being away from home, the same approach implemented for this system will be used to enable control over various appliances before reaching home, saving time and effort. Such application examples can be monitor temperature and humidity, and being able to control them, controlling stoves and microwaves, locking and unlocking doors autonomously depending on the visitor face recognition, controlling all the home appliances and devices very easily using voice commands, monitor and manage the running large Industrial automation system through Internet, combine the automation system AI. This system can reduce the weak points of security problems in restricted areas and combination of IoT with many other devices will be used as smart appliances autonomy.

## REFERENCES

- [1] Al-Ali A. R., Rousan M. A., Mohandes M., “GSMbased Wireless Home Appliances Monitoring & Control System”, IEEE International Conference, ISBN: 0-7803- 8482-2, (2004) 237-238
- [2] MQTT-S A publish/subscribe protocol for Wireless Sensor Networks, Hunkeler, U.; IBM Zurich Res. Lab., Zurich; Hong Linh Truong; Stanford-Clark, A, Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference
- [3] “Cloud based low-cost Home Monitoring and Automation System”, Shruthi Raghavan and Girma S.Tewolde, 2015 ASEE North Central Section Conference
- [4] “IoT-based Smart Security and Home Automation System” (2016), Ravi Kishore Kodali, Vishal Jain, Suvadeep Bose and Lakshmi Boppana, International Conference on Computing, Communication and Automation (ICCCA2016)
- [5] “A Low Cost Smart Security and Home Automation System Employing an Embedded Server and a Wireless Sensor Network”, Semanur Karaca, Dr. Alper ŞİŞMAN, İbrahim SAVRUK, 2016 International Conference on Consumer Electronics.
- [6] Eclipse Mosquitto. (2016, Feb. 12). [Online]. Available: <http://mosquitto.org/-Commun> vol. 4, no. 11, pp. 13121324, Jul. 2011.
- [7] K. Tang et al., “Design and implementation of push notification system based on the MQTT protocol,” in International Conference on Information Science and Computer Applications, September 2013.
- [8] S.K. Shriramoju, J. Madiraju, and A.R. Babu, “An approach towards publish/subscribe system for wireless networks,” International Journal of Computer and Electronics Research, vol. 2.,pp. 505-508, August 2013.
- [9] Raspberry Pi as a Wireless Sensor Node: Performances and Constraints, Vladimir Vujovic and Mirjana Maksimovic, Faculty of Electrical Engineering, East Sarajevo, Bosnia and Herzegovina, IEEE 2014
- [10] “Wireless Home Security and Automation System Utilizing ZigBee based Multi-hop Communication”, Mohd Adib B. Sarijari, Rozeha A. Rashid, Mohd Rozaini Abd Rahim, Nur Hija Mahalin, IEEE 2008 6th National Conference on Telecommunication Technologies.
- [11] Ahmed El Shafee and Karim Alaa Hamed (2012), “Design and Implementation of a Wi-Fi Based Home Automation System”, World Academy of Science, Engineering and Technology, Vol. 6.
- [12] Rajeev Piyare, Department of Information Electronics Engineering, Mokpo National University, Korea. “Internet of Things: Ubiquitous Home Control and Monitoring System using Android based Smart Phone”



[13] Hamid Hussain Hadwan(M.E. Student, Mech. Mechatronics, SCOE, Pune, India)  
, Y. P. Reddy (Professor in Mech., SCOE, Pune, India ). “Smart Home Control by  
using Raspberry Pi & Arduino UNO”

## **PUBLICATION**

[1] Kaung Nyunt San and Hlaing Thida Oo, “Implementation of IoT-based Home Control and Monitoring System using Raspberry Pi and NodeMCUs”, *Parallel and Soft Computing Journal*, University of Computer Studies, Yangon, Myanmar, 2018.