الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي

Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

**People's Democratic Republic of Algeria**
**Abu Bakr Belkaid University - Tlemcen**
**Faculty of Sciences**
**Computer Science department**

**End of studies thesis**
**For obtaining the Master's degree in Computer Science**
**Option: Network and Distributed Systems (R.S.D)**

*Theme*

# *Smart Home Control Using MQTT Protocol*

**Proposed & Submitted by:**

- BOUMERZAK Kheira
- BERRICHI Akram

**Presented on September 2021 by the following Board:**

| | | | |
|---|---|---|---|
| • **Mr.** BELHOCINE Amine | MAA | University of Tlemcen | President |
| • **Ms.** ABDELDJELIL Hanane | MCB | University of Tlemcen | Examiner |
| • **Mr.** ZIANI-CHERIF Salim | MCB | University of Tlemcen | Supervisor |

Academic year: 2020-2021

# Acknowledgment

*Our deep and sincere gratitude, appreciation and respect to one of the best professors we encountered during our academic years, to our wonderful supervisor*
***Mr. Ziani-Cherif Salim**,*

*Thank you for your valuable support and guidance.*

*We would like to extend our sincere gratitude and appreciation as well, to the jury for accepting to read, review and evaluate this work.*

*We would also like to give our thanks to all the employees of our university studies at the computer science department.*

# Dedication

*First of all, I thank* ***ALLAH THE ALMIGHTY WHO*** *gave me the courage, strength and ability to accomplish this work.*

*Then, I dedicate with all my heart this modest work:*

*To* ***my dear mother****, the sweetest person I ever know, thank you so much for raising me better, you are the source of my life and happiness.*

*To the dearest one in my heart, the one who always makes me feel happy and worth it, my best friend and everything in my life* ***my dear father****.*

*To the angelic person who represent my second family, who is like a father To me* ***Bz. Mohammed*** *I appreciate your support and love, I will always be grateful, Thank you*

*To my soulmates, the source of my strength in life, my incredible sisters, whom I love so much:* ***Meriam, Amal, Ikram, Chaymae, Yousra*** *and to my one and only, my great beloved brother* ***Mohammed,*** *am so grateful for everything you did for me so far.*

*In the memory of my dear and wonderful* ***grandfather,*** *for the great love he always gave, you were such an angel who carved a special place in my heart that will forever remain the same.*

*Lastely, to my dear friend* ***Chaouki****, the wonderful person you are, I deeply thank you for the big support, so many appreciations,*

*I love you all beautiful people.*

*Kheira BOUMERZAK*

# Dedication

*First of all, I thank* ***ALLAH THE ALMIGHTY WHO*** *gave me the courage, strength and ability to accomplish this work*

*Then, I dedicate with all my heart this modest work:*

*To whom I owe everything, my parents "Hassina" &"Mohamed" with love to my amazing brothers "Amine", "Reda" and my wonderful sister "Lina"*

*In the memory of my grandparents from both my mom and my dad side, uncle Sassi, uncle Ali, uncle Mouloud and cousin Sofiane*

*To my dear colleague Kheira BOUMERZAK*

*To AIESEC in Tlemcen, my second family*

*And very special gratitude is due to all those extraordinary people who have stood by me in very hard moments.*

*Akram BERRICHI*

**Abstract:** We all know that technology keeps advancing within the years to provide a comfortable environment in order for humans to live a better life. Speaking of which, the one among them that caught our attention was IoT (Internet of Things). Thinking of that, we developed a real interest in how Things will work. Therefore, we've chosen "Smart Home Control using Mqtt Protocol", to be our main project. Hence, To explore how things are connected with each other, from using MQTT protocol, to what software and what equipment were used in order to make things alive, we will explain it all since the main goal of this project is to provide security with innovation at home.

**Keywords:** Smart Home, Internet of Things, Home Appliances, Raspberry Pi, Nod-Red, Esp8266, Arduino, MQTT, Automation

**Résumé :** Nous savons tous que la technologie continue de progresser au fil des ans pour fournir un environnement confortable permettant aux humains de vivre une vie meilleure. En parlant de cela, celui d'entre eux qui a attiré notre attention était l'IoT (Internet des objets). En pensant à cela, nous avons développé un réel intérêt pour la façon dont les choses fonctionnent. Par conséquent, nous avons choisi "Domotique intelligente basée sur le Protocol Mqtt" comme projet principal. Par suite, pour explorer comment les choses sont connectées les unes aux autres, de l'utilisation du protocole MQTT, à quels logiciels et quels équipements ont été utilisés pour rendre les choses vivantes, nous expliquerons tout car l'objectif principal de ce projet est d'assurer la sécurité avec l'innovation à la maison.

**Mots clés :** Maison intelligente, Internet des objets, Electroménager, Raspberry Pi, Nod-Red, Esp8266, Arduino, MQTT

**ملخص :** نعلم جميعًا أن التكنولوجيا تستمر في التقدم مع مرور السنوات بهدف توفير بيئة مريحة من أجل أن يعيش البشر حياة أفضل. بالحديث عن ذلك ، الذي لفت انتباهنا، كان إنترنت الأشياء و عند التفكير في ذلك، طورنا اهتمامًا حقيقيًا عن كيفية عمل الأشياء. لذلك، اخترنا أتمام "المنزل الذكي المبني على أساس البروتوكول Mqtt" ، ليكون مشروعنا الرئيسي. مع استكشاف كيفية ارتباط الأشياء ببعضها البعض ، من استخدام بروتوكول Mqtt ، إلى البرامج و المعدات المستخدمة من أجل جعل الأشياء حية، سنشرح كل ذلك بدقة اذ ان الهدف الرئيسي لهذا المشروع هو توفير الأمن و الراحة في بيت مفعم بالأجهزة الذكية.

**الكلمات المفتاحية:** المنزل الذكي, إنترنت الأشياء, الأجهزة المنزلية, اردوينو

# Contents Table

# List of Figures

# List of tables

# General Introduction

Our future lies in our hands, so why not make it better and more appropriate?
In this computer era, everyone has a very busy life. Therefore, one should consider living in a comfort zone rather than being stressed. So, why not benefit from technological advancements?

Home Automation is the leading approach to this goal, which has become and will continue to become a trend in the upcoming century. It's a term used to describe how all of our domestic amenities and appliances operate together and how we can control them with ease from our smartphones, tablets, or computers having Internet connectivity; because they are the means from which one can use the Internet at any time and from anywhere. As a result, it opens a door that connects us to various home appliances that we use on a daily basis, allowing anyone to control their own home.

Therefore, we decided to turn this intriguing idea concept into a real-world project, in order to experience how it really works.

For a better understanding of our project and its content, here is a brief description of the chapters.

In the first chapter, we will present a general outline of the Internet of Things, Smart Homes, their architecture, the advantages of employing them, and problems they solve. In addition, their state inside & outside of Algeria.

In the second chapter, we will discuss the various protocols that can be used to manage and monitor the home, as well as a comparison of why some protocols are better than others. Then go over MQTT in depth, as it's the preferred protocol we've chosen as our main protocol and one that plays a major role in appliance communication. We'll go through how it's built and why it's the ideal option for use in.

In the third chapter, we shed the light on the different tools of hardware and software that were used, how they mash up together, their functionalities and features. Thus, the connection between them.

In the last chapter, we will discuss how we came to achieve our aim of building a smart home that we can control from afar, as well as how we programmed it.

# CHAPTER I: SMART HOMES

# CHAPTER I: SMART HOMES

## I.1. Introduction

In recent decades, home automation systems have grown in popularity as they improve comfort and quality of life. With the integration of IoT (Internet of Things) services by embedding intelligence into sensors and actuators, then networking the smart things to facilitate interaction between them. Doing so will result in increasing computational power, storage space and improving data exchange efficiency. Therefore, it becomes easier to manage multiple operations/objects since they are all interconnected. In fact, IoT household gadgets help save money, energy, not to mention time as well.

Such systems depend on the collection of data. The data is then used for monitoring, controlling, processing & transferring information to other devices via the internet. This allows specific actions to be automatically activated whenever certain situations arise. In a simple example, consider a smart kettle. The kettle can be programmed to automatically turn off once it reaches a specific temperature. Then send a notification to the user's smartphone.

Now apply the same concept to the entire home and all the devices present. That is a smart home powered by IoT. Instead of manually going up to the device and taking action, those actions can be taken at the press of a button. These days, most smart IoT home automation devices allow you to control them via an app or even via voice commands.

Now imagine if you did not even need to undertake such actions. In other words, the smart home will know when to take certain actions and automatically take them. This is what our project will try to convey.

## I.2. Domestic Technology Today

Nowadays most homes include small domestic appliances that cover the most difficult tasks and many other devices were designed to make home living more enjoyable. However, most devices are in need of some manual orders and close range interaction. These manual orders are no longer compatible with the fast pace of technology development. Therefore, many embedded systems developers started working on projects called "IoT Smart Homes" in order to make home life much easier, more fun and extra secure.

## I.3. Internet of Things

The IoT can be used to improve every aspect of life, bringing into existence many scenarios, such as smart homes, self-driving cars, smart cities, smart agriculture and much more.
What options do home sensors and gadgets have to communicate with one another? What's more, how do you make a house smart? Don't worry. That is exactly what our initiative is about.

### I.3.1. Definition

The Internet of Things, or IoT for short, was initially mentioned in 1999, in a speech given by the British engineer Kevin ASHTON. It was used to describe a system that connects physical objects to the Internet. They are also data-creating and data-transmitting systems that provide value to their users through various services (data analytics, machine learning…etc.) [19].

It refers to the interconnection of devices over internet, which allows them to send and receive data without human intervention. Smart homes are an important aspect of the Internet of Things, as they allow internet-connected appliances and devices to be managed automatically and frequently via a smartphone.

### I.3.2. IoT in Smart Homes

IoT systems are made up of numerous IoT system building blocks that work together to ensure that sensor-generated device data is gathered, saved, and processed in a big data warehouse, and that device actuators carry out orders received through a user application. The figure1 below depicts the approach to IoT architecture [20].

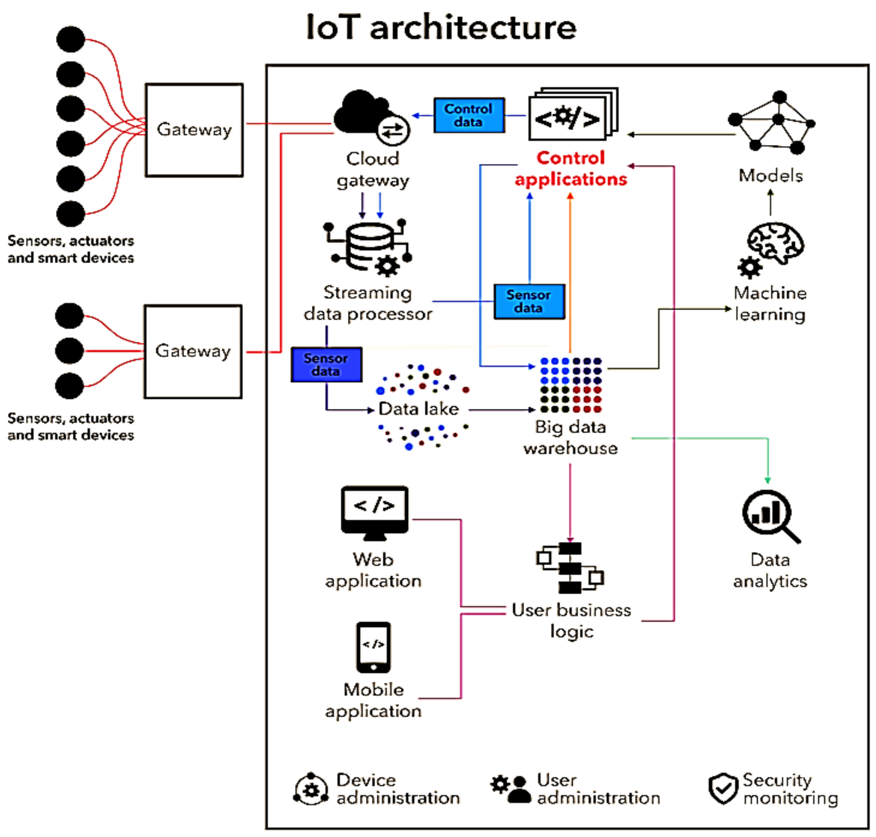


Figure I.1 IoT Architecture [20]

### I.3.3. Components of IoT Architecture

In simple terms, the IoT architecture explains the following elements [20]:

- **Things** equipped with **sensors** to gather data and **actuators** to perform commands received from the cloud.
- **Gateways** for data filtering, preprocessing, and moving it to the cloud and vice versa, receiving commands from the cloud.
- **Cloud gateways** to ensure data transition between field gateways and central IoT servers.
- **Streaming data processors** to distribute the data coming from sensors among relevant IoT solution’s components.

- **Data Lake** for storing all the data of defined and undefined value.
- **Big data warehouse** for collecting valuable data.
- **Control applications** to send commands to actuators.
- **Machine learning** to generate the models which are then used by control applications.
- **User applications** to enable users to monitor and control their connected things.
- **Data analytics** for manual data processing.

When developing an IoT architecture of a particular solution, it’s also important to focus on two important elements:

- **Consistency**: paying enough attention to every element of the IoT architecture and making them work together.
- **Flexibility**: the ability to add new functions and new logic.

## I.4. Smart Homes

### I.4.1. Definition

Smart Homes, also known as automated homes, intelligent buildings, integrated home systems, or domotics, are a recent design trend that uses internet- connected gadgets to provide remote monitoring and management of appliances and systems such as lighting and heating[21].

However, smart home technology has progressed to the point where almost any electrical component in the house can now be integrated into this intelligent automation system. Furthermore, rather than simply turning on and off equipment, smart home technology can monitor the indoor environment and the activities that occur when the house is occupied. As a result of these technological advancements, a smart home can now monitor its occupants' actions as well as operate gadgets independently in predetermined patterns as the user desires, simply by using their smartphone as a controller, as illustrated in figure2.

Figure I.2 Smart Home Automated Network example [21]

### I.4.2. Smart House, Past Present and Future

a) **Past:** while the potential of electricity for domestic use was discovered around 1882, it wasn't until the 1920's that it started to become a staple in most homes. In many ways, electricity was the beginning of technology in the home. To say that we've come a long way since then is an understatement. It wasn't long before inventors and manufacturers recognized the potential of electricity for domestic use. By the mid-1900s, homes boasted a wide range of features that were unimaginable the decade before. It wasn't smooth sailing for these brands, however. Lucky for us all, home device s have gotten smaller, sleeker, and more advanced since then [4].

b) **Present:** fast forward to today, having a virtual assistant such as Alexa or Google Home is the standard for ordinary families. These devices allow you to set timers, create lists, ask questions, and play music by simply making requests. Home security has also never been more evolved than it is right now. Video doorbells allow you to check who is at the front of your door on your phone. You can even control your door locks, garage doors, and security cameras all from your phone or tablet. Everything from smart faucets with motion sensors that help conserve water to digital refrigerators featuring touch-screens and voice command helps make the home an ever- evolving hub of connectivity. The next step in home technology is connectivity, also known as home automation. Home automation consists of connecting all your devices to the internet. Imagine being able to control everything in your home with just one touch. With home automation, these scenarios are not unrealistic. On the flip side, remember when you used to have to manually turn the channel on the television or go to the library when you were writing a research paper – if you're of a certain age, of course. It wasn't very long ago that you had to run towards the radio to press the "play" and "record" button at the same time to record your favorite song on a cassette player. With today's technology, those instances seem as if they were hundreds of years ago, not twenty [4].

c) **Future:** while it may seem like we've already reached the peak of possibilities in home technology and automation, there is still plenty to look forward to. As devices in the home continue to advance, users can expect significant money and energy savings. Individuals can also expect technology to anticipate their needs and wants before being notified of them. The development of more smart appliances and security can also be anticipated. From lights that turn on and off by themselves without being prompted to doors that open with face recognition, there are still lots to look forward to in this space. Kitchen gadgets are expected to reach new heights as well. Consumers are used to having only a few options when it comes to home technology, but as more companies explore the market, there will be more variety that may also be more affordable. The possibilities are seemingly endless regarding technology in the home. As it continues to evolve and gain popularity, users can expect to save time and money. However, a world where everything is connected and shared with big companies seems like something to be wary of to a certain degree. In the end, the consumer will have to decide what they are comfortable with [4].

### I.4.3. Architecture of Smart Homes

Residents can communicate with smart devices; and in order to send or receive commands, all of those things will be connected to the home network.

Home networking enables the home to be fully connected and controlled both internally and externally. External connectivity is provided via Ethernet or the Internet network through the home gateway.

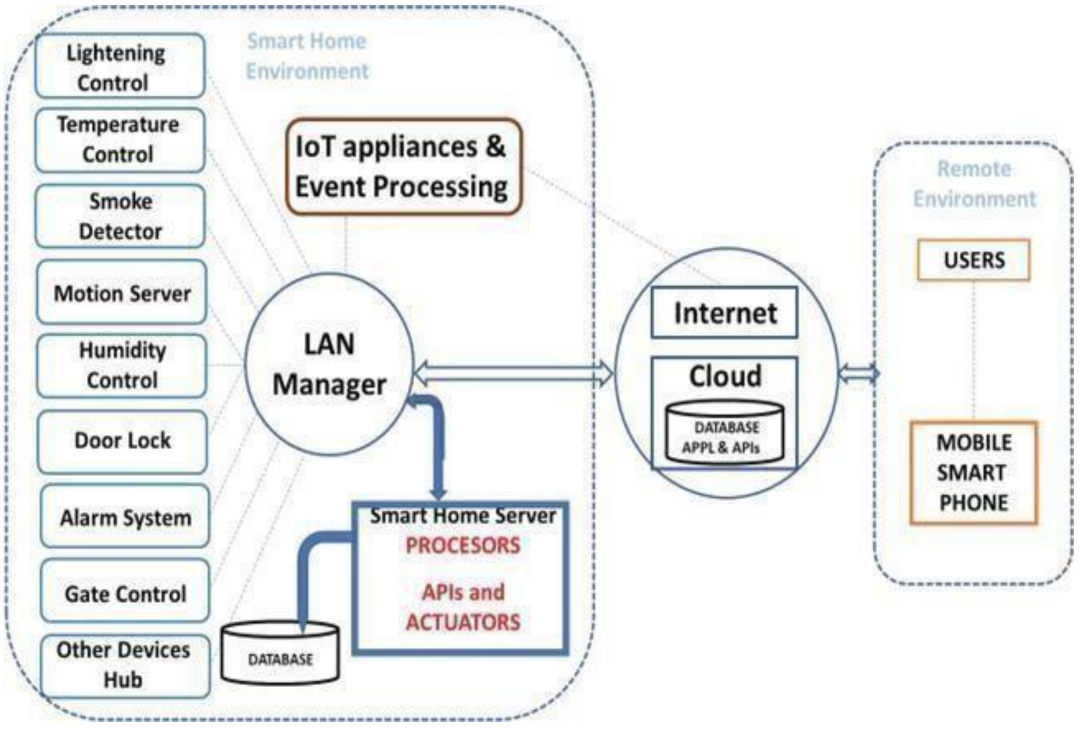


Figure I.3 Architecture of smart Home interaction between indoors & outdoors [2]

We can see in Figure 3, that the appliances and user interfaces are all located inside the house, which represents: offline mode "indoors" which is completely secure, especially if homeowners are concerned about cyber security attacks,safe because no internet connectivity is used, so they can't access to the home system, but the thing is, the appliances will work separately which means each task is performed independently, for example: you can scan the tag to open your door but can't turn on the fan in this case, because it is controlled remotely and in this case we will need to use internet in order to monitor all of the home from a distance using the control panel to perform tasks without requiring human intervention which represents outdoors where we remotely control the devices through a mobile, pc, etc...

This connection between indoors and outdoors via the internet is what makes the home smart.

## I.4.4. Types of Smart Home Systems

There are two types of smart home systems:

- The wireless system (Wi-Fi, radio waves…etc.).
- The smart wiring system (wired system).

### I.4.4.1. Wireless System

Wi-Fi is uniquely placed to support IoT applications that require internet connectivity in order to work. Some IoT applications, such as vehicular services, or video-based apps like connected security cameras, will need the bandwidth of the wireless broadband network, implemented to enable other requirements [24] [25].

### I.4.4.2. Smart Wiring System

Smart-wiring is a part of a smart home to help achieve home-automation, although many smart home devices are wireless, others need a direct physical connection, it's a way to better support wireless devices, because Not all devices controlled by smart home are smart themselves, it's the combination we create between them that make the system intelligent, therefore it enables many different types of wires used around the home into a single location, which allows for the integration of many different smart home systems and sensors throughout the home to be controlled easily[45]. As figure 4 illustrate how all rooms wiring are managed by one central hub which is the utility room as you can see in the image, therefore signal transmission through wires won't face any noise or interruption which define a good monitoring quality.

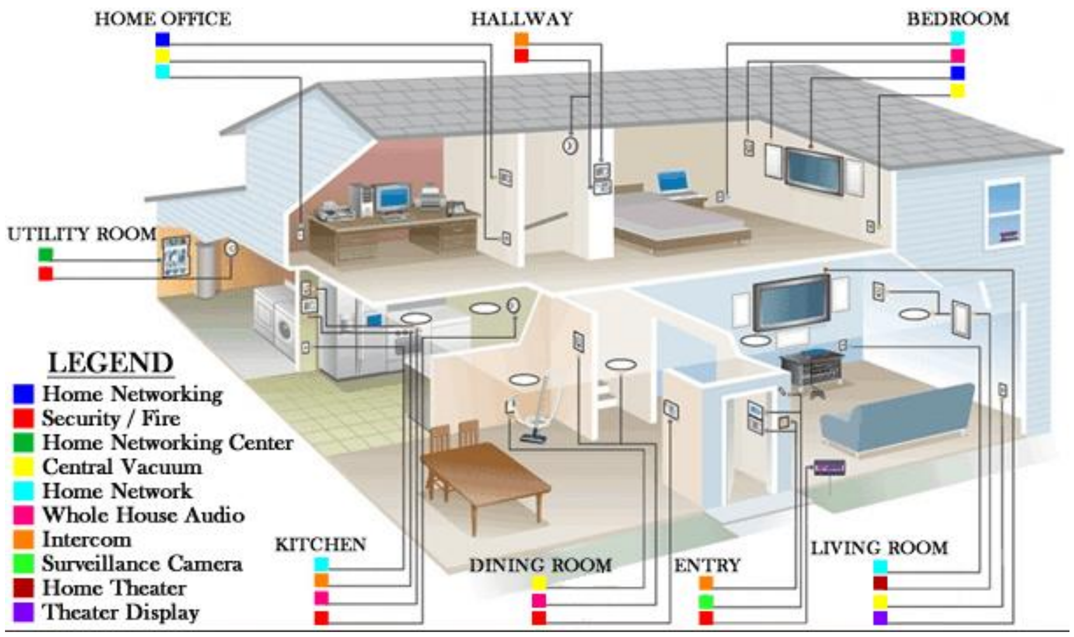


Figure I.4 Example of smart wiring [45]

### I.4.5. What Are the Advantages of Home Automation Using IoT?

The following are the primary services provided by Smart Homes:

- **Security & safety:** Improving security and safety are the most critical elements of household electronics. Therefore, many basic security options, such as Wi- Fi-enabled cameras, smart sensors, like the ones which can detect water leaks, humidity levels, carbon monoxide, motion-heat…etc. might be a life saver, all thanks to their ability to communicate with us, by notifying us in our smartphones, wherever we are, to prevent any possible danger.

- **Energy Efficiency & saving:** This category tries to reduce home energy usage by scheduling and adjusting the working time of home devices based on energy availability [3]

- **Convenience:** A smart home's residents should have the finest possible life experience. Where the home environment adapts to what owners are doing or wishes, with just a touch of a button.

One of the key advantages of home automation utilizing IoT is that this sort of service seeks to lessen the difficult household chores while also ensuring healthcare, childcare, and elder care [3].

### I.4.6. Problems Smart Homes Solve

The modern home is a very different place, to how it was even 50 years ago where smart was to be able to turn lights on or off with a switch on the wall. But not all that has happened since then. By the turn of 21$^{st}$ most homes had things that were one thought to be pure luxuries like multiple televisions, refrigerators.., but all those devices had restrictions when compared to smart devices, they need an actual human being to make them do things. The stage was set for the next evolution in home appliance convenience.

So, creating a smart home? What does it really mean!! We think it's smart when we can take control of our home from your smartphone, tablet or computer or even better when things manage themselves without us having to do anything at all, like the lights in our house being turned on or off automatically when you arrive home! What about those mornings when there's been unexpected meeting, you're in hurry and you forgot to turn the lights off or forget to close garage!

Well there are several nagging tasks that we can pass to the house to do instead. However, here we are going to mention couple ones as follow [5]:

**1- Did I leave the garage door open?**

When it comes to home security, there is probably nothing worse than leaving your garage door open. The garage door is one of the most preferred break-in points by most burglars, and leaving it open makes it and the rest of your home an irresistible target to the bad guys.
Thankfully, home automation allows you to add smart garage monitoring and controlling devices that alert you if you left your garage door open and will enable you to lock it remotely.

**2- Did I leave the lights open?**

It happens so very often that this is the most common and the worst annoying problem of any homeowner. You get home from work just to find out that the lights had been on throughout the day. Or even better, your children are in the habit of forgetting to switch off the lights when they go out of the home. The easiest and the most intelligent solution to this problem is the use of smart lights. With the help of smart lights, you can forget about leaving them switched on while you step out of your home every day for work. The control that this particular smart device provides to the homeowner is flawless and needless to say it adds up to a considerable energy saving too. Which in return, ends up in cost-savings.

**3- Am I safe in my house?**

With all kinds of sensors that we can use for all the smart houses out there, you can ensure your safety while being indoors no matter what kind of danger there is. Gas sensors to detect gas, fire sensors to detect fire, and a way to monitor all of that data and more even while being outdoors to make sure your house is perfectly safe.

With prove smart home appliance we can create schedule for automation each with a specific task, wouldn't that be useful, to be able to interact and control our homes anywhere, anytime we wish through our smartphone. Therefore, making use of smart technology will make our life so much easier.

### I.4.7. Smart Homes in Algeria (Statistics)

First, considering the number of companies active in the field is less present compared to what is actually done abroad. This is due to the absence of the big companies in the sector, either voluntarily or involuntarily on the national market or exposed techniques that do not adapt to the style or standard of living of Algerians.

Home automation is almost impossible to find or very little used in Algerian housing. Because it is not seen as a need or a necessity. In a country where a building with an elevator is seen as a luxury/prestige as well as apartments delivered in 2001 have not yet been completed and where electricity and water failures persist.

In Algeria, the electronic remote control protocol is little used (automatic light, watering, opening and closing doors, garages, windows automatically), despite the fact that the electrical energy meets the local needs in the country.

Despite many aspects an intelligent home can offer and perform better than a human can. We choosed to talk about two most important ones in our opinion, which are: safety and energy saving:

1. The economic boom from the 2000s led to the creation of large residences (pavilions, villas, buildings and duplexes) sumptuous and well appointed, thus, the home automation market was revived. But if Home automation were to be implemented, economy would be a significant development and beneficial to people, here are two main advantages:
    - Almost no waste, therefore so much Energy saved. And this is one of the most compelling reasons to save one budget.
    - Controlling light consumption saves money. Automation of devices, such as light systems, opening or closing blinds, kitchen appliances… prevents loss and reduces annual electricity costs by almost 20-50%. Installing such a system increases the value of the home if it is sold.

2. When it comes to security, it means the residents and their belongings' safety. It is among the main bases of home automation to consider security a top priority in the majority of homes, especially in these times when innovation abounds in anti-intrusion systems, such as sensors, alarm detectors and surveillance videos. Once these security systems are installed whether a homeowner is within or outside their home they can feel very safe knowing they will be notified of anything that is happening in and out of where they live (for example setting alarms on while sleeping, if detecting any danger the alarm will be set on, make a noise in order for the homeowner to wake up), which is a detail that most Algerian homes do not have but need.

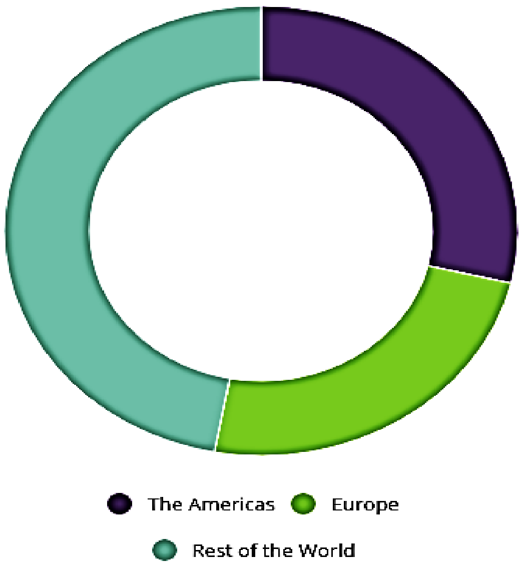
Though, in Algeria Home automation has long been unfavorable due to non-fertile environment: less use of online payment, remote monitoring is not easy, Digital camera installation is very complicated and time consuming. Concerning potentially dangerous domestic incidents such as water or gas leaks, fires, and floods, as well as other issues that are waiting to be resolved. We believe that these are more than sufficient reasons to allow us to develop this safe implementation in our homes in order to increase safety

As a result, we believe that this project is critical enough to invest time and effort into. In order to implement a bright future and, hopefully, a better standard of living in the coming years.

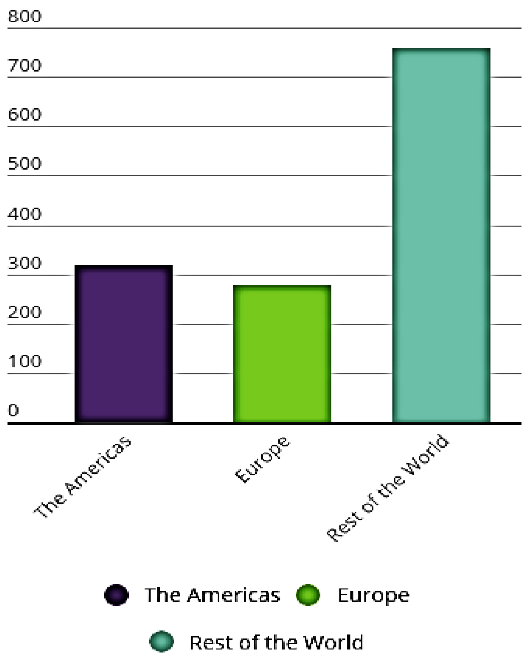
On the other hand, Smart Homes in global scale, this info graphic (figure5) below shows the projected total global number of connected smart home monitoring/automation units in service and the total revenues for smart home automation/monitoring technologies in 2024, as forecast by Juniper Research:

## Smart Homes: Market Summary & Key Takeaways

**Total Connected Smart Home Automation/Monitoring Units in Service (m) by 2024, Split by Region - 1.3 billion**



**Total Revenue from Smart Home Automation/Monitoring ($m) by 2024, Split by Region - $57 billion**



**555 million**

Voice assistants used to control smart home devices are expected to reach 555 million by 2024.

**Alexa, Assistant & Siri**

Lead the way but Chinese manufacturers such as iFlytek and Baidu are gaining momentum.

**25**

High-profile partnerships developed by Roost, smart home telematics provider for the property insurance industry.

**Amazon, Alphabet & Samsung**

Lead the way when it comes to smart security solutions.

JUNIPER RESEARCH

Figure I.5 Smart homes in market [47]

## I.5. Conclusion

The smart home is designed in order to provide comfort and safety for the family inside the home and to provide a decent living for them. It also contains features that facilitate the elderly and patients to carry out those normal daily activities that they are unable to do in the normal state. These services are represented in the ability to operate home appliances from afar, monitor the house and make sure that the doors are closed and the alarm system is activated without the need to return and do this manually and provide a safe and comfortable atmosphere for the children. And other tasks that the smart home performs. And as we saw in this chapter previously. What distinguishes home automation is the control system, which makes it easy to control, monitor and connect devices to each other, and those interaction are studied in next chapters.

Home automation improves people's lives by automating a variety of tasks, where the sky's the limit when it comes to what you can make a home do.

Knowing the state of smart homes in Algeria specially, our project aims to demonstrate that such a project is achievable and very doable through solving the basic problems that we can have and finding more efficient ways to make life easier through its implementation.

Out of many proposed systems, we've choosed six functions that we thought were the most interesting and focused on implementing them. They can be summed up as follows:

- ✓ Fire & gas detectors that inform us through phone notifications about the fire & gas levels of the house and whether we should worry about it or not.
- ✓ Displaying house Temperature & humidity status.
- ✓ Home garage opening & closing through a phone tap, plus automatic evacuation through a fan in case of gas leak.
- ✓ Door lock security system that allows you to have an extra layer of security & make sure you know who is in and out of the house.
- ✓ Controlling lights-lighting across the house from wherever you are, plus lighting based on movement without manual interfering.
- ✓ Security camera that detects movement and warns in case something fishy is going on.

How we achieved these objectives? That is what we will discover in the next chapters.

# CHAPTER II : IOT COMMUNICATION PROTOCOLS

# CHAPTER II: IOT COMMUNICATION PROTOCOLS

## II.1. Introduction

In the communication domain for smart homes, we find two challenging objectives. The first one is how to make possible the communication of the equipment inside the house. The second one is to connect the smart house to the outside Internet world.

However, the most important part relies on the selection of the relevant connection structure including the effective messaging protocol. Therefore, before selecting the appropriate messaging protocol for IoT systems, the pre- requisite is to get a better understanding of a target IoT system and its message/data sharing requirements.

Unlike the Web, which uses a single standard messaging protocol (HTTP/HTTPS), IoT cannot rely on a single protocol for all its needs. What we mean by that is in the IoT structure we can use multiple protocols, where each one does a specific task, compared to a single protocol that is only designed for a sole purpose, back to Http examples that were implemented in the goal of accessing web pages or websites. Consequently, many messaging protocols are available to choose for various types of requirements of the IoT system. Some of them have been designed to address applications requiring fast and reliable business transactions such as AMQP (Advanced Message Queuing Protocol) and JMS (Jakarta Messaging). A numerous have been designed to address applications requiring data collection in constrained networks such as MQTT and CoAP. Many of them have been designed to address applications requiring instant messaging (IM) and online presence detection such as XMPP and SIP. A few of them have been designed to address web applications requiring communicating over the Internet such as Restful client/server protocols HTTP and CoAP [6].

This clearly shows that the future of the IoT lies in several messaging protocols, but also it depends mostly on the case studied or used, for example in our case we focused mostly on one protocol, which is responsible for appliances communication. Consequently, it is necessary to investigate the pros and cons of the widely accepted and emerging messaging protocols for IoT systems to determine their best-fit scenarios.

## Available Technologies in Communication Domain

There are Different kinds of Area Networks, and we can see on the Figure below, five of them. The first one, WANs (Wide Area Networks), generally consist of satellites or antennas installed on towers or on buildings. They serve great geographical areas. These networks can be served by satellite or terrestrial cellular technologies or by fixed solutions without wire. The second one, MANs (Metropolitan Area Networks) serve an area, for example the customers of a district. The third one, LANs (Local Area Network) serve the personal needs for an individual who is responsible to manage his own network. The fourth one, PANs (Personal Area Networks) serve the needs for a user with close objects such as a mobile telephone. The fifth one, BANs (Body Area networks) are a continuity of the personal network, but on a smaller scale. This type of network is mainly based on the principle of smart objects localized on the body and, even in the body of the user [2].

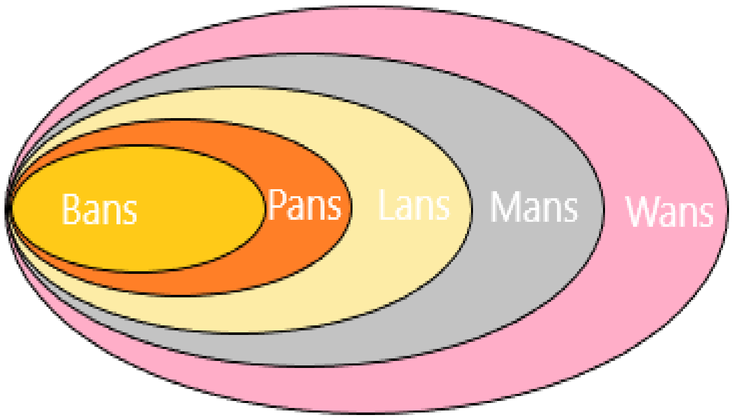


Figure II.1 Different kinds of Area Networks [2]

WANs and MANs are used for outdoor environment:

- For the WANs, we find the UMTS, EDGE, GPRS or satellite technologies. Those technologies are wireless (WWANs: Wireless Wide Area Networks) and are able to transmit information at a distance of up to 30 Kilometers.
- For the MANs, we find WIMAX which is able to transmit information at a distance of up to 20 Kilometers.
- LANs, PANs and BANs are used in indoor environments.
- For LANs, Wi-Fi and HyperLan are mainly wireless solutions. Ethernet is the main wire solution.
- For PANs, Bluetooth, RFID, ZigBee, UWB are wireless solutions. CEBus, Convergence, emNET, HAVi™, HomePNA™, HomePlug™, HomeRF™, Jini™ technology, LonWorks, UPnP, VESA, USB and serial links are wire solutions.
- For BANs, few solutions are existing now. We can note BodyLAN solutions who use the skin to transmit data.

We will focus on indoor solutions to transmit data. For the communication in a smart home, we will focus on PANs because they are the most adapted in terms of distance and flow. We can find residential networking standards and initiatives that we are going to develop.

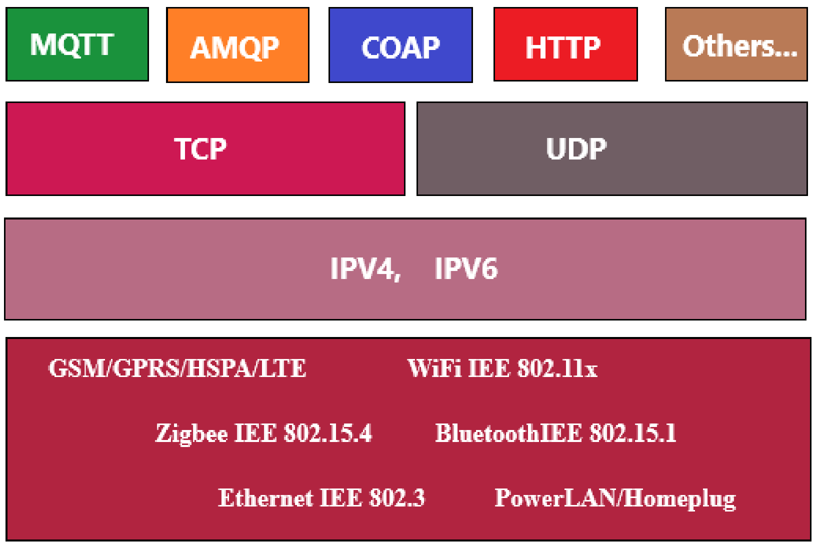


Figure II.2 Effective Messaging Protocols in different layers [6]

Figure 2 above, shows widely used messaging protocols and its layer position in IOT networks, where the:

- **Application layer (1st Line):** is the layer that provides protocols that allow user to send and receive Information to users, a few example of protocols are the: Http, Mqtt, Amqp, Coap, and others.
- **Transport Layer (2nd Line):** manages the delivery and error checking of data packets. It regulates the size, sequencing, and transfer data between systems and hosts. Takes data transferred in the application layer and breaks it into segments on the transmitting host. It is responsible for reassembling on the receiving host, turning it back into data that can be used by the application layer. The transport layer san data at a rate that matches the connection speed of the receiving device, an error control, checking if data was received incorrectly an if not requesting it again
- **Network Layer (3rd Line):** this layer is responsible for routing the data via the best path where it receives frames from data link layer, then deliver them to their intended destination among base on the addresses obtained inside the frame, it finds the destination, one of the most common examples of the network layer is IP (internet protocol). At this layer routers are crucial components used to quite literally route information where it needs to go between networks.
- **Data Link Layer (4th Line):** this layer is responsible for converting data streams to signals bit by bit and sending that over the underlying hardware. At the receiving host, Data link layer picks up data from hardware which are in the form of electrical signals, assembles them in a recognizable format, and hands over to the upper layer. Example of data link protocols are Ethernet, WiFi, ZigBee, …etc

We simply defined the layers in the Osi model (figure II.2) so that we could compare them to the IoT layers and see the difference. So, from the image below (II.3), we can see that there are four layers present, which can be divided as follows:

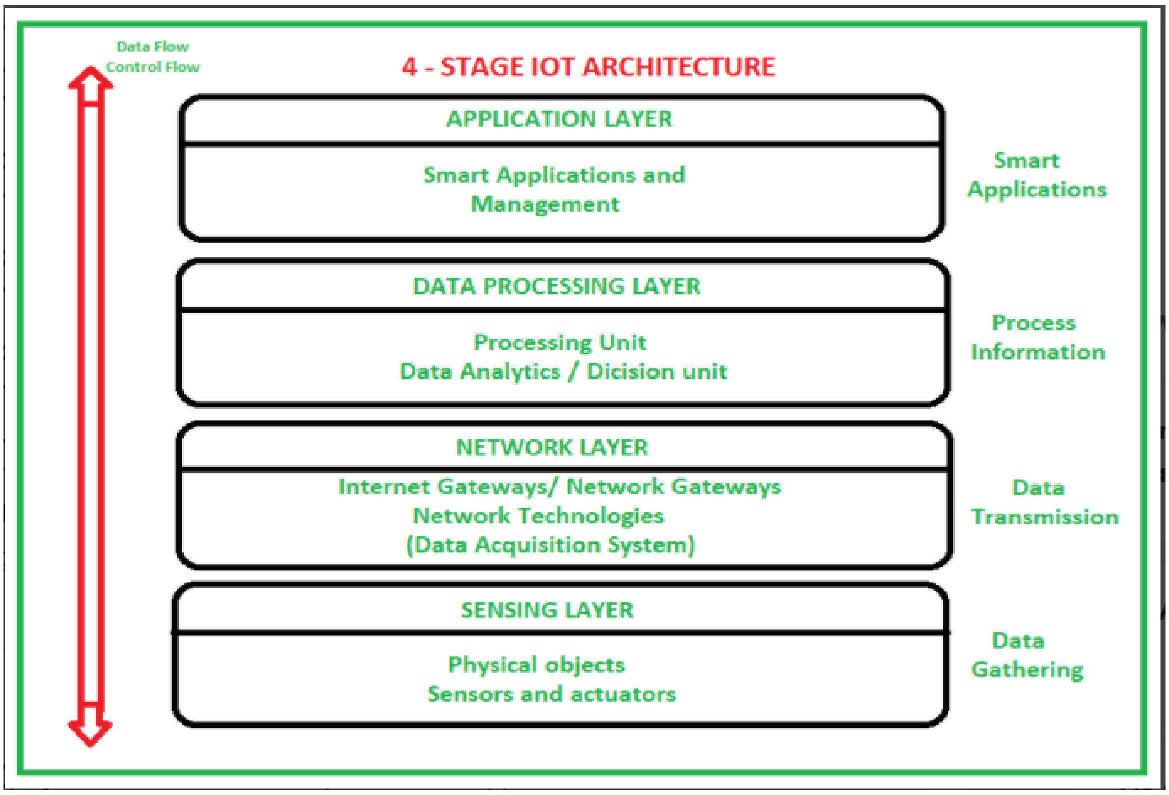


Figure II.3 The 7 layers of the Osi Model [44]

- **Sensing Layer:** sensors, actuators, devices are present in this Sensing layer. These Sensors or Actuators accepts data (physical/environmental parameters), processes data and emits data over network.
- **Network Layer:** internet/Network gateways, Data Acquisition System (DAS) are present in this layer. DAS performs data aggregation and conversion function (Collecting data and aggregating data then converting analog data of sensors to digital data etc). Advanced gateways which mainly opens up connection between Sensor networks and Internet also performs many basic gateway functionalities like malware protection, and filtering also sometimes decision making based on inputted data and data management services, etc.
- **Data processing Layer:** this is processing unit of IoT ecosystem. Here data is analyzed and pre-processed before sending it to data center from where data is accessed by software applications often termed as business applications where data is monitored and managed and further actions are also prepared. So here Edge IT or edge analytics comes into picture.
- **Application Layer:** this is last layer of 4 stages of IoT architecture. Data centers or cloud is management stage of data where data is managed and is used by end-user applications like agriculture, health care, aerospace, farming, defense, etc.

So, depending upon different application areas of Internet of Things, it works accordingly as per it has been designed/developed upon its functionality and implementation in different sectors of use.

## II.2. Major Protocols Used in Home Automation

The IoT devices are typically connected to the Internet via an IP (Internet Protocol) network. However, Bluetooth and RFID features allow IoT devices to connect locally. Therefore, there is a difference in power consumption, range, and memory usage. Connections through IP networks are comparatively complex, require more memory and power but work with high ranges and bandwidth. On the other hand, non-IP networks demand comparatively less power and memory but have a range limitation and low bandwidth [7].

As far as the IoT communication protocols or technologies are concerned, a mix of both IP and non-IP networks can be considered depending on usage.

IoT protocols standards can be broadly classified into two separate categories: network protocols and messaging protocols.

### II.2.1. Wireless Network Protocols

IoT network protocols are used to connect devices over the network. There is a set of communication protocols typically used over the Internet. For example, in wireless communication: signals are transmitted via the air without the use of cables, wires, or other electrical conductors.

As we know, with the rise in wireless technology, many additional chances are open to think about implementing creative ideas since it become easier with the great connectivity network availability nowadays. As an example building an automation network system at home, which can handle everything from entertainment centers, heating and lighting, to manage your home's wireless network consisted of the wanted appliances to control. We've already discussed a few protocols that can be used to implement within the system, Here’s some below:

#### II.2.1.1. Wi-Fi (Wireless Fidelity)

Wi-Fi is a wireless protocol that was built with the intent of replacing Ethernet using wireless communication over unlicensed bands. Most Wi-Fi devices use 2.4GHz frequency and implement frequency division multiplexing technology. Therefore, using Wi-fi help Smart home's users, to exchange information with home appliances and to monitor and control them from afar by performing certain commands using a personal computer or a mobile phone [26].

#### II.2.1.2. Bluetooth

Bluetooth is a standard IoT protocol used for wireless data transmission. This communication protocol is secure and perfect for short-range, low- power, low- cost, and wireless transmission between electronic devices. Also, BLE (Bluetooth LowEnergy) is a low-energy version of Bluetooth protocol that reduces the power consumption and plays an important role in connecting IoT devices.

Bluetooth protocol is mostly used in smart wearables, smartphones, and other mobile devices, where small fragments of data can be exchanged without high power and memory. [26]

#### II.2.1.3. Z-wave

The Z-Wave is a relatively new wireless home automation protocol. It’s a radio frequency (RF) based communications technology, designed particularly for control, monitoring and status reading of household applications. It supports full mesh networks, enabling numerous Z-Wave devices to communicate with each other simultaneously.

Z-Wave allows for secure and lower consuming communication between approve Z-wave devices. Due to its interoperability, Z-wave encompasses a broad ecosystem of intelligent products that work together between brands and models. With the advanced technology of Z-Wave, there is no interference from Wi-Fi, ZigBee, or other 2.4GHz wireless technologies in a similar ban [26].

#### II.2.1.4. ZigBee

ZigBee is an IoT protocol that allows smart objects to work together. It is commonly used in home automation. More famous for industrial settings, ZigBee is used with apps that support low-rate data transfer between short distances. Street lighting and electric meters in urban areas, which provides low power consumption, the ZigBee communication protocol is also used in security systems and in smart homes [26].

#### II.2.1.5. Radiofrequency

This is the frequency band used for communications transmission and distribution. We can control home appliances using a pair of RF modules (transmitter and receiver). It is one of the wireless waves with a form of electromagnetic radiation with frequencies ranging from 3 kHz to 300 GHz. ZigBee is a more powerful and longer-range RF device [27].

#### II.2.1.6. Global System for Mobile Communication (GSM)

Is technology used to control home appliances such as light, conditional system, and security system via Short Message Service (SMS) text messages. GSM protocol allows the user to control the target system away from residential using the frequency bandwidths. The concept of serial communication and AT commands has been applied towards development of the smart GSM-based home automation system. Home owners will be able to receive feedback status of any home appliances under control whether switched on or off remotely from their mobile phones [28].

### II.2.2. Messaging Protocols

IoT data protocols are used to connect low power IoT devices. These protocols provide point-to-point communication with the hardware at the user side. Connectivity in IoT data protocols is through a wireless, wired or a cellular network.
Messaging protocols can be used in the application layer of the OSI model with several communication protocols. Some of the messaging protocols are [7]:

#### II.2.2.1. Message Queue Telemetry Transport (MQTT)

MQTT started as an IBM proprietary protocol developed by Andy Stanford-Clark and Arlen Nipper of Arcom Control Systems Ltd (Eurotech) in 1999.Were used to communicate with SCADA systems in the Oil and Gas industry. It is now an open-source protocol that is overseen by the Organization for the Advancement of Structured Information Standards (OASIS). The MQ in MQTT stands for “Message Queuing”, however, there is no message queuing in MQTT communication anymore. The protocol now provides publish-and-subscribe messaging and has become popular with smart automation systems. Today, MQTT is one of the leading open-source protocols used in fog and edge computing, and for connecting the Internet of Things (IoT). In addition to MQTT, there are other popular messaging protocols that support IoT applications. These include Advanced Message Queuing Protocol (AMQP), Constrained Application Protocol (CoAP), Extensible Messaging and Presence Protocol (XMPP), Data Distribution Service (DDS), ZigBee, and ZWave. That we are going to mention some down below. Then, take a look at the MQTT protocol and its architecture, how it works and how it’s used in IoT, including some real-world applications [6] [8] [9].

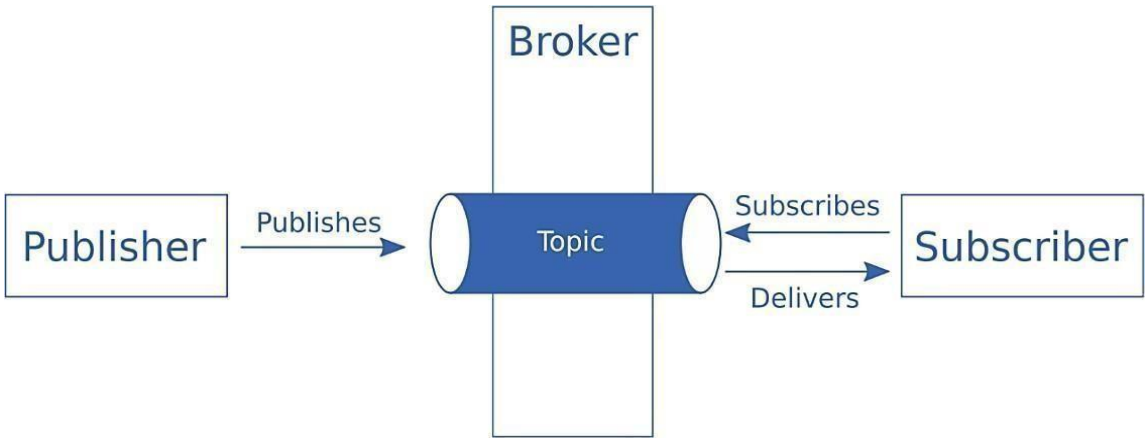


Figure II.2 Message delivery in MQTT [13]

In figure 4, we see the sending and receiving process is a publish/subscribe model, the figure here represents the case of one publisher and one receiver, but we can have multiple publisher with multiple subscribers, it depends on the situation used.

The publisher plays the role of the sender (e.g. temperature sensing device), it publishes the temperature data “payload” under its topic (e.g. Temp). The Broker handles that topic with its payload and delivers it immediately to the devices that are already subscribed to the same topic (e.g. devices that need the temperature data to handle other commands must be subscribed to “Temp” topic before data delivery process) when it is received.

Here’s an Overview of Mqtt working, that we will discuss later in details. For example, let’s say we have a setup where a humidity sensor needs to send its readings to the broker, and on the other end, a computer and a mobile device need to receive this humidity value. In order to get the readings across to the receiving devices, the MQTT publish/subscribe data flow process goes through the following steps:

- Firstly, the humidity sensor defines the topic it wants to publish on, in this case, “Temperature”. Then, it publishes the message “Temperature value”.

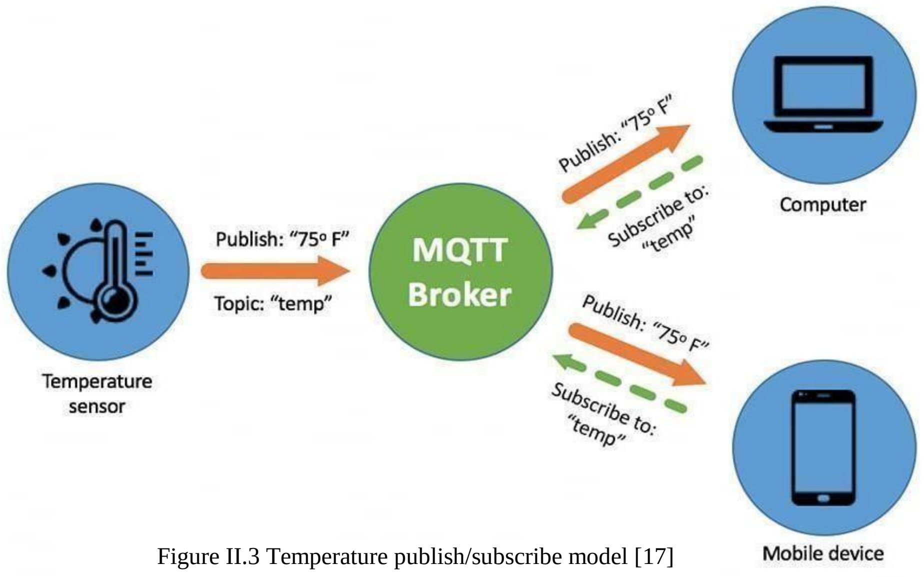


Figure II.3 Temperature publish/subscribe model [17]

- Secondly, the computer and mobile device at the receiving end subscribes to the topic “Temperature”. This enables them to receive the message that the humidity sensor has published– humidity value.

As stated earlier, the role of the broker here is to take the message “Temperature value” and deliver it to the receiving computer and mobile device.

### II.2.2.2. Hypertext Transfer Protocol (HTTP)

HTTP is predominantly a web messaging protocol, which was originally developed by Tim Berners-Lee. Later, it was developed by IETF and W3C jointly and first published as a standard protocol in 1997.

It supports request/response RESTful Web architecture. Analogous to CoAP, HTTP uses Universal Resource Identifier (URI) instead of topics. Server sends data through the URI and client receives data through particular URI. HTTP is a text-based protocol and it does not define the size of header and message payloads, rather it depends on the web server or the programming technology. It also uses TCP as a default transport protocol and TLS/SSL for security. Thus, communication between client and server is connection-oriented. It does not explicitly define QoS and requires additional support for it. It was designed for greatest interoperability on the Web and did not include reliability as a core feature [6].

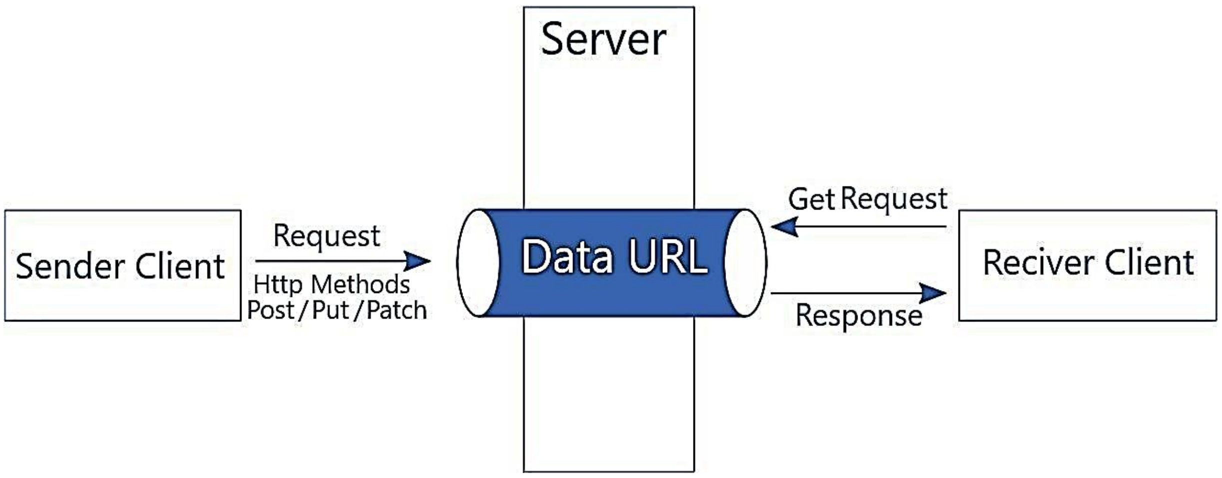


Figure II.4 Message delivery in HTTP [48]

Sender client/ server / receiver client: generally explain the actions of each. HTTP server generally serves only the requested HTTP methods, therefore, we consider a message delivery in HTTP when a client requests to update data in a specific URL and another client requests to read data from the same URL.

### II.2.2.3. Constrained Application Protocol (CoAP)

CoAP protocol is mainly used in automation, mobiles, and microcontrollers. The protocol sends a request to the application endpoints such as appliances at homes and sends back the response of services and resources in the application. For light-weight implementation, it uses UDP (User Datagram Protocol) as a transport protocol and DTLS for security and reduces space usage. The protocol uses binary data format EXL (Efficient XML Interchanges).

CoAP is mainly developed to interoperate with HTTP and the RESTful Web through simple proxies. Unlike MQTT, CoAP uses Universal Resource Identifier (URI) instead of topics .CoAP normally requires a fixed header of 4-bytes with small message payloads up to maximum size dependent on the web server or the programming technology. Thus, clients and servers communicate through connectionless datagrams with less reliability.

Though CoAP does not provide explicit QoS, it facilitates the use of non-confirmable messages (NONE) and confirmable messages (CON), which is very similar to MQTT QoS 0 and QoS 1.

CoAP is a part of the Web architecture and best suited for devices that support UDP or a UDP analogue, however, making it limited to a few special kinds of IoT devices because since reliability is a must while UDP won't provide that [6].

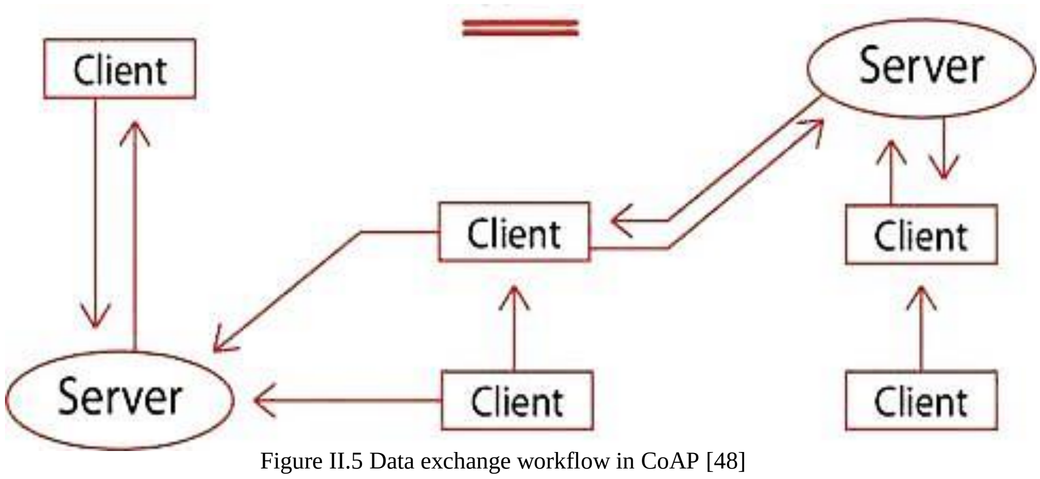


Figure II.5 Data exchange workflow in CoAP [48]

CoAP messaging protocol is based on Request/Response abstraction; generally, the servers are the devices that have sensors or control units and the clients are the devices that apply actions depending on the response results from the servers.

### II.2.2.4. Advanced Message Queuing Protocol (AMQP)

AMQP is a lightweight M2M protocol, which was developed by John O'Hara at JPMorgan Chase in London, UK in 2003. It is a corporate messaging protocol designed for reliability, security, provisioning and interoperability. AMQP supports both request/response and publish/subscribe architecture. It offers a wide range of features related to messaging such as reliable queuing, topic-based publish-and-subscribe messaging, flexible routing and transactions.

AMQP communication system requires that either the publisher or consumer creates an "exchange" with a given name and then broadcasts that name. Publishers and consumers use the name of this exchange to discover each other. Subsequently, a consumer creates a "queue" and attaches it to the exchange at the same time. Messages received by the exchange have to be matched to the queue via a process called "binding".

AMQP exchanges messages in various ways: directly, in fan out form, by topic, or based on headers. AMQP normally requires a fixed header of 8-bytes with small message payloads up to maximum size dependent on the broker or the programming technology. AMQP uses TCP as a default transport protocol and TLS/SSL and SASL for security. Thus, the communication between client and broker is connection-oriented.

Reliability is one of the core features of AMQP, and it offers two preliminary levels of Quality of Service (QoS) for delivery of messages: Unsettle Format (not reliable) and Settle Format (reliable).

AMQP protocol is mainly used in the banking industry. Whenever a message is sent by a server, the protocol tracks the message until each message is delivered to the intended users/destinations without failure.

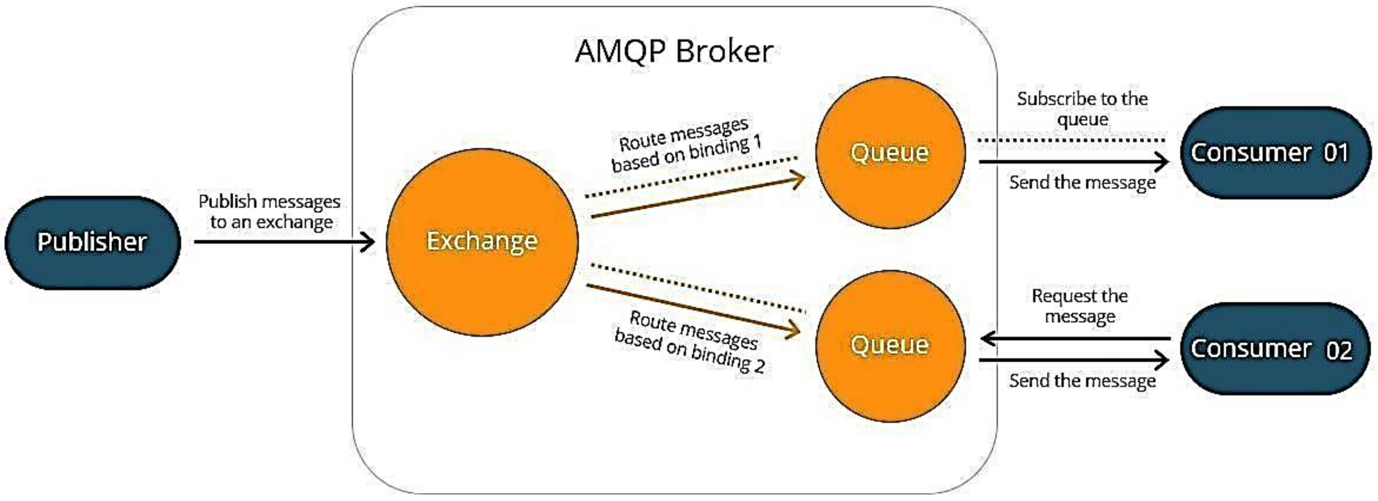


Figure II.6 Message delivery in AMQP [49]

AMQP messaging protocol works in two abstractions : (publisher/subscriber or request/response), and both could be combined together: the publisher is the device that sends data to the AMQP broker, then, the broker route the data based on its binding to a specific queue (see figure 5).

The consumer 01 is subscribed to the specified queue, therefore it will receive the data immediately when the publisher sends it, but the consumer 02 is not subscribed, so it uses the (Request/response) mode. In this case, when the publisher sends data and the broker specified binding to queue of the consumer 02, the data will stay stored there and may be changed even if the consumer did not request it. As a result, the consumer will get new data only if it requests it from the broker.

Now that we got to know a little bit about each messaging protocol. Find below a table and a few charts that compare these protocols on many aspects:

| **Criteria** | **MQTT** | **CoAP** | **AMQP** | **HTTP** |
|---|---|---|---|---|
| **1. Year** | 1999 | 2010 | 2003 | 1997 |
| **2. Architecture** | Client/Broker | Client/Server OR Client/Broker | Client/Server OR Client/Broker | Client/Server |
| **3. Abstraction** | Publish/ Subscribe | Request/Response OR Publish/Subscribe | Request/ Response OR Publish/Subscrib e | Request/ Response |
| **4. Header Size** | 2 Byte | 4 Byte | 8 Byte | Undefined |
| **5. Message Size** | Small and Undefined (up to 256 MB maximum size) | Small and Undefined (normally small to fit in single IP datagram) | Negotiable and Undefined | Large and Undefined (depends on the web server or the programing technology) |

| | | | | |
|---|---|---|---|---|
| **6. Semantics/ Methods** | Connect, Disconnect, Publish, Subscribe, Unsubscribe, Close | Get, Post, Put, Delete | Consume, Deliver, Publish, Get, Select, Ack, Delete, Nack, Recover, Reject, Open, Close | Get, Post, Head, Put, Patch, Options, Connect, Delete |
| **7. Quality of Service (QoS)/ Reliability** | QoS 0 - At most once (Fire-and-Forget), QoS 1 - At least once, QoS 2 - Exactly once | Confirmable Message (similar to At most once) or Non-confirmable Message (similar to At least once) | Settle Format (similar to At most once) or Unsettle Format (similar to At least once) | Limited (via Transport Protocol - TCP) |
| **8. Standards** | OASIS, Eclipse Foundations | IETF, Eclipse Foundation | OASIS, ISO/IEC | IETF and W3C |
| **9. Transport Protocol** | TCP (MQTT-SN can use UDP) | UDP, SCTP | TCP, SCTP | TCP |
| **10. Security** | TLS/SSL | DTLS, IPSec | TLS/SSL, IPSec, SASL | TLS/SSL |
| **11. Default Port** | 1883/ 8883 (TLS/SSL) | 5683 (UDP Port)/ 5684 (DLTS) | 5671 (TLS/SSL), 5672 | 80/ 443 (TLS/SSL) |
| **12. Encoding Format** | Binary | Binary | Binary | Text |
| **13.Licensing Model** | Open Source | Open Source | Open Source | Free |
| **14. Organisational Support** | IBM,Facebook, Eurotech, Cisco, Red Hat, Software AG, Tibco, ITSO, M2Mi, Amazon Web Services, InduSoft, Fiorano | Large Web Community Support, Cisco, Contiki, Erika, IoTivity | Microsoft , JP Morgan, Bank of America, Barclays, Goldman Sachs, Credit Suisse | Global Web Protocol Standard |

TABLE II.1Comparative Analysis of Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP [6]

In the following graphs, we will look at a comparison of which points each protocol good at. First, we will start with message size and power consumption in figure 9 and 10:

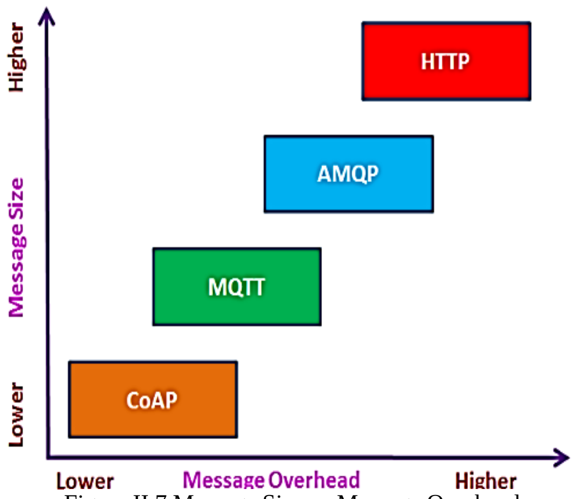


Figure II.7 Message Size vs Message Overhead

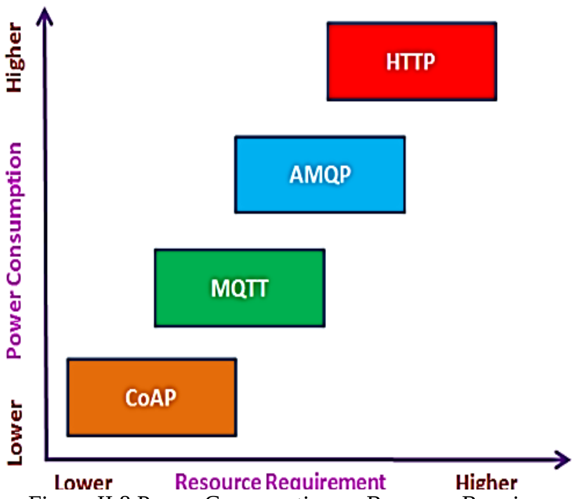


Figure II.8 Power Consumption vs Resource Requirement

1- *message size Vs. message overhead in figure 9 can be summed up as follow:*

| **MQTT** | **CoAP** | **AMQP** | **HTTP** |
|---|---|---|---|
| -Designed for low bandwidth and resource constrained devices<br>-No packet loss | -Requires lowest power and resource. | -Requires slightly higher power and resources due to performing other necessary operations for provisioning and reliability | -Requires the highest power and resource than any other protocols.<br>-needs greater processing power and resources for the same operation. |
| **MQTT** | **CoAP** | **AMQP** | **HTTP** |
| -Has the least header size of 2-byte per message<br>-Its requirement of TCP connection increases the overall overhead, thus the whole message size<br>- Less overhead with no packet loss | -Incurring the lowest message size and overhead and thus the whole message size. | -Its support for security, reliability, provisioning and interoperability increases the overhead and message size | -It requires maximum overhead and message size.<br>-among all four is the most verbose and heavyweight protocol. |

2- *Power Consumption vs Resource Requirement in figure 10 can be summed up as follow:*

Then, see the diffrence when it comes to bandwidth and reliability in figure 11 and 12:

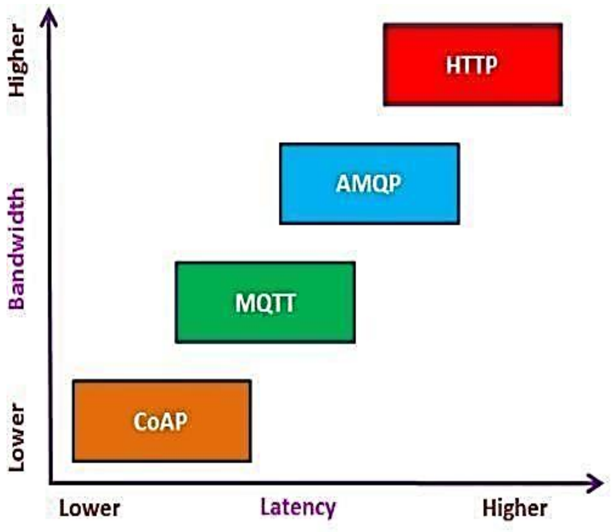


Figure II.10 Bandwidth vs. Latency

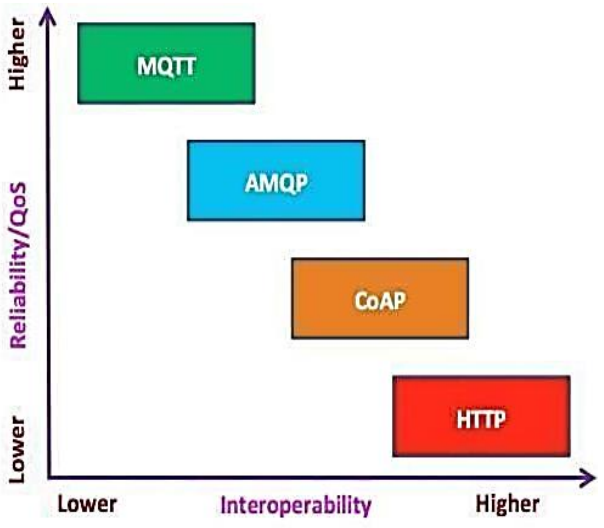


Figure II.9 Reliability/QoS vs. Interoperability

3- *Bandwidth vs. Latency in figure 11 can be summed up as follow:*

| **MQTT** | **CoAP** | **AMQP** | **HTTP** |
|---|---|---|---|
| -Consumes higher bandwidth than CoAP for transferring same payload under same network condition<br>-The bandwidth usage of MQTT was approximately double than CoAP. This is because of the four-way handshake mechanism of QoS 2. | -Involves lowest bandwidth and latency.<br>-In CoAP, a UDP transaction requires only two UDP datagrams, one in each direction; this reduces the network load response times. | -AMQP's extra services demand moderately higher bandwidth and latency | - Takes significantly largest bandwidth and latency than any other protocols. |

4- *Reliability/QoS vs. Interoperability in figure 12 can be summed up as follow:*

| **MQTT** | **CoAP** | **AMQP** | **HTTP** |
|---|---|---|---|
| -Offers the highest level of quality of services (three QoS) 0- at most once 1- at least once 2- exactly once<br>-Additionally, it also provides "last will and testament" | -Compensates for the unreliability of UDP protocol by defining a retransmission mechanism and providing resource discovery mechanism with resource description. | AMQP defines two QoS levels: Settle Format (similar to MQTT QoS 0) and Unsettle Format (similar to MQTT QoS 1). | -The QoS is not a default service of HTTP; therefore, its default reliability is the TCP guarantee. |

Finally, comparing security aspects (though security is a vast area to dig more details in, this graph only shows more likely an overview). In addition comparing which protocol is most suited in machine to machine environment:

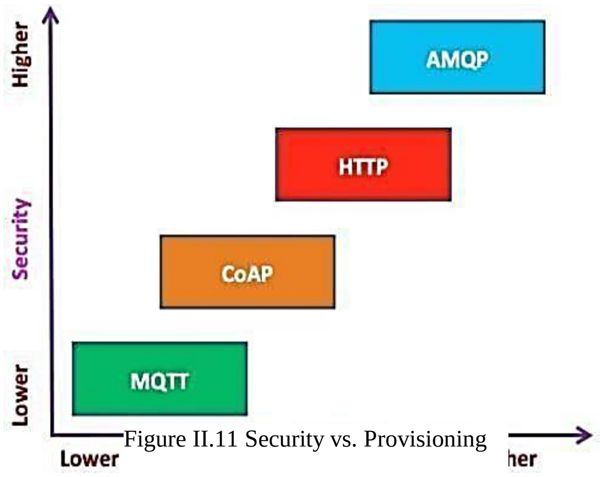


Figure II.11 Security vs. Provisioning

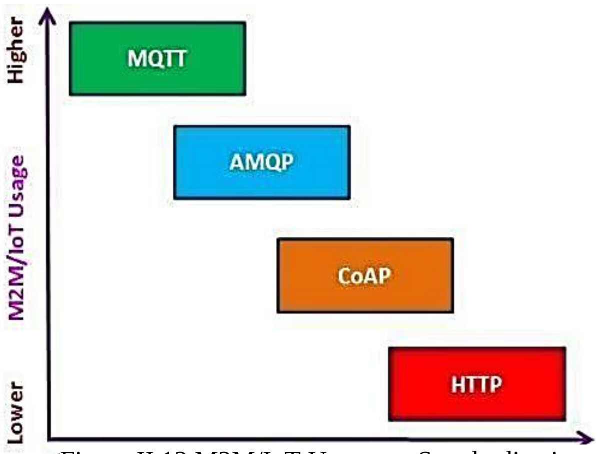


Figure II.12 M2M/IoT Usage vs. Standardization

5- *Security vs. Provisioning in figure 13 can be summed up as fellow:*

| **MQTT** | **CoAP** | **AMQP** | **HTTP** |
|---|---|---|---|
| -TLS/SSL service,<br>-Require username and password. | - Uses two methods DTLS and IPsec for authentication, integrity and encryption. | -Has the highest level of support for security and additional services. | -DTLS and IPsec for authentication, integrity and encryption. |

6- *M2M/IoT Usage vs. Standardisation in figure 14 can be summed up as fellow:*

| **MQTT** | **CoAP** | **AMQP** | **HTTP** |
|---|---|---|---|
| -Ideal for IoT<br>-Hosted by OASIS open standards consortium and Eclipse Foundation<br>-Used and supported by the large number of organisations such as IBM, Facebook, Eurotech, Cisco, Red Hat, M2Mi, Amazon Web Services (AWS), InduSoft and Fiorano. | -CoAP has been swiftly gaining momentum and supported by many large companies such as Cisco (Field Area Network), Contiki, Erika and IoTivity.<br>-CoAP is an IETF standard | - has been employed in projects such as Oceanography’s monitoring of the Mid-Atlantic Ridge, NASA’s<br>Nebula Cloud Computing and India's Aadhar Project. | -HTTP is a global web standard but mostly not suitable and used in the IoT industry.<br>the usage of HTTP in the IoT is limited due to its heavyweight size and slow performance.<br>-HTTP is an IETF and W3C standard and already established as a global standard for the Web. |

If you are not sure about which protocol to use, it’s up to your home requirements, which area you should focus on. Whether you want your whole house appliances to interact with each other, or only some individuals, if the reliability of messages required and transmission bandwidth and latency are the most important factors …ect.
Yes, we did a comparison but it was not about who is the best in all areas, rather, each one can be used depending on the proposed and method of usage required. Therefore, we can't decide which one is the best protocol.

## II.3. Decision of Choosing Our Right Messaging Protocol:

From the previous study, we decided to choose MQTT as our main messaging protocol because we were aiming for: Reliability, low power consumption, low latency, and respectable bandwidth speed and has low message size, which we found these specifications in this protocol. Therefore, a better fit for our project. In addition, it has a simpler functioning architecture, so all these features help to choose low processing power components and develop projects faster. And it was worth it to implement Mqtt as a way for devices to communicate, as for others we acquired a good knowledge, because every protocol has its own value.

Well you may be thinking now, that coap is better than mqtt in terms of message size, resource requirements, and bandwidth as shown in last charts(9,10,11,12), but if reliability isn't achieved among protocol specifications, then previous requirements are mostly meaningless especially If reliability is the most important factor. In short, Coap using UDP defends our decision to go with Mqtt because retransmission of lost data packets is only possible with TCP and not UDP. Now, the time we present how it functions in details:

## II.4. MQTT Protocol (Message Queue Telemetry)

We already have defined Mqtt in messaging protocol section (page27) where we talked about its history. In addition, this section emphasis more details:

First things to know, Mqtt is appropriate to be used on devices that have limited capabilities in terms of bandwidth and data transmission. It's based on a publish/subscribe model which to be considered as MQTT client, whereas Broker is MQTT server, suitable for M2M (machine to machine) & WSN (Wireless Sensor Networks) [8] [9].

Furthermore, it is a lightweight application layer protocol based on TCP/IP, With a light header of 2Bytes size, and QoS key feature that ensures message delivery, as the figure below shows:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 1 | Message Type | | | | Dup flag | QoS level | | RETAIN |
| Byte 2 | Remaining Length | | | | | | | |

TABLE II.2 MQTT Message Format (2Bytes Header) [8]

### II.4.1. Basic Concepts of MQTT

In simple terms, Mqtt basic components are the following:

#### II.4.1.1. Publish/Subscribe:

When a device (or client) wants to send data to a server (or broker) it is called a publisher. When the operation is reversed, it is called a subscribe. Multiple clients can connect to the broker and subscribe to topics that they are interested in. Clients can also publish messages to specific topics of their interest through the broker. The broker is a common interface for devices to connect to, and exchange, data. The publish /subscribe messaging model works somewhat like a TV station. A TV station broadcasts a TV program using a specific channel and a viewer tunes into that channel to view the broadcast. There is no direct connection between the broadcast station and the viewer.

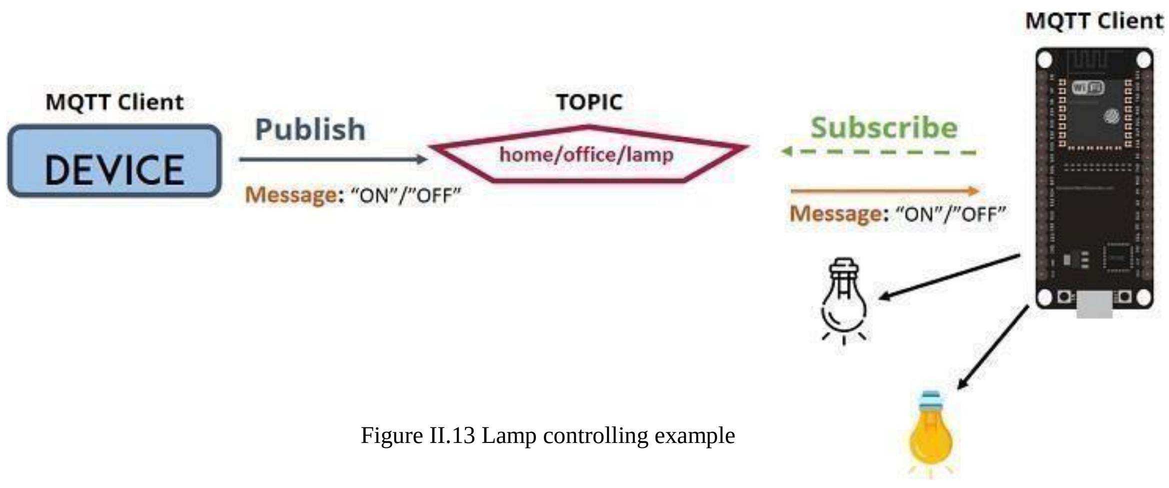


Figure II.13 Lamp controlling example

#### II.4.1.2. Broker

Instead of a device sending messages directly to the recipient device(s), they are sent to the broker, and the broker forwards it to everyone that needs that message. MQTT uses a subject line called the 'topic'. If anyone wants a copy of that message, all they need to do is to subscribe to that topic. On the other hand, if a broker receives a message on a topic for which there are no current subscribers, the broker discards the message unless the publisher of the message designated the message as a retained message.

### II.4.1.3. Topics and Subscriptions:

In MQTT, a Publisher sends (publishes) a message under a topic name. Subsequently, all the subscribers under the topic name receive the message through a broker. We can have as many topics as we want, with each one having different clients on either end. This can be one-to-one, many-to-many, or any combination. The publish/subscribe architecture is very flexible and scalable, making it great for IoT applications.

As stated earlier, the role of the broker here is to take the message “Temperature value” and deliver it to the receiving computer and mobile device.

### II.4.1.4. Quality of Service Levels:

QoS is a key feature of the MQTT protocol that provides additional messaging qualities of service that ensure that the message in transit is delivered as required by the service [10] [11]. It gives the client the power to choose a level of service that matches its network reliability and application logic, because MQTT manages the re-transmission of messages and guarantees delivery of a message. Although TCP / IP provides guaranteed data presentation, data loss may occur if a TCP connection is broken and transmitted messages are lost. Therefore, MQTT adds three QoS levels to the top of TCP to transmit messages [12]:

- At most once (0)
- At least once (1)
- Exactly once (2)

| **QoS 0** | **QoS 1** | **QoS 2** |
|---|---|---|
| -Only TCP guarantee<br>-Often called ‘fire and forget’, Delivery of a message is not guaranteed.<br>-The recipient doesn’t acknowledge receipt of the message & The message isn’t stored and re-transmitted by the sender, | -MQTT guarantee with confirmation<br>-Most frequently used because it guarantees that a message is delivered at least one time to the receiver but allows for multiple deliveries.<br>-The sender stores the message until it gets a Puback packet from the receiver that acknowledges receipt of the message. | -MQTT guarantee with handshake<br>-The highest level of service in MQTT<br>-The message is sent precisely once,<br>-It guarantees that each message is received only once by the intended receiver.<br>-The safest & slowest QoS level. |

TABLE II.3 The different QoS work Concept

The selection of the QoS level depends on the system case, like if it:

| **QoS 0** | **QoS 1** | **QoS 2** |
|---|---|---|
| have a completely stable connection over wired connection, suitable if data loss is acceptable, or the no need of message Queuing,ie: requires the lowest amount of network traffic | The need to get every message, at least once but duplicates are allowed. It delivers messages much faster than QoS 0. | Suitable for scenarios that require a message to be guaranteed and not duplicated, QoS2 interaction takes more time to complete. |

TABLE II.4 When to use Which QoS

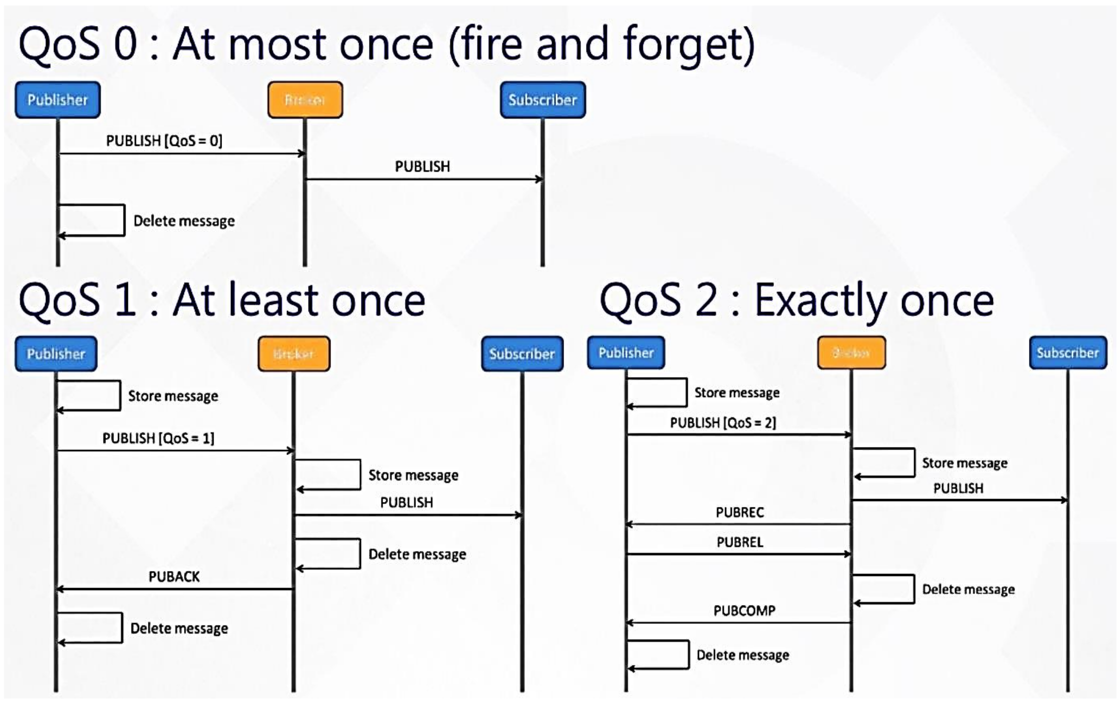


Figure II.16 Message delivery at different QoS levels [46]

The work concept of these messages delivery is explained in table 4 (The different QoS work Concept).

### II.4.1.5. Retained Messages:

A retained message is a normal MQTT message with the retained flag set to true. The broker stores the last retained message and the corresponding QoS for that topic. Each client that subscribes to a topic pattern that matches the topic of the retained message receives the retained message immediately after they subscribe. The broker stores only one retained message per topic.

### II.4.1.6. Wills

Another extra feature MQTT server has is last will & testament (LWT). It is // a message used to notify the reason why a connection between a broker and client, who was actively subscribing or publishing to a topic, was terminated. The notification message is sent to all clients subscribed to that topic, so that any authorized client in the system can publish a new value back to the edge-network device, LWT is not only generated by the lack of a keep alive response but also in case a client ends a connection without a proper disconnect from the broker.

### II.4.1.7. Keep Alive & Client Take-Over

a) **Keep Alive:** ensures that the connection between the broker and client is still open and that the broker and the client are aware of being connected. When the client establishes a connection to the broker, the client communicates a time interval in seconds to the broker. This interval defines the maximum length of time that the broker and client may not communicate with each other.

b) **Client Take Over:** Usually, a disconnected client tries to reconnect. Sometimes, the broker still has a half-open connection for the client. In MQTT, if the broker detects a half-open connection, it performs a 'client take-over'. The broker closes the previous connection to the same client (determined by the client identifier), and establishes a new connection with the client. This behavior ensures that the half-open connection does not stop the disconnected client from re-establishing a connection.

### II.4.1.8. Persistent Session and Queuing Messages

To receive messages from an MQTT broker, a client connects to the broker and creates subscriptions to the topics in which it is interested. If the connection between the client and broker is interrupted during a non-persistent session, these topics are lost and the client needs to reconnect. Re-subscribing every time the connection is interrupted is a burden for constrained clients with limited resources. To avoid this problem, the client can request a persistent session when it connects to the broker. Persistent sessions save all information that is relevant for the client on the broker. The client Id that the client provides when it establishes connection to the broker identifies the session.

## II.5. MQTT Architecture

MQTT's communication model avoids direct connections between devices by relaying data through a central server called the broker. This is really desirable in IoT because it's easy to add new devices without touching the existing infrastructure, and since new devices only need to communicate with the broker, they don't actually need to be compatible with the other clients [10]. To understand more, let's see the following key parts. This system is made up of:

- Client (publisher & subscriber)
- MQTT Broker (the main central processing unit)
- Topic (a reference of the data transmitted)

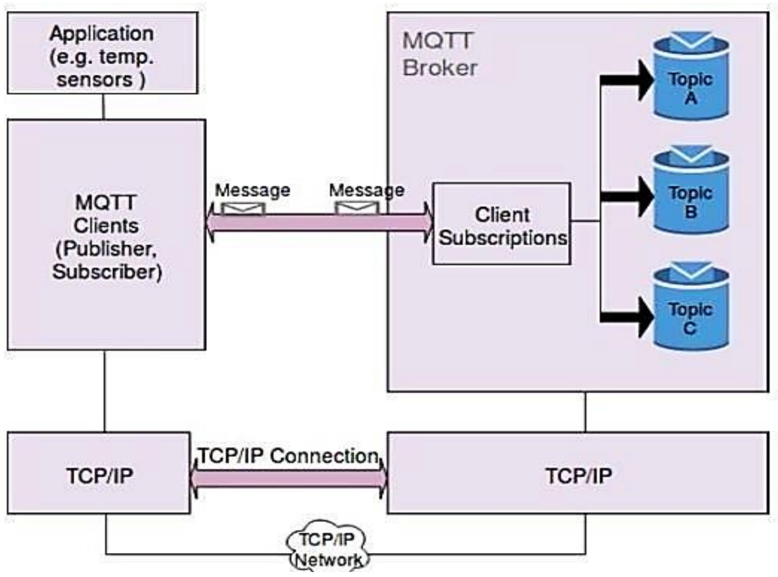


Figure II.17 message transmission in MQTT [10]

As we saw in figure 18, which represents the protocol's architecture. Where 3 main elements are to pay attention to:

a) **Client:**
- Client could be a Publisher or Subscriber or both and it always establishes the network connection to the Server (Broker).
- The publisher is the source of the data on the system to be sent. A subscriber is a client who wants to receive messages from the publisher. In the case of subscription, it gets logically attached to the topic of its interest. When the client subscribes to a topic of its interest it can exchange messages with this topic. It becomes beneficial to use this protocol in the scenario where we want to exchange small messages which require less bandwidth.

b) **Broker:**
- Is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients.
- An MQTT broker or server is software running on a computer that receives messages from external sources– publishers, and then routes them to the appropriate destination subscribers. The computer can be a Raspberry Pi, an on premise desktop PC, or a cloud based server running open-source or proprietary software. One of the most popular open-source message brokers is the Mosquito broker.
- Depending on the implementation, a broker can manage up to thousands of simultaneously connected MQTT clients. Therefore when choosing an MQTT broker, you should consider factors such as scalability and integration. In addition to receiving and routing messages to clients, the broker also delivers other capabilities such as Quality of Service, Store and Forward, Security.

c) **Topic:**
- Every message is published to an address, known as a topic. The address of a message, which represents the subject of the published message, and is used by brokers to filter messages for each connected client. The topic consists of one or more topic levels. Each topic level is separated by a forward slash.

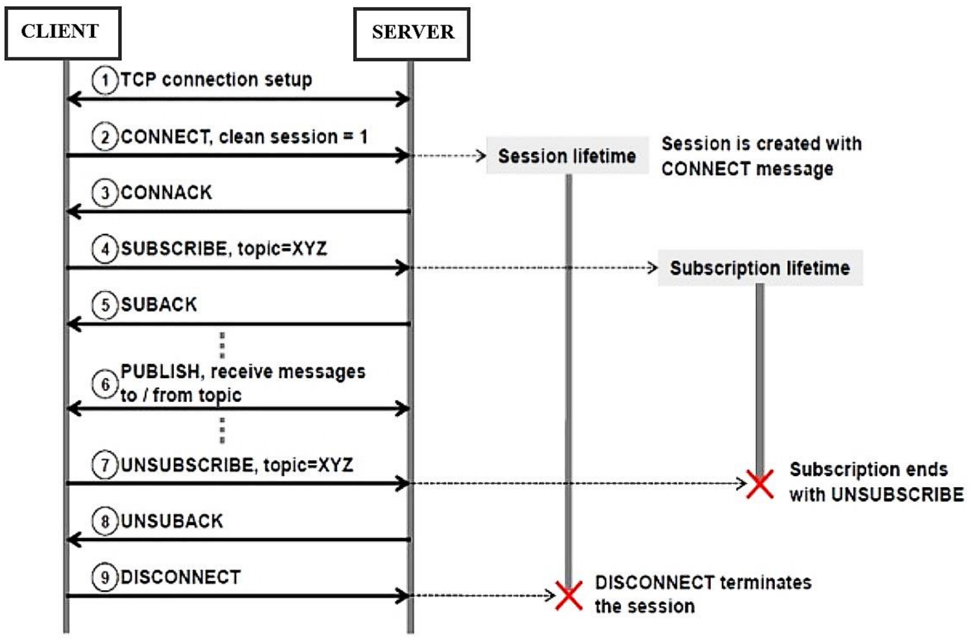

CLIENT
SERVER
1 TCP connection setup
2 CONNECT, clean session = 1
3 CONNACK
4 SUBSCRIBE, topic=XYZ
5 SUBACK
6 PUBLISH, receive messages to / from topic
7 UNSUBSCRIBE, topic=XYZ
8 UNSUBACK
9 DISCONNECT
Session lifetime
Session is created with CONNECT message
Subscription lifetime
Subscription ends with UNSUBSCRIBE
DISCONNECT terminates the session

Figure II.18 Typical message flow in MQTT [46]

Control packets can be summarized in the following groups:

- Connection Management: CONNECT, CONNACK, DISCONNECT, PINGREQ, PINGRESP
- Subscription Management: SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK
- Message Delivery: PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP
- Authentication: AUTH

The AUTH packet was introduced in MQTT version 5.0. PINGREQ is sent from client to server to indicate that the client is alive though not sending any other control packet.

## II.5.1. Brokers of MQTT

The MQTT broker is the heart of each MQTT arrangement. It provides a connecting link between applications or physical devices and enterprise systems. Brokers are in charge of subscription, determined sessions, missed messages and general security, including authentication and authorization. The following table describes mostly used brokers with their features and limitations [10].

| Broker | About | Features | Limitations |
|---|---|---|---|
| **Mosquitto** | It supports MQTT version 3.1 and it is open source. | - All QoS<br>- Authentication<br>- Bridge<br>- Dynamic topics<br>- Web sockets | - Clustering<br>- Fewer Configuration<br>- Not allow simultaneous connection with using authentication |
| **RSMB (Really small message broker)** | It is a tiny broker which supports V3 and V3.1 | - All QoS<br>- Bridge<br>- Dynamic topics | - Security<br>- Web sockets<br>- Cluster |
| **MQTT.js** | It is an MQTT broker among client/server API production | - All QoS<br>- Dynamic topics<br>- Web sockets<br>- SSL | - Bridge<br>- Authentication<br>- Cluster |
| **HiveMQ** | HiveMQ empowers organization To attach all devices and services with nominal effort by victimization the de-facto | - All QoS<br>- Bridge<br>- Dynamic topics<br>- Web sockets<br>- TLS/SSL<br>- Cluster | - Open Standard<br>- Performance degradation because of TLS |

| | | | |
|---|---|---|---|
| **VerneMQ** | VerneMQ could be a superior, distributed MQTT message broker supports MQTT version 3.1 and 3.1.1 | - All QoS<br>- Bridge<br>- Authentication<br>- Dynamic topics<br>- Web sockets<br>- Encryption | - Performance degradation because of TLS |

TABLE II.5 Brokers with features and limitations

### II.5.2. Where Else We Use MQTT

Some of the popular MQTT supported platforms include ThingWorx, IBM Watson IoT Platform, AWS IoT, Fogwing, Braincube, among others. These platforms provides MQTT extension to enable connectivity for IoT. Notable examples and use cases for MQTT technology in the IoT infrastructure include [28]:

- **Facebook Messenger:** in order to solve problems of limited battery life and internet bandwidth associated with smartphone use, Facebook innovatively deployed MQTT for its messenger and Instagram chats that would enable it to function effectively even with the varying internet connections available across the world. Chats are associated with an MQTT topic, and all members of a chat group subscribe and publish to that topic. A "Topic Director" steers the MQTT chat packets to the different brokers that forward them to the appropriate destination subscribers.

- **Wearable:** MQTT protocol is commonly used to enable low-power IoT wearable. The lightweight nature of the MQTT protocol makes it suitable for wearable devices due to its limited memory and bandwidth capacity. Data is published when it is pushed from the wearable device to the server and the receiving devices subscribe to data pushed to the server. Because the size of the data being sent over MQTT is small by design, messages can be sent quickly, making the hardware very responsive.

- **Agriculture Technology:** this sector has the potential to increase and enhance the efficiency of the farming systems by providing continuous monitoring and reporting of the weather and soil conditions such as soil pH, soil temperature and moisture, air temperature, humidity, and sunlight availability. MQTT protocol and IoT sensors for agriculture enable seamless data collection and availability to cloud servers (broker) even for a challenged Internet connection.

- **Remote Sensing:** sensors used in monitoring remote environments are often low power devices operating in locations with a poor internet connection.

### II.5.3. MQTT Benefits and Challenges

| Advantage(s) | Disadvantage(s) |
|---|---|
| - A big advantage of MQTT is that the message used in it can have any format. This means that the publisher and subscriber need to establish beforehand the data format.<br>- Lightweight.<br>- Extremely reliable, ensuring message delivery.<br>- Battery friendly. | - As perfect as it sounds MQTT protocol it has One disadvantage though, which is MQTT dependency to the broker. In MQTT the broker affects the scalability of the overall system, meaning that the network can expand as much as the broker can support it. Also, the broker can be a point of failure in an automation system because the broker usually is plugged directly in a wall socket while the devices that send/retrieve data from it are most of the time battery operated. So, in case of a power outage the broker will be offline while the devices will try to publish data to it [12]. |

### II.5.4. Does MQTT Support Security?

We already discussed a portion of security in figure 14 (Security vs. Provisioning), and we've seen that AMQP outperforms MQTT, but that doesn't mean it will completely replace or surpass it. Security is always a challenging matter that keep addressed by capable developers or cyber security enthusiastic, so as long as we face vulnerabilities, security development is essential. Well it’s a completely different topic to discuss. For the time being, if we're talking about mqtt, then:

MQTT brokers may require username and password authentication from clients to connect for security. To ensure the privacy of messages in transit, TCP connections may be encrypted with SSL/TLS to encrypt and protect data [14], which makes the communication reliable. Additional security can be added by an application encrypting data that it sends and receives, but this is not something built into the protocol, in order to keep it simple and lightweight. However, studying security is a completely different project that we are considering for future work and will undoubtedly devote a significant amount of time to.

## II.6. Conclusion

The choice of MQTT as the main communication protocol between devices was seen as suitable for an IoT smart home system, goes back to its ideal use in constrained environments like IoT which has low power, limited computation capability and memory, limited bandwidth, and simplicity of use, rapid deployment and its reliability at transferring data and commands to remote devices.

Therefore, we dedicated a fair time to understand its working, its architecture, the domains where it is mostly used in, and its various brokers with their limitations and features, even though we never studied this protocol, nor have the slightest info about it.

# CHAPTER III: IOT SMART HOME PROJECT COMPONENTS

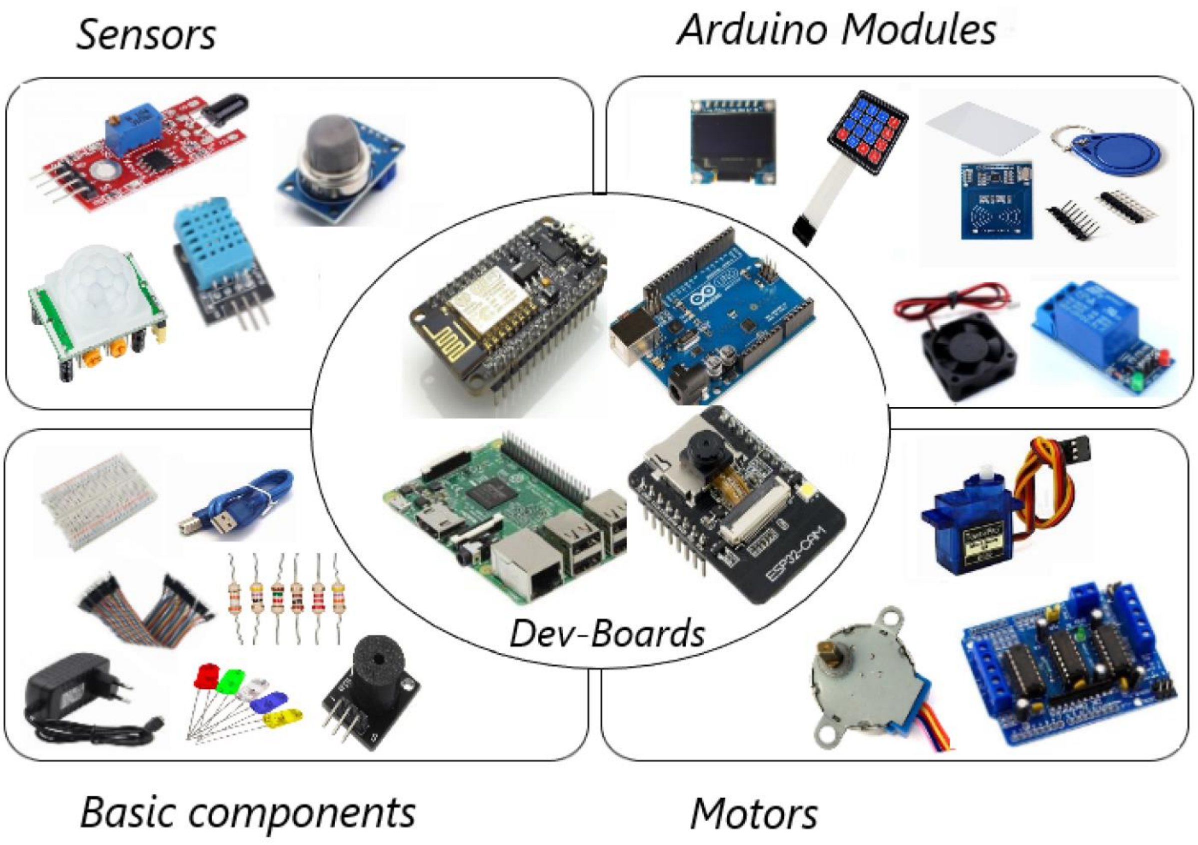
# CHAPTER III: IOT SMART HOME PROJECT COMPONENTS

## III.1. Introduction

IoT smart homes creation became one of the brightest projects on this last years, they created devices that can be installed at homes and linked with some supported appliances to command them. Despite the fact that these devices are expensive, they can effectively interact with human.

However, from our point of view, the drawback of all of these devices is their fully connection to the internet. Thus, it would be an issue. As a result, the main goal of this project is creating devices to do an IOT smart home by a minimal communication with internet services, in order to maximize home independency and privacy with lower cost.

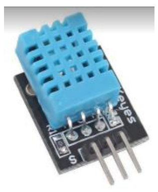
Therefore, on this chapter, we are going to discuss hardware components (as this image below shows, classified equipment) and software used in order to make this project work, their features and their work concept with their configuration process.



## III.2. Hardware

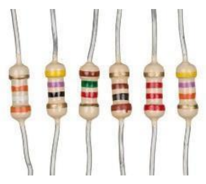
- **List of Components:**

The following list shows all the components and parts required to complete the build of home automation project.

| Figure | Name | Description |
|---|---|---|
| | Raspberry Pi Model B | Is a development board in the PI series. It can be considered as a mini computer that works on the LINUX operating system. The board not only has tons of features it also has terrific processing speed making it suitable for advanced applications [29]. |
| | ESP-8266 (NodeMcu) | Is a microcontroller with built-in Wi-Fi that gives us access to the internet quickly. Apart from Wi-Fi functionality, it allows you to control inputs and outputs as you would do with an Arduino, but it comes with Wi-Fi. So, it is great for home automation/internet of things applications [30]. |
| | Arduino Uno | Is basically a small development board with a brain (microcontroller) where we can store our programs that will tell the Arduino what to do. We can connect to electrical circuits, this makes it easy to read inputs – read data from the outside – and control outputs - send a command to the outside [31]. |
| | ESP-32 Cam | Is a development board with an ESP32-S chip, an OV2640 camera, micro SD card slot and several GPIOs to connect peripherals [32]. |
|  | DHT11 Temperature & Humidity Sensor | It's a great little sensor to measure temperature and humidity in a room or outside. It integrates seamlessly with the Arduino. And communicates using its own proprietary One Wire protocol. This protocol requires very precise timing in order to get the data from the sensor [33]. |
| | MQ-5 Gas Sensor | The MQ5 Gas Sensor module is useful for gas leakage detection. It is suitable for detecting LPG, Natural Gas. Due to its high sensitivity and response time, measurements can be taken as soon as possible. The sensitivity of the sensor can be adjusted by using the potentiometer [34] |

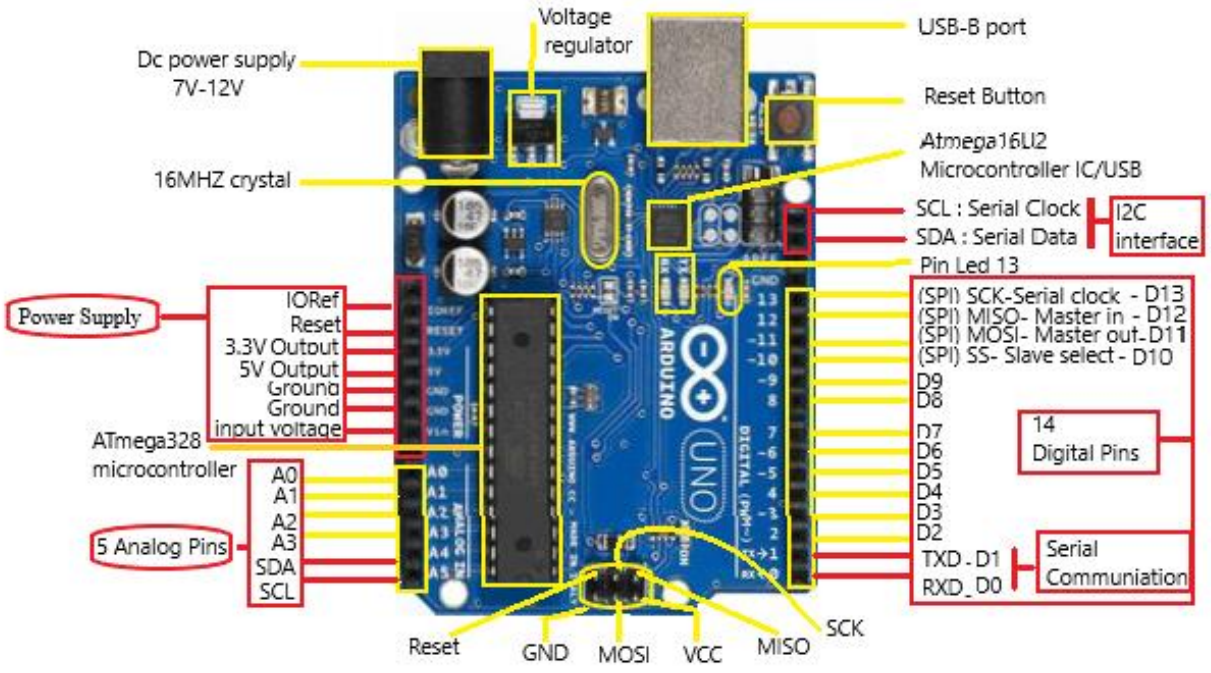
| | | |
|---|---|---|
| | Flame Sensor | The flame sensor is used to detect the fire or other infrared light which are in the range of wavelength from 760 nm to 1100nm. The module consists of an IR sensor, potentiometer, OP-Amp circuitry and a LED indicator. When a flame is detected, the module will turn on its red led. This module is sensitive to flame but it can also detect ordinary light. The detection point is 60 degrees. The sensitivity of this sensor is adjustable and it also has a stable performance [35]. |
| | PIR Motion Sensor HC-SR501 | A Passive Infrared is ideal to detect movement. Basically, the PIR motion sensor measures infrared light from objects in its field of view. So, it can detect motion based on changes in infrared light in the environment. It is ideal to detect if a human has moved in or out of the sensor range [36]. |
| | RFID RC522 (Key, Reader, Card) | RFID means radio-frequency identification. It uses electromagnetic fields to transfer data over short distances, it's useful to identify people, to make transactions….etc, only the person with the right information on his card is allowed to enter. |
| | Relay | The relay is an electrical switch that uses an electromagnetic system to move the switch from the on to off position instead of manual action.<br>In a basic relay, there are three contactors: normally open, normally closed, and common. At no input state, the COM is connected to normally close. |
| | OLED | Is one of the most attractive displays available for a microcontroller. It has a good view angle and pixel density which makes it reliable for displaying small level graphics. It doesn't require backlight, which results in a very nice contrast in dark environments. Additionally, its pixels consume energy only when they are on, so the OLED display consumes less power when compared to other displays. |

| | | |
|---|---|---|
| | 4x4 Membrane Keypad | This 16-button uses a combination of four rows and four columns to provide button states to the host device, typically a microcontroller. Underneath each key is a pushbutton, with one end connected to one row, and the other end connected to one column [39]. |
| | L293D Motor Driver Shield | The L293D Motor driver shield is one of the best ways for controlling DC motors, Servo motors, and Stepper motors in a single board. It can control the rotation direction and speed of four DC motors, two Servo motors, and two Stepper motors. It is easy to connect with an Arduino UNO or MEGA. |
| | Stepper Motor 28-BYJ48 | It is one of the most commonly used stepper motors. We can find this or similar motors in our DVD drives, Motion camera and many more similar devices. The motor has a 4 coil unipolar arrangement and each coil is rated for +5V hence it is relatively easy to control with any basic microcontrollers. |
| | Servo Motor SG90 | Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. |
| | Fan | By supplying the integrated driver with the supported DC current. It will help us in evacuating the house in case of bad air detected. |
| | Bread board | Is used for developing an electronic circuit and wiring for projects with microcontroller boards like Arduino. |

| | | |
|---|---|---|
|  | Resistors | Are defined as a passive two-terminal electrical component that implements electrical resistance as a circuit element. Resistors' primary job is to regulate or to set the flow of electrons (current) through them by using the type of conductive material from which they are composed. The colored bands are used to denote the resistance, tolerance and the temperature coefficient. |
| | JumperWires | Are used for making connections between items on the breadboard and on Arduino's header pins to wire up all built-in circuits. |
| | Leds<br>5mm (diameter) | Use very little electricity, they come with different colors, the difference depends on the semiconductor they are made of. It has a positive(longer leg) and a negative(shorter) lead and will not light if they are the wrong way around and it cannot be directly connected to the voltage source else it will burnout destroying the 'junction' where the light is produced. Therefore, a resistor must be used with, to limit or 'choke' the amount of current flowing through the LED. |
| | Buzzer | A simple sound making module. It is set high or low to drive this module. By changing the frequency a different sound will be heard. |

## III.2.1. Development Boards

### III.2.1.1. Arduino Uno



**Features and Application:**

Applications are a way to present where you can use these components other than home automation.

**Features**

- Flash Memory: 32KB(0.5KB is used for Boot loader), SRAM is 2 KB, EEPROM is 1KB
- Digital input/output pins are 14 (Out of which provide PWM output), Analog input pins are 6 (A0-A5)
- Operating Voltage 5V, Recommended input Voltage 7-12V, limit 20V
- CLK Speed is 16 MHz, a power jack, a reset button, USB connection
- DC Current on 3.3V Pin is 50mA & on I/O pin is 40mA
- Microcontroller: ATmega328P -- 8 bit AVR family microcontroller

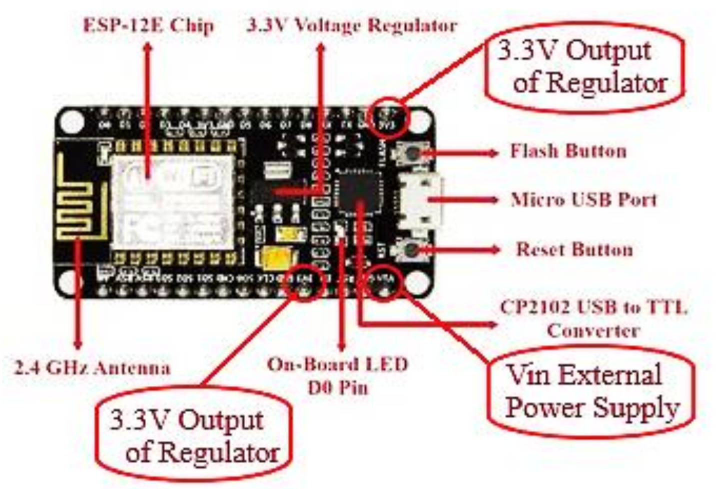
**Applications**

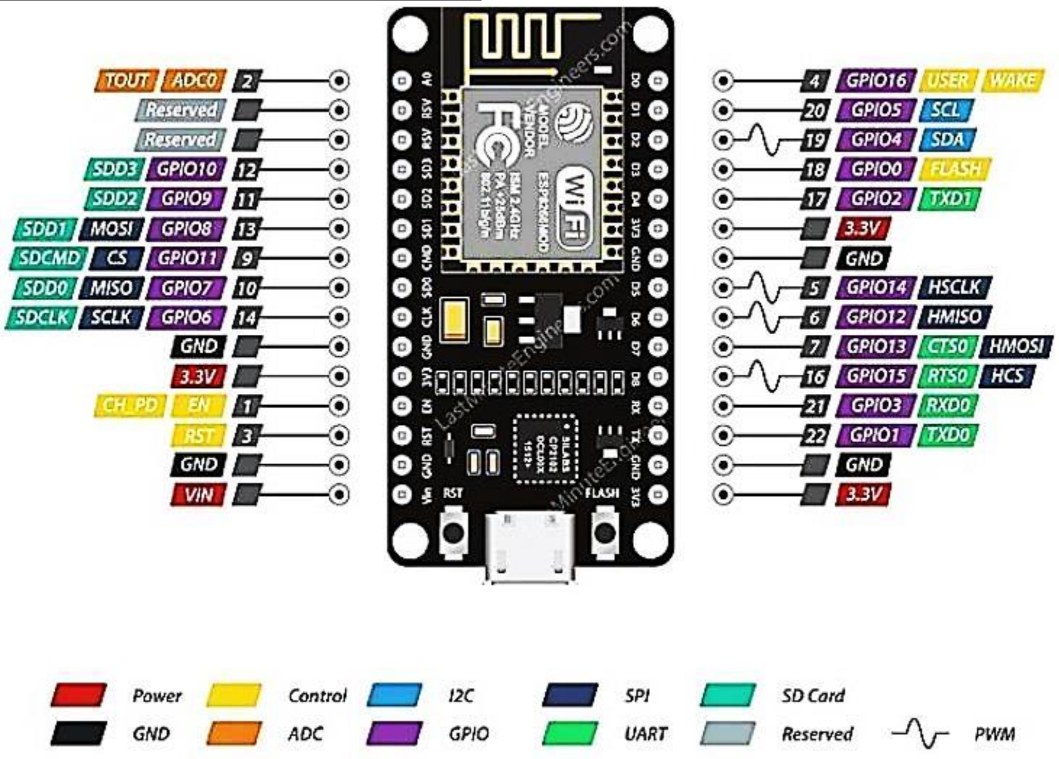
- Prototyping of Electronics Products and Systems
- Multiple DIY Projects (Do It Yourself)
- Easy to use for beginner level DIYers and makers
- Projects requiring Multiple I/O interfaces and communications.

### III.2.1.2. NodeMCU (ESP8266 Wi-Fi Module)

This board is based on ESP8266MOD Wi-Fi module and it contains the ESP8266EX System on chip (SoC) from Espressif Systems Company but it has 4 Megabytes of flash memory.

| Specification |
|---|
| UARTs: 1, SPIs: 1, I2Cs: 1 |
| RAM: 128 KB, Flash Memory: 4MB, SRAM: 64KB |
| Built-in low-power 32-bit CPU ,Clock Speed: 80 MHz |
| Wi-Fi built-in (11 b/g/n protocol), Integrated TCP/IP protocol |
| PCB Antenna |
| Operating Voltage: 3.3V |
| Input Voltage: 7-12V |
| Digital I/O Pins (DIO): 16 |
| Analog Input Pins (ADC): 1 |
| Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106 |





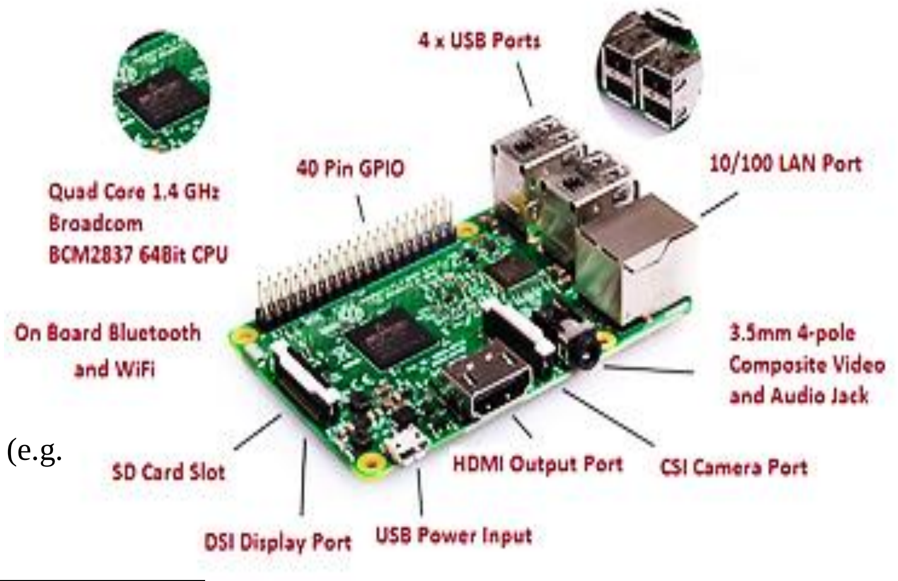
#### Application

- Prototyping of IoT devices
- Low power battery operated applications
- Network projects
- Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities

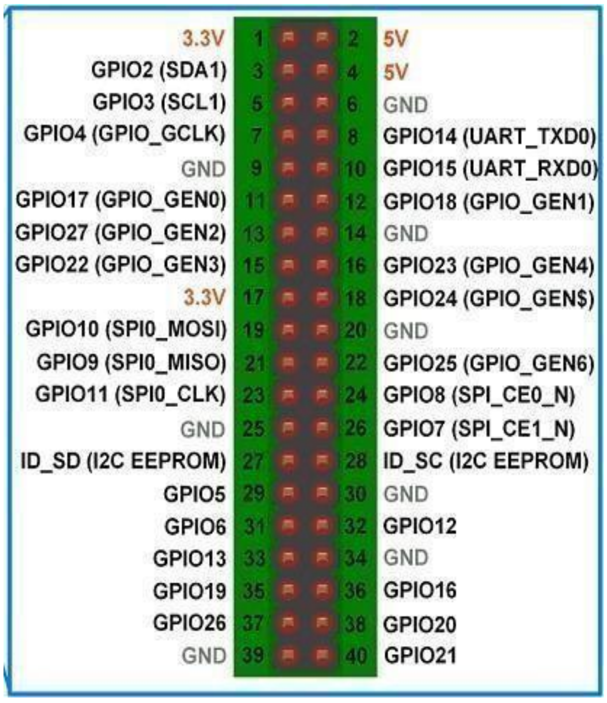
### III.2.1.3. Raspberry PI 3 Model B

#### Applications

- Media center
- Build a home automation system
- Industrial/Home automation
- Server/cloud server
- Print server
- Security monitoring
- Robotics
- Gaming
- Web camera
- Environmental sensing/monitoring (e.g. WEATHER STATION)
- Low cost PC/tablet/laptop


4 x USB Ports
Quad Core 1.4 GHz Broadcom BCM2837 64Bit CPU
40 Pin GPIO
10/100 LAN Port
On Board Bluetooth and WiFi
3.5mm 4-pole Composite Video and Audio Jack
SD Card Slot
HDMI Output Port
CSI Camera Port
DSI Display Port
USB Power Input

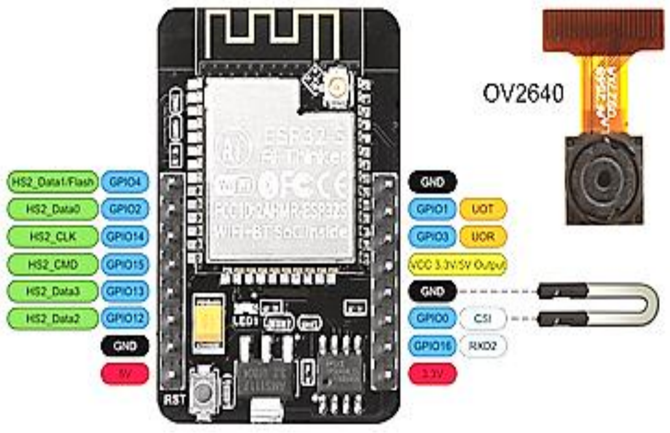
| Specification |
|---|
| 1GB RAM, Flash Memory 16 Gbytes SSD memory card |
| BCM43438 (802.11 b/g/n) wireless LAN and Bluetooth Low Energy 4.1(BLE) on board |
| 40-pin extended GPIO |
| Clock speed 1.2GHz |
| Full size HDMI, 4 Pole stereo output, composite video port |
| Memory Card Slot Push/Pull Micro SDIO |
| Processor Operating Voltage 3.3V, Raw Voltage input 5V |
| 2A power source, Power Micro USB socket 5V1, 2.5A |
| Maximum current through each I/O pin 16mA |
| CSI camera port for connecting a Raspberry Pi camera |
| DSI display port for connecting a Raspberry Pi touchscreen display |
| Micro SD port for loading your operating system and storing data |
| Operating System Boots from Micro SD card, running a version of the Linux operating system or Windows 10 |



| | | | |
|---|---|---|---|
| 3.3V | 1 | 2 | 5V |
| GPIO2 (SDA1) | 3 | 4 | 5V |
| GPIO3 (SCL1) | 5 | 6 | GND |
| GPIO4 (GPIO_GCLK) | 7 | 8 | GPIO14 (UART_TXD0) |
| GND | 9 | 10 | GPIO15 (UART_RXD0) |
| GPIO17 (GPIO_GEN0) | 11 | 12 | GPIO18 (GPIO_GEN1) |
| GPIO27 (GPIO_GEN2) | 13 | 14 | GND |
| GPIO22 (GPIO_GEN3) | 15 | 16 | GPIO23 (GPIO_GEN4) |
| 3.3V | 17 | 18 | GPIO24 (GPIO_GEN$) |
| GPIO10 (SPI0_MOSI) | 19 | 20 | GND |
| GPIO9 (SPI0_MISO) | 21 | 22 | GPIO25 (GPIO_GEN6) |
| GPIO11 (SPI0_CLK) | 23 | 24 | GPIO8 (SPI_CE0_N) |
| GND | 25 | 26 | GPIO7 (SPI_CE1_N) |
| ID_SD (I2C EEPROM) | 27 | 28 | ID_SC (I2C EEPROM) |
| GPIO5 | 29 | 30 | GND |
| GPIO6 | 31 | 32 | GPIO12 |
| GPIO13 | 33 | 34 | GND |
| GPIO19 | 35 | 36 | GPIO16 |
| GPIO26 | 37 | 38 | GPIO20 |
| GND | 39 | 40 | GPIO21 |

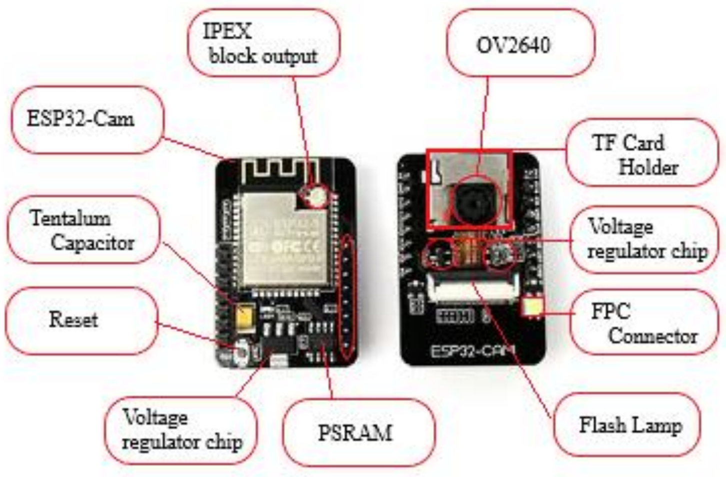
### III.2.1.4. ESP-32 CAM

| Specification |
|---|
| 802.11b/g/n Wi-Fi module |
| Up to 160MHz clock speed |
| summary computing power up to 600 DMIPS |
| Built-in 520 KB SRAM, external 4MB SRAM |
| Supports UART/SPI/I2C/PWM/ADC/DAC |
| Support OV2640 and OV7670 cameras, built-in flash lamp |
| Supports multiple sleep modes |
| Embedded Lwip and FreeRTOS |
| Support Smart Config/AirKiss technology |
| Support for serial port local and remote firmware upgrades (FOTA) |



#### Application

- Video streaming
- Security systems



- For other components details, Check this.

### III.3. Software

What we followed as process along the Journey to build the Home Automation System from start to finish, is the following:

- Interacting  Raspberry Pi, ESP8266 and Arduino boards with each other
- Node-RED, Node-RED Dashboard installed in Raspberry Pi and Linux commands
- Establishing an MQTT connection with multiple devices in the network
- Controlling any device
- Read sensor data (temperature, humidity, luminosity and more)
- Trigger events based on sensor data, time and set modes
- Display home data in gauges and charts
- Build a smoke and motion detection system with email notifications
- Make dashboard password protected and accessible from anywhere in the world

Therefore we will see the software used first, then jump to configuration process:

## Raspberry Pi to ESP8266 over Wi-Fi Using MQTT

For better understanding, we have used the example below, as a means of explaining how devices communicate in between.

Before we continue we will first define software used along the project, then, hit the installation process which is divided in 4 parts:

- Defining the Raspberry pi Ip address as static..
- Set up MQTT server on RPi and install some libraries
- Set up the NOD-RED Dashboard interface on Rpi as well.
- Install libraries in Arduino IDE for NodeMCU(Esp8266) to work with MQTT

Here, server is RPi and client is Esp8266, we are using it as an MQTT publish subscribe client.

From a PC on a local network. Esp8266 and RPi are going to use the same local network which we created in the last blog and ESP8266 as an MQTT publish subscribe client.

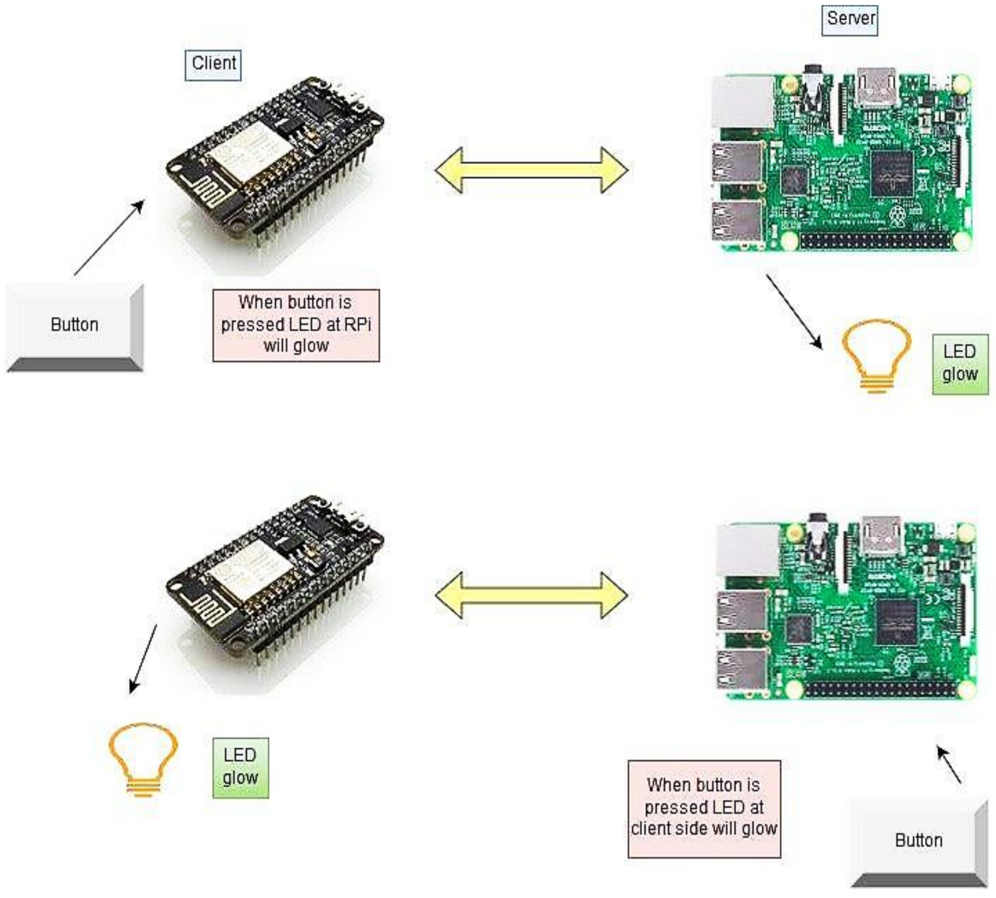

Client
Server
Button
When button is pressed LED at RPi will glow
LED glow
LED glow
When button is pressed LED at client side will glow
Button

Figure III.1 Raspberry Pi to ESP8266 installation process

## III.3.1. Raspberry PI OS (Raspbian)

The Raspberry Pi is a computer and like any other computer it needs an OS installed, here it is called Raspbian [15].
Raspbian is a free operating system based on Debian optimized for the Raspberry Pi hardware. It comes with over 35,000 packages precompiled software bundles in a nice format for easy installation on the Raspberry pi.

- **Raspbian Images**

A Raspbian image (Adafruit Raspberry Pi Linux Distro) is a file that can be downloaded onto an SD card which in turn can be used to boot your Raspberry Pi. The installation process of Raspbian OS to the card is done via Raspberry Imager, the Steps are below.

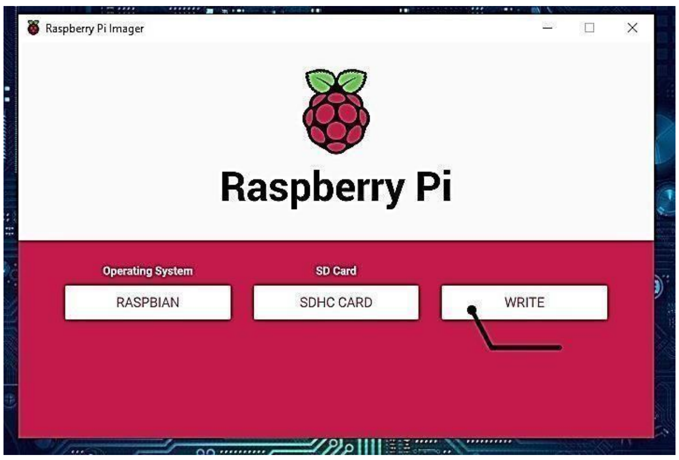
- **How to setup static Ip address on Raspberry pi**

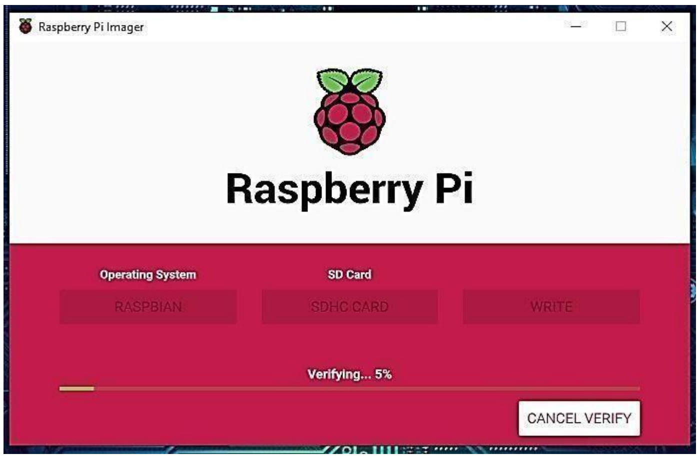
What's important is to set the raspberry pi address to static, to make sure that the same IP will work each and every time. Because, If we leave the IP address as a dynamic it will change all the time.

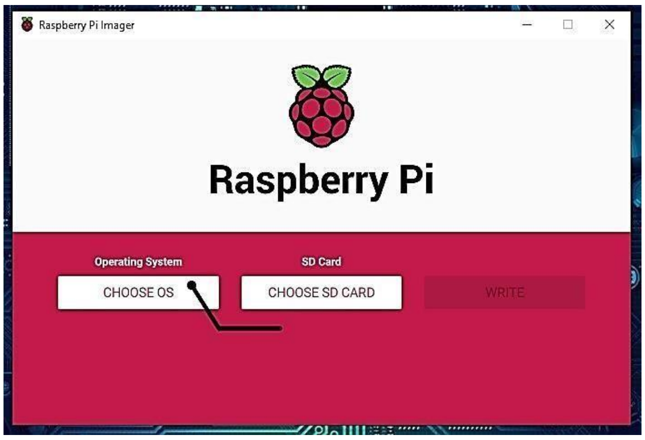
Go to directory cd /etc and open file dhcpcd.conf using any editor. At the end, write these four lines.

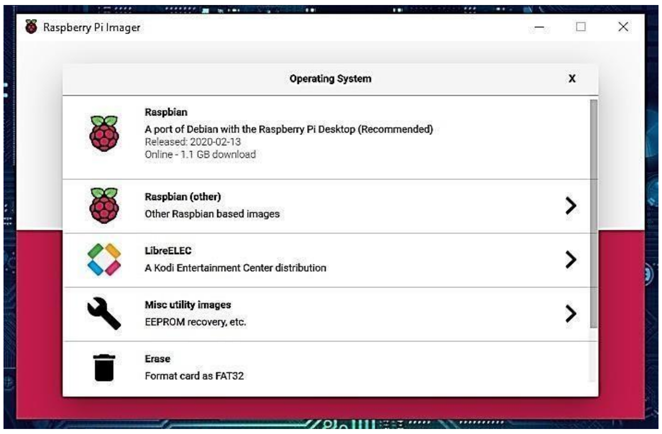
```
interface eth0
static ip_address=192.168.43.99 // for ethernet use interface wlan0
static ip_address=192.168.43.99
static routers=192.168.1.1 // your Default gateway static
```

After that, we save it and reboot the pi to apply the changes.









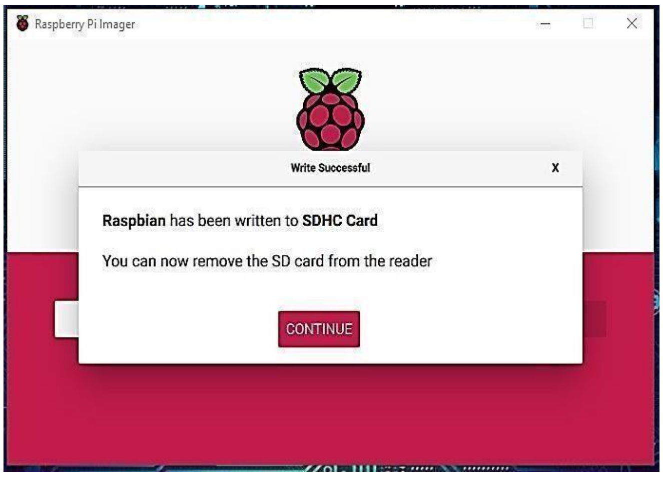
Raspbian
A port of Debian with the Raspberry Pi Desktop (Recommended)
Released: 2020-02-13
Online - 1.1 GB download

Raspbian (other)
Other Raspbian based images

LibreELEC
A Kodi Entertainment Center distribution

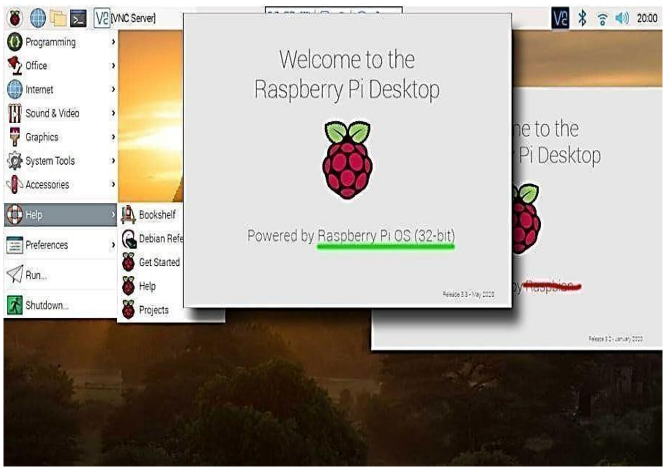
Misc utility images
EEPROM recovery, etc.

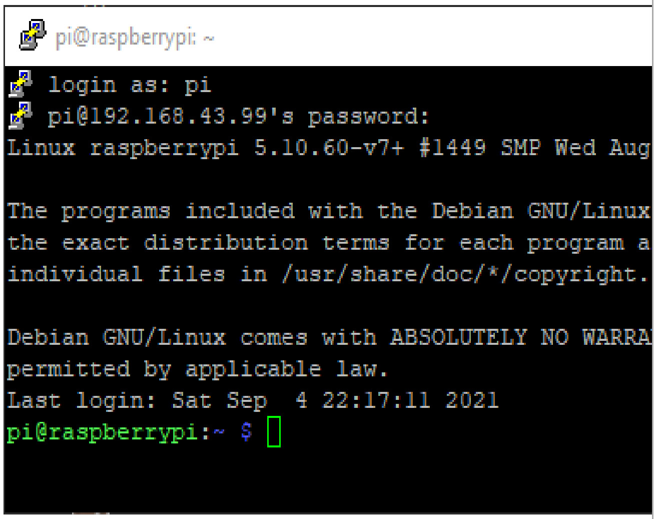
Erase
Format card as FAT32



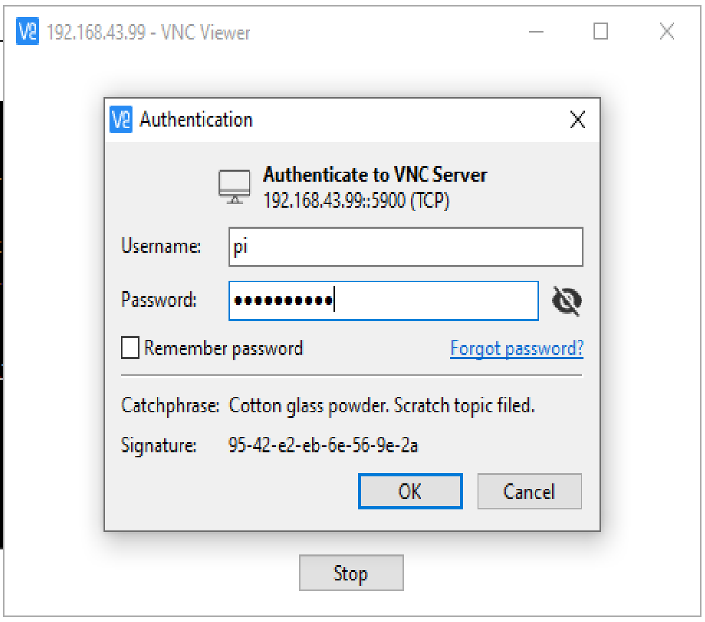
**Raspbian** has been written to **SDHC Card**

You can now remove the SD card from the reader





```
login as: pi
pi@192.168.43.99's password:
Linux raspberrypi 5.10.60-v7+ #1449 SMP Wed Aug

The programs included with the Debian GNU/Linux
the exact distribution terms for each program a
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRA
permitted by applicable law.
Last login: Sat Sep  4 22:17:11 2021
pi@raspberrypi:~ $
```



**Authenticate to VNC Server**
192.168.43.99::5900 (TCP)

Username: pi

Password: ••••••••••

☐ Remember password    Forgot password?

Catchphrase: Cotton glass powder. Scratch topic filed.

Signature: 95-42-e2-eb-6e-56-9e-2a

### III.3.2. Node-RED

Node-RED is a graphical programming language built on Node.js. It implements a server and runs what are called “Flows": programs based on JavaScript. Why would you want to run a server-side IDE for your programs? Because Node-RED also makes it dead simple to spin up web apps and use them as your online information and control system [16].

Node-RED is an open-source software platform for data computation coming from external hardware devices like PLC, Raspberry pi, Arduino’s, micro-controller via various communication protocols like OPC-UA, MQTT, Ethernet, API or MODBUS**.**

- It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

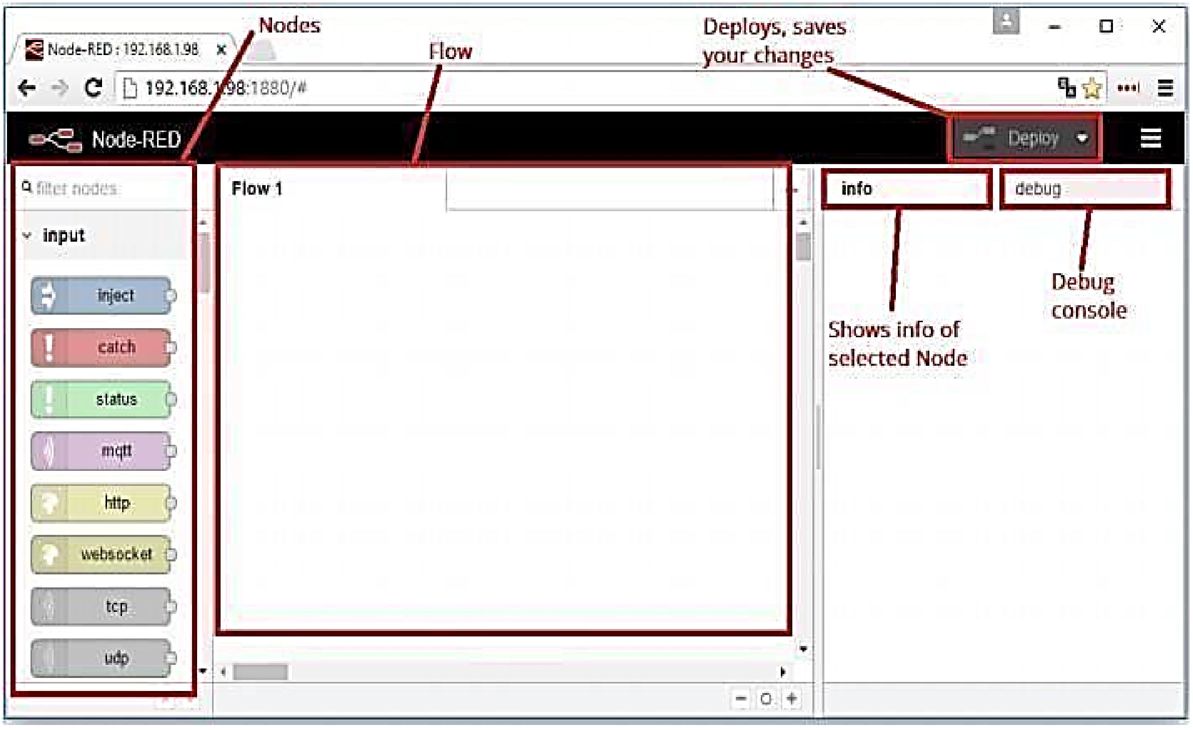
- The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

- The messages in NODE-RED are straightforward JavaScript objects that can have any arrangement of properties.

- Originally developed by IBM’s Emerging Technology Services team and now a part of the JS Foundation.

In short, the Node-RED is a simple “wiring” of nodes or predefined code blocks to perform the tasks. As we see below, are the main sections of the node-red interface, where on the left-side, we can see a list with a bunch of blocks. These blocks are called **nodes** and they are separated by their functionality. If we select a node, we can see how it works in the **info** tab. In the center, we have the **Flow** and this is where we place the nodes.

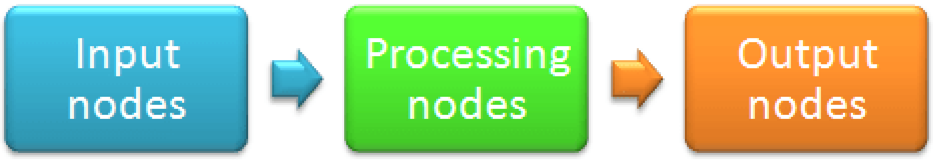
d



## ➢ The Connected Nodes and the Flow

The nodes when wired together are called **flows**.

The connected nodes consists of the input nodes, the processing nodes, and the output nodes.



**1- Input Nodes:** it allows you to enter the input data into the flow. They have a grey square on their right side which you can connect to the other nodes.By connecting this node to another node, you can send the input data. You can also connect services like Twitter, Google, serial, web sockets and send those data as inputs to the other nodes. Or else you can manually enter the input data using the node called "inject".

- ❖ **Inject Node:** This node permits manual activating of streams. It encourages us to infuse occasions at booked spans.
- ❖ Examples of **Input nodes:** inject, catch, status, Mqtt, etc.

**2- Output Nodes:** it allows you to send the data to the outside world, to other services for example twitter. It has a grey square on their left side which can be connected to either input node or function. If you don't want to send the data to any services like Twitter or email nodes, you can simply debug those messages by using the tab "deploy" on the top right corner of the editor.

- ❖ **Output nodes:** debug, link, Udp, Tcp, Web Socket, Http request, etc

**3- Processing Nodes:** the processing nodes are used to process the data in which they have one input endpoint and one or more output endpoints. They are used to transform the data types, to write the custom codes and to trigger a message.

- ❖ **Processing nodes:** comment, delay, function, switch, change, etc

## ➢ Running Node-Red

When the Node-RED establishment and introductory arrangement is finished, it can be used. Three ways to run NODE-RED:

✓ Locally

✓ On a device

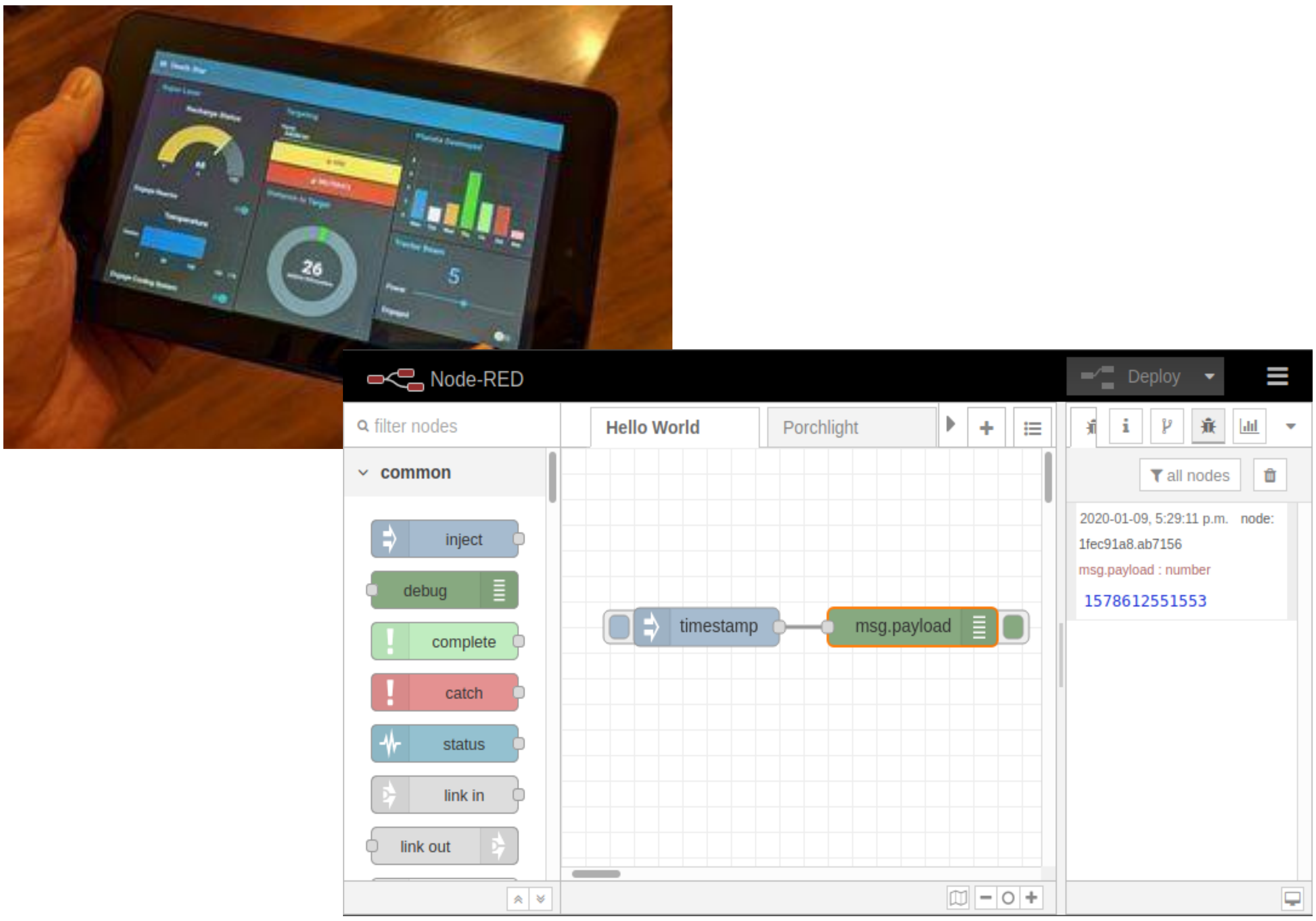
✓ In the identified cloud environment

## ➢ Few Use Cases

- Home Automation
- Dashboard creation and live data streaming to dashboards
- Connect multiple IOT devices

- Visual representation of wiring hardware, services and other components, And many more...

In order to access the Node-RED program, type in the browser the Raspberry Pi IP address followed by the port number 1880.



```
2020-01-09, 5:29:11 p.m.   node:
1fec91a8.ab7156
msg.payload : number
1578612551553
```

## ➢ Node-RED Dashboard

This plugin opens up a new level of web visualization and dashboard customization. The plugin gives the user a clean and easy-to-use platform of which the visualization features can be tailored to each and every individual's liking. The platform is hosted on the IQ Home gateway itself, so it is available across the network. With the plugin, the sensor data can be displayed as a line, bar, pie chart or use gauges.

## ➢ Node-RED makes it Easy to:

- Access RPi GPIOs
- Establish an MQTT connection with other boards (Arduino, ESP8266, etc.)
- Create a responsive graphical user interface for building projects
- Retrieve data from the web (weather forecast, stock prices, emails. etc.)
- Create time triggered events
- Store and retrieve data from a database

- **Installing Node-RED**

```
pi@raspberry:~ $ bash <(curl -sL https://raw.githubusercontent.com/nodered/raspbian•deb•package/mas
ter/resources/update-nodejs-and-node red)
```

- **Auto start Node-RED on boot**

```
pi@raspberry:~ $ sudo systemctl enable nodered.service
```

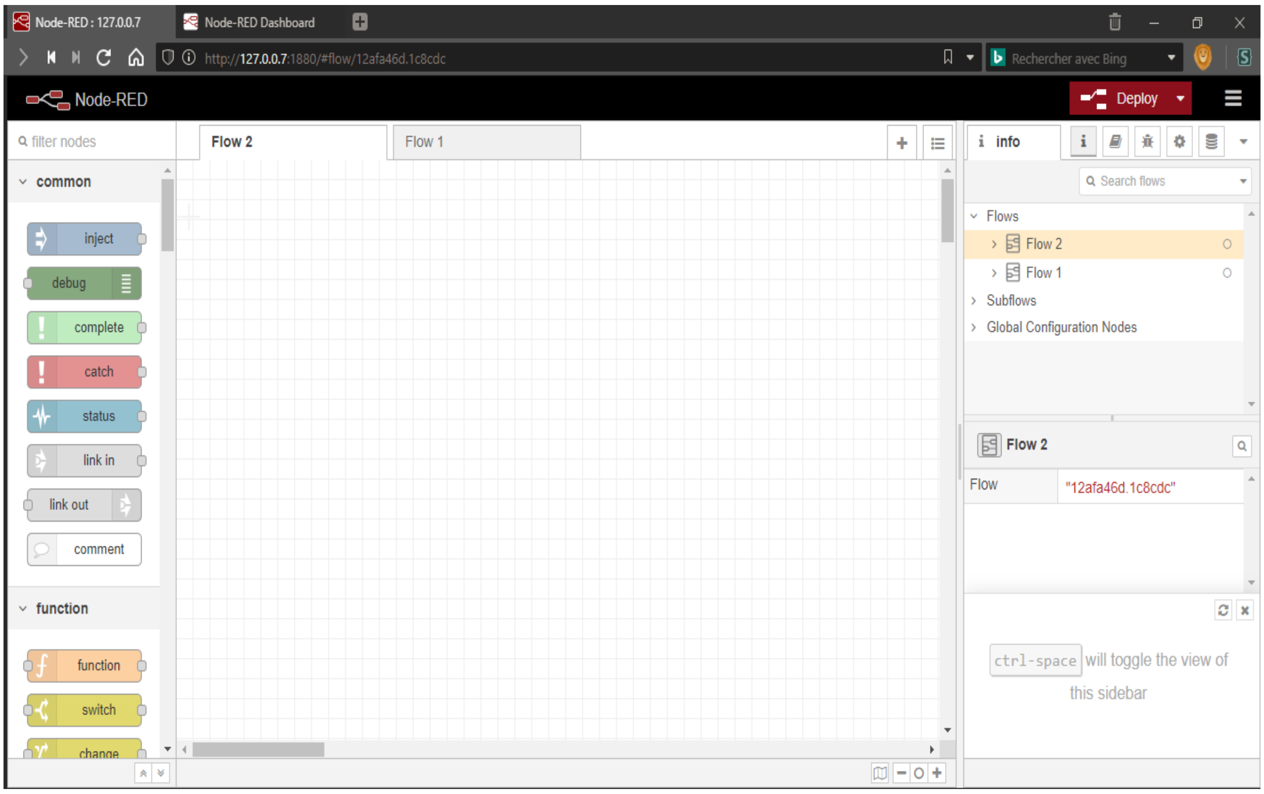
- **To restart Pi so the autos-tart takes effect:**

```
pi@raspberry:~ $ sudo reboot
```

- **To start working on node-red interface:**

When Pi reboot on, we can test the installation by entering the IP address of our raspberry Pi in the web browser followed by the **1880** port number:

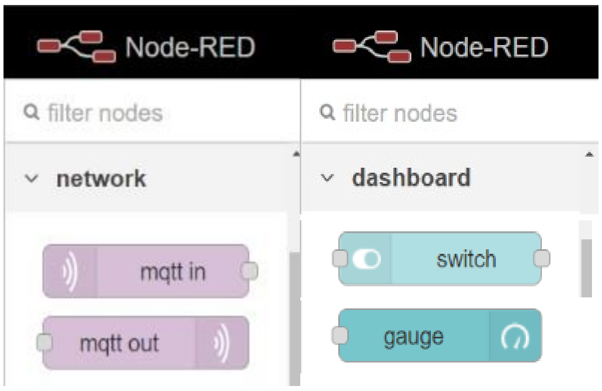
http://127.0.0.1:1880/ **or** http://192.168.43.99:1880/ A page like this appears:



**Note:** node red includes elements called 'nodes' used to interact with mosquitto server that it is running inside the raspberry.

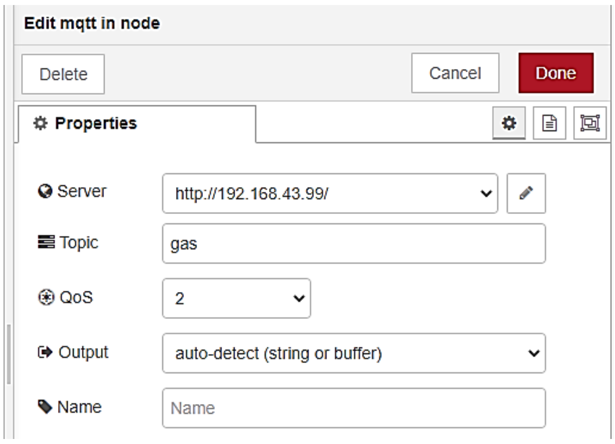
## - Node-red Nodes:

Node red nodes are arranged in specified fields each of them have its own functions, now we will use elements from network and dashboard nodes.



## - MQTT in:

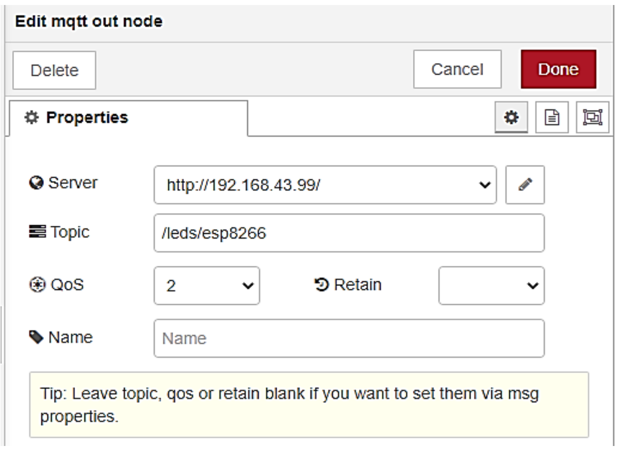
This element is the one who receives the message when it is published from the client with the specified topic, as mentioned in the figure, we must configure it to connect to our MQTT server and set the topic 'gas' and QoS.



Server: http://192.168.43.99/
Topic: gas
QoS: 2
Output: auto-detect (string or buffer)
Name: Name

## - MQTT out:

It is a non-visible element,it sends (publishes) the message to all clients subscribed to the specified topic, as mentioned in the figure, we must configure it to connect to our MQTT server and set the topic 'led' and QoS and Retain.

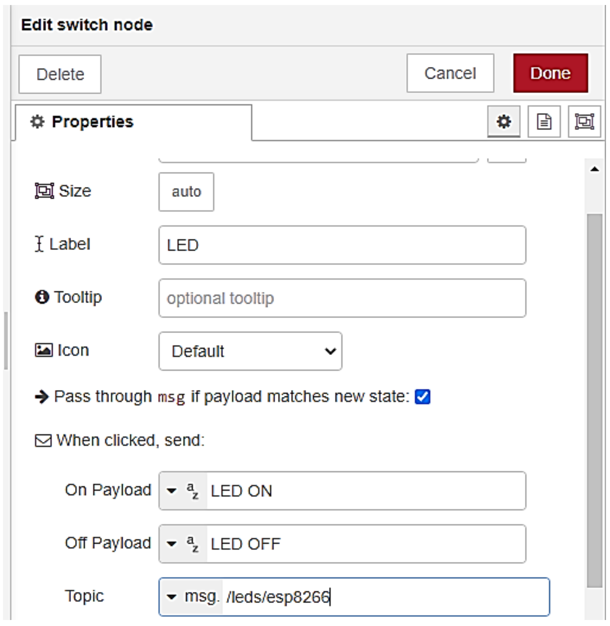


Server: http://192.168.43.99/
Topic: /leds/esp8266
QoS: 2
Retain:
Name: Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

## - Switch:

The switch element is the one who can send "on" and "off" commands, generally this element must be connected at least with one MQTT out element in order to send the configured 'On/ Off' payload.



Edit switch node

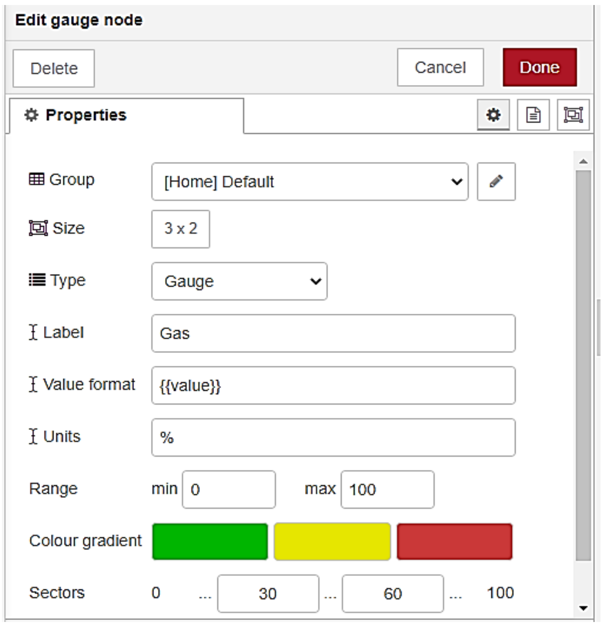
Delete | Cancel | Done

Properties

Size: auto
Label: LED
Tooltip: optional tooltip
Icon: Default
Pass through msg if payload matches new state: ✓
When clicked, send:
On Payload: LED ON
Off Payload: LED OFF
Topic: msg. /leds/esp8266

## - Gauge:

The gauge element is mostly used to monitor the received data therefore we must link it with MQTT in element in order to monitor the gas data



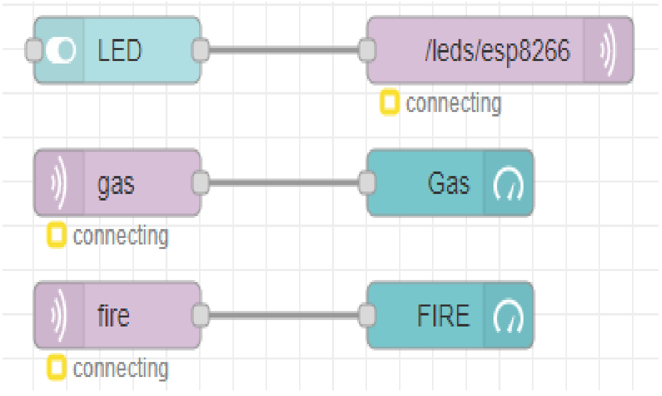
Edit gauge node

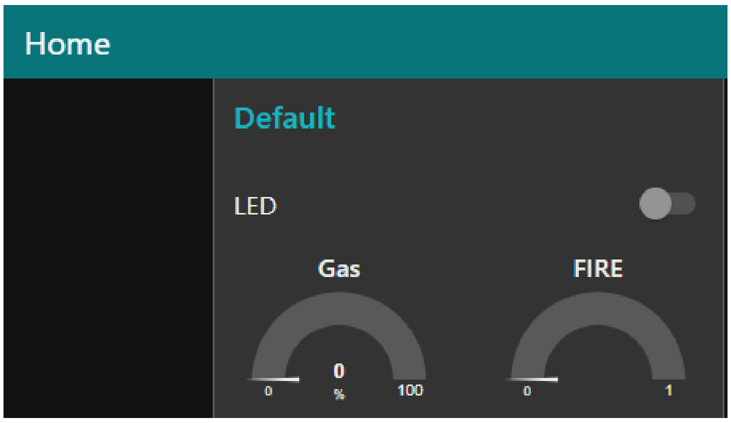
Delete | Cancel | Done

Properties

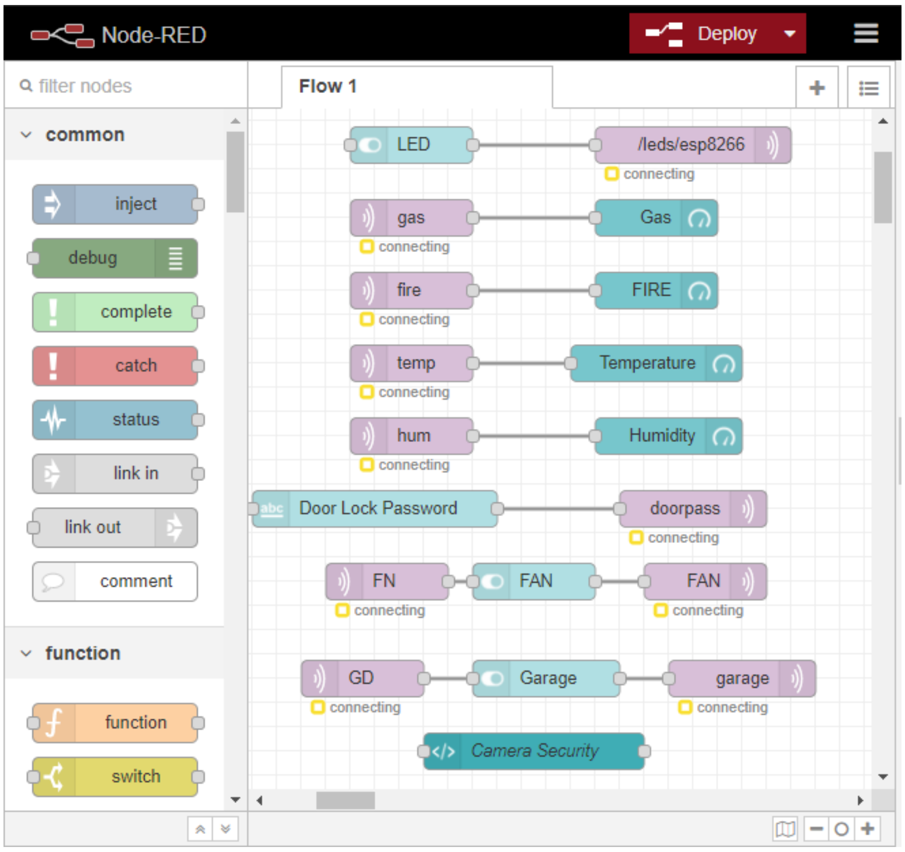
Group: [Home] Default
Size: 3 x 2
Type: Gauge
Label: Gas
Value format: {{value}}
Units: %
Range: min 0 max 100
Colour gradient: 🟩 🟨 🟥
Sectors: 0 ... 30 ... 60 ... 100

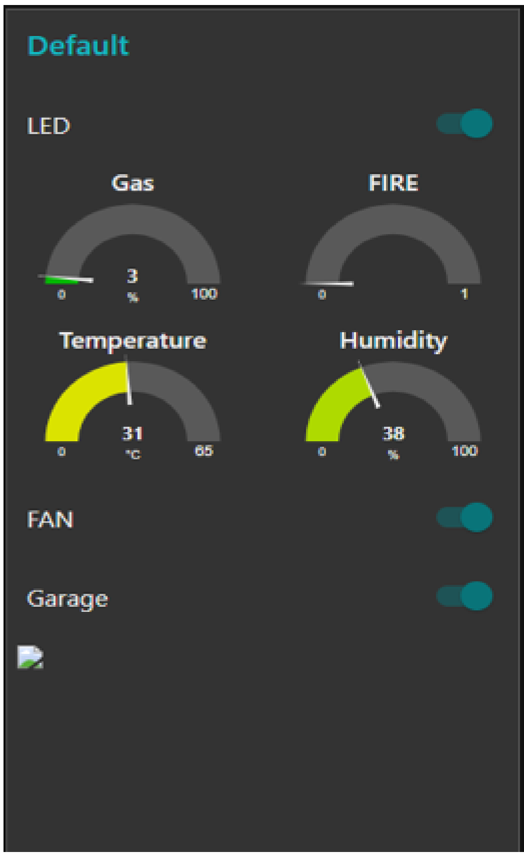
- After deployment we accessed to [http://127.0.0.1:1880/ui](http://127.0.0.1:1880/ui) and seen this result





We developed our final project dashboard using some of the previous elements and these are the results.





## III.3.3. Mosquitto Broker (Server)

Eclipse Mosquitto is an open source message broker that implements the MQTT protocol versions 3.1 and 3.1.1. It is one of the most famous MQTT brokers.
It's very easy to install and easy to use. It is lightweight and is suitable for use on all devices from low power single board computers to full servers. The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers. The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular mosquitto_pub and mosquitto_sub command line MQTT clients. It can be installed on UNIX machines. It can be secured via SSL and passwords.

### 1- Its importance

- After setting up mosquitto MQTT server on a Raspberry Pi. Then, it allows you to connect all DIY nodes to someone's very own MQTT broker.
- In MQTT, the broker is primarily responsible for receiving all messages, filtering the messages, deciding who is interested in it and then publishing the message to all subscribed clients.
- To communicate with the ESP8266 in WiFi (or from the Internet),the
- MQTT (Mosquitto broker) communication protocol is used.
- They receive the messages from the publishers and take care to properly maintain that information stream so that only publishers that have proper authentication can send messages.

The Broker then provides the data in “Topics” to the “Subscribers”. In the same vein, the Broker checks the authenticity of each “Subscriber” and only provides each Subscriber with the topics it is subscribed to.

- Mosquitto belongs to the Messaging Oriented Middleware (MOM) software category which supports sending and receiving of messages between distributed systems.

## 2- Server ports

The server listens on the following ports:

- 1883 : MQTT, unencrypted, unauthenticated (SSL)
- 1884 : MQTT, unencrypted, authenticated
- 8883 : MQTT, encrypted, unauthenticated (SSL)
- 8884 : MQTT, encrypted, client certificate required (SSL)
- 8885 : MQTT, encrypted, authenticated
- 8887 : MQTT, encrypted, server certificate deliberately expired
- 8080 : MQTT over WebSockets, unencrypted, unauthenticated
- 8081 : MQTT over WebSockets, encrypted, unauthenticated
- 8090 : MQTT over WebSockets, unencrypted, authenticated
- 8091 : MQTT over WebSockets, encrypted, authenticated

But the most common standard ports for Mqtt to use are TCP/IP port 1883 is reserved with IANA for use with MQTT. TCP/IP port 8883 is also registered, for using MQTT over SSL.
The authenticated listeners require a username / password:

- **rw** / **readwrite** : read/write access to the # topic hierarchy
- **ro** / **readonly** : read only access to the # topic hierarchy
- **wo** / **writeonly** : write only access to the # topic hierarch

## 3- Raspberry pi as MQTT Broker

In order to use the raspberry pi as MQTT server, we installed the mosquitto MQTT broker which is an open-source broker in raspberry pi. We can secure the broker by providing credentials such as username and password.

First, we install the Mosquitto server using.

```
sudo apt-get install mosquitto
```

After installing the MQTT server, install the client using command.

```
sudo apt-get install mosquitto-clients
```

Then check if it is running using command.

```
systemctl status mosquitto.service
```

Then run this command to modify binding configuration

```
sudo nano /etc/mosquitto/conf.d/mosquitto.conf
```

Modify the listener line including the raspberry static IP address and save the modification.

```
# listener port-number [ip address/host name/unix socket path]
listener 1883 192.168.1.88
```

After rebooting the raspberry, run this command to run the mosquito with custom configuration in the background.

```
mosquitto -c /etc/mosquitto/conf.d/mosquitto.conf &
```

We successfully installed the MQTT server and client.

- Commands to publish data from raspberry to the clients subscribed to the “led” topic example:

```
mosquitto_pub -h raspberrypi -t "led" -m "LED ON"
mosquitto_pub -h raspberrypi -t "led" -m "LED OFF"
```

- Where (-h) stands for: hostname, (-t) for topic and (-m) for message

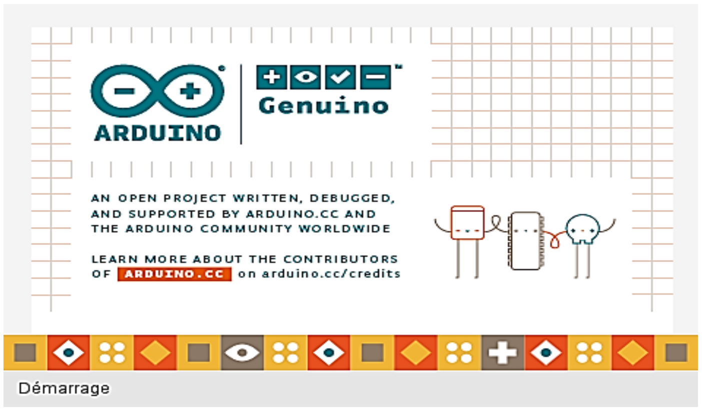
## III.3.4. Arduino IDE

The Arduino IDE (Integrated Development Environment) is where to develop programs that will tell the Arduino what to do. We can load new programs onto the main chip, the ATmega328p, via USB using the Arduino IDE. This development environment has a high number of libraries and active community, it also covers all registers configuration so it is less complicated [31].

Arduino IDE contributors integrated ESP-IDF compiler on their IDE; hence, most of ESP development board become compatible with Arduino IDE to program this microcontroller with C and C++ language. Though, it’s written in the Java programming language, and which represents the programming medium between the user and the board._This development environment has a high number of libraries and active community, it also covers all registers configuration so it is less complicated than ESP-IDF. To get started of Arduino IDE with ESP8266 boards, we referred to the source provided. We used The Arduino IDE in our project for:

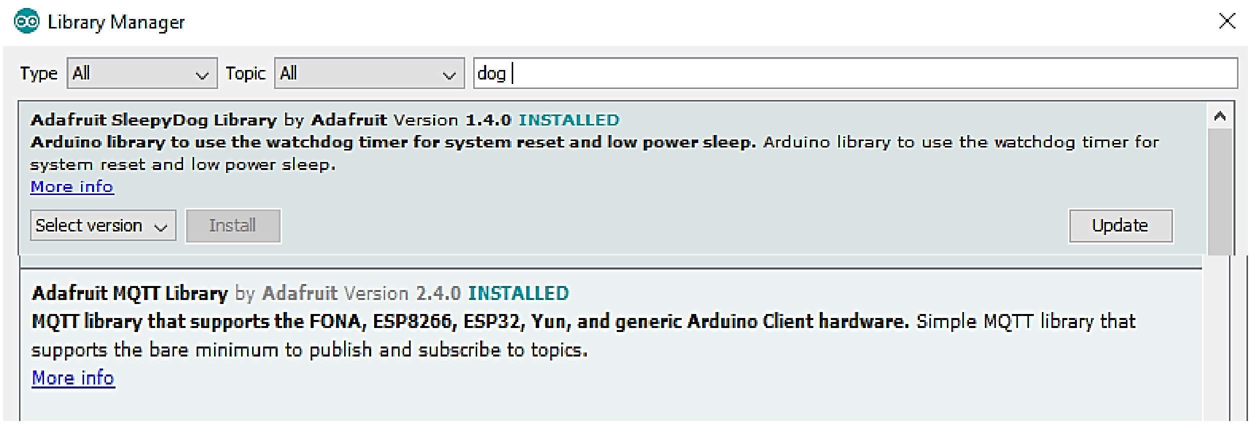
- Write and compile programs for the NodeMcu card and arduinos.

- Connect and Communicate with the NodeMcu board & arduino's to transfer the programs to it.


ARDUINO
Genuino
AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits
Démarrage

**NodeMCU Arduino Libraries**

For the ESP8266 to interact with the MQTT server, we also must install **sleepydog** library to use the watchdog timer for system reset and low power sleep, and **mqtt** library by Adafruit .This library allow ESP8266 to talk with node-red over Raspberry pi [30].

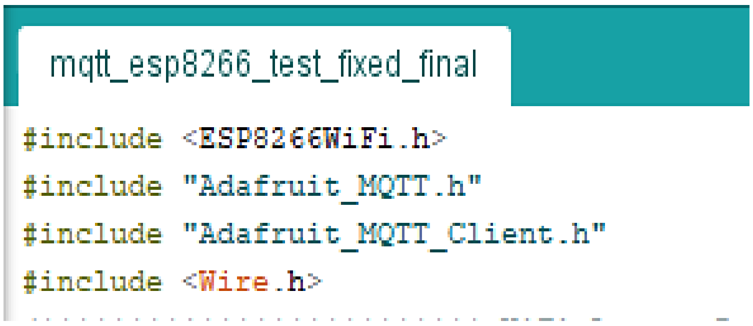


**Adafruit SleepyDog Library** by **Adafruit** Version **1.4.0** INSTALLED
**Arduino library to use the watchdog timer for system reset and low power sleep.** Arduino library to use the watchdog timer for system reset and low power sleep.
More info

**Adafruit MQTT Library** by Adafruit Version 2.4.0 INSTALLED
**MQTT library that supports the FONA, ESP8266, ESP32, Yun, and generic Arduino Client hardware.** Simple MQTT library that supports the bare minimum to publish and subscribe to topics.
More info

After installing these two libraries, to program the NodeMCU as subscriber or publisher to topics we must follow these programming steps.

- Declare the used Libraries.



```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <Wire.h>
```

- Declare WLAN and MQTT server Credentials.

```
/************************* WiFi Access Point ****************
#define WLAN_SSID "shadowBroker"
#define WLAN_PASS "@llasy!@"
#define MQTT_SERVER "192.168.43.99" // give static address
#define MQTT_PORT 1883
#define MQTT_USERNAME ""
#define MQTT_PASSWORD ""
```

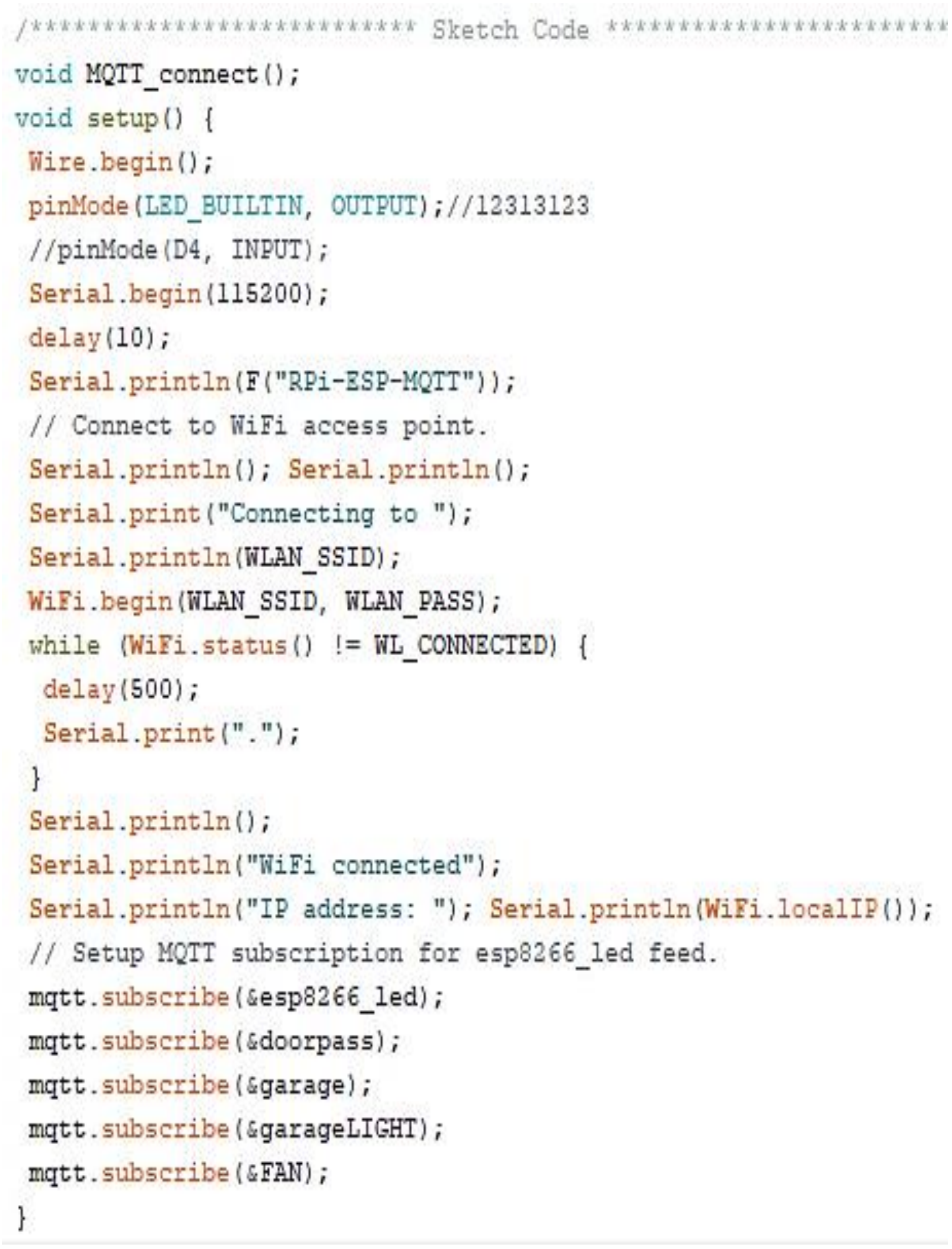
- Create Wi-Fi client and MQTT client objects

```
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// Setup the MQTT client class by passing in the WiFi client and MQTT server and login det
Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, MQTT_PORT, MQTT_USERNAME, MQTT_PASSWORD);
```

- Create subscribe and publish objects defined with their topics

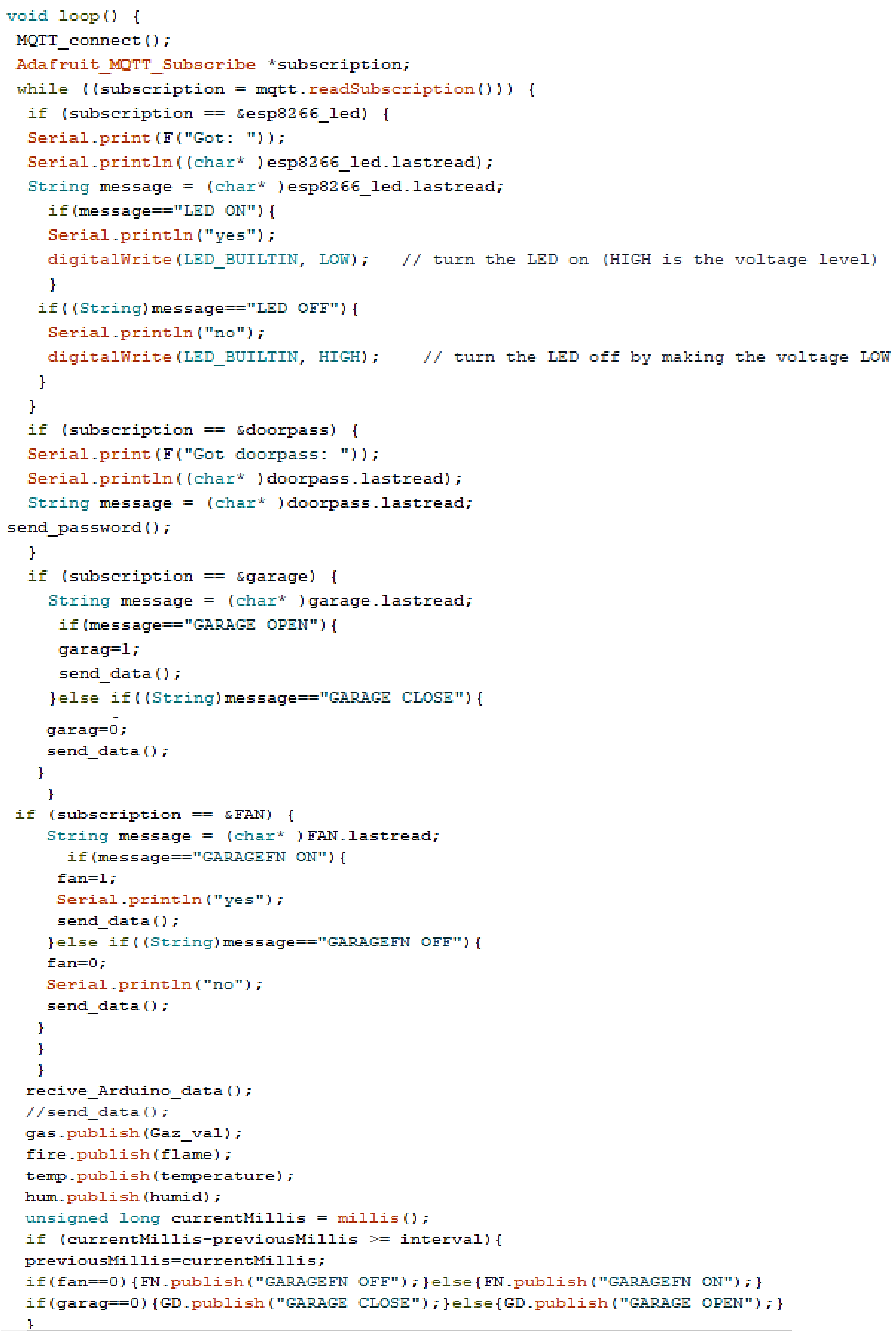
```
Adafruit_MQTT_Publish gas = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "gas");
Adafruit_MQTT_Publish fire = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "fire");
Adafruit_MQTT_Publish temp = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "temp");
Adafruit_MQTT_Publish hum = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "hum");
Adafruit_MQTT_Publish GL = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "GL");
Adafruit_MQTT_Publish GD = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "GD");
Adafruit_MQTT_Publish FN = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "FN");
// Setup a feed called 'esp8266_led' for subscribing to changes.
Adafruit_MQTT_Subscribe esp8266_led = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME "/leds/esp8266");
Adafruit_MQTT_Subscribe doorpass = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME "doorpass");
Adafruit_MQTT_Subscribe garage = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME "garage");
Adafruit_MQTT_Subscribe garageLIGHT = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME "garageLIGHT");
Adafruit_MQTT_Subscribe FAN = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME "FAN");
```

- In void setup, connect to Wi-Fi and subscribe to the declared subscribe object



```
/*************************** Sketch Code ************************
void MQTT_connect();
void setup() {
 Wire.begin();
 pinMode(LED_BUILTIN, OUTPUT);//12313123
 //pinMode(D4, INPUT);
 Serial.begin(115200);
 delay(10);
 Serial.println(F("RPi-ESP-MQTT"));
 // Connect to WiFi access point.
 Serial.println(); Serial.println();
 Serial.print("Connecting to ");
 Serial.println(WLAN_SSID);
 WiFi.begin(WLAN_SSID, WLAN_PASS);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
 Serial.println();
 Serial.println("WiFi connected");
 Serial.println("IP address: "); Serial.println(WiFi.localIP());
 // Setup MQTT subscription for esp8266_led feed.
 mqtt.subscribe(&esp8266_led);
 mqtt.subscribe(&doorpass);
 mqtt.subscribe(&garage);
 mqtt.subscribe(&garageLIGHT);
 mqtt.subscribe(&FAN);
}
```

- In void loop create subscription pointer to get if there is new data transferred to specified topic then do operation depending on the message , to publish gas value data simply do the instruction **gas.publish(Gaz_value);**



```
void loop() {
 MQTT_connect();
 Adafruit_MQTT_Subscribe *subscription;
 while ((subscription = mqtt.readSubscription())) {
  if (subscription == &esp8266_led) {
  Serial.print(F("Got: "));
  Serial.println((char* )esp8266_led.lastread);
  String message = (char* )esp8266_led.lastread;
    if(message=="LED ON"){
    Serial.println("yes");
    digitalWrite(LED_BUILTIN, LOW);   // turn the LED on (HIGH is the voltage level)
    }
   if((String)message=="LED OFF"){
    Serial.println("no");
    digitalWrite(LED_BUILTIN, HIGH);    // turn the LED off by making the voltage LOW
   }
  }
  if (subscription == &doorpass) {
  Serial.print(F("Got doorpass: "));
  Serial.println((char* )doorpass.lastread);
  String message = (char* )doorpass.lastread;
send_password();
  }
  if (subscription == &garage) {
    String message = (char* )garage.lastread;
     if(message=="GARAGE OPEN"){
     garag=1;
     send_data();
    }else if((String)message=="GARAGE CLOSE"){
    garag=0;
    send_data();
   }
    }
 if (subscription == &FAN) {
    String message = (char* )FAN.lastread;
      if(message=="GARAGEFN ON"){
     fan=1;
     Serial.println("yes");
     send_data();
    }else if((String)message=="GARAGEFN OFF"){
    fan=0;
    Serial.println("no");
    send_data();
   }
   }
   }
  recive_Arduino_data();
  //send_data();
  gas.publish(Gaz_val);
  fire.publish(flame);
  temp.publish(temperature);
  hum.publish(humid);
  unsigned long currentMillis = millis();
  if (currentMillis-previousMillis >= interval){
  previousMillis=currentMillis;
  if(fan==0){FN.publish("GARAGEFN OFF");}else{FN.publish("GARAGEFN ON");}
  if(garag==0){GD.publish("GARAGE CLOSE");}else{GD.publish("GARAGE OPEN");}
  }
```

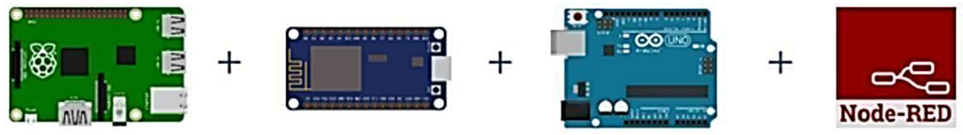
### III.3.5. SSH (Secure Shell)

SSH (which stands for secure shell) is a method of establishing a communication with another computer securely. All data sent via SSH is encrypted.

- Is based on a Unix shell
- A method of establishing a communication with another computer securely.
- All data sent via SSH is encrypted.so it allows access to Raspberry Pi files from a remote machine by using terminal commands. It has grown to be one of the most popular methods for communication between different devices.
- After installing it, an SSH communication will be established with Raspberry Pi, (Putty software needs to be installed to run ssh for windows users).
- PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

## III.4. Conclusion

The connection between Raspberry pi, Node-red, Arduinos and MQTT in short:



- The Node-RED software is running on a Raspberry Pi in order to create the interface where the communication between the ESP8266, and Arduino's interaction is achieved via the MQTT protocol which is responsible for data exchange.
- Node Red receives data from ESP8266 and allow to control arduino's connected to it from a dashboard created in Node Red Dashboard.
- Arduino IDE code performs several steps but define major broker connection with MQTT, function post (send messages) even specific topic and subscribe function (receive messages) even specific topic.
- Node client receiving messages: Mqtt (subscribe) from Esp8266 through the topic "event" node debugging and debugging. Example, sending node values MQTT (publish) which are "On" "Off" to ESP8266 topic "event" to to display LED status.
- At each change Node-RED publishes an MQTT message. For example, servo / pan, 92.
- The message is transmitted over the WiFi local area network toESP8266.
- After decoding the message, the respective servo-motor is moved to the required angular position.
- We can also control the articulated system from the Internet browser of a tablet or a smartphone.
- We will have a home automation application running on our RPi that allows us to monitor and control various devices in the house.
- We will access our application from any web browser: whether it's using a laptop, a tablet or a smartphone. It will be accessible from anywhere.

**Example:**

✔ controlling an output by sending a command, Lamp example
✔ reading data from a sensor and publish it, Sensor example

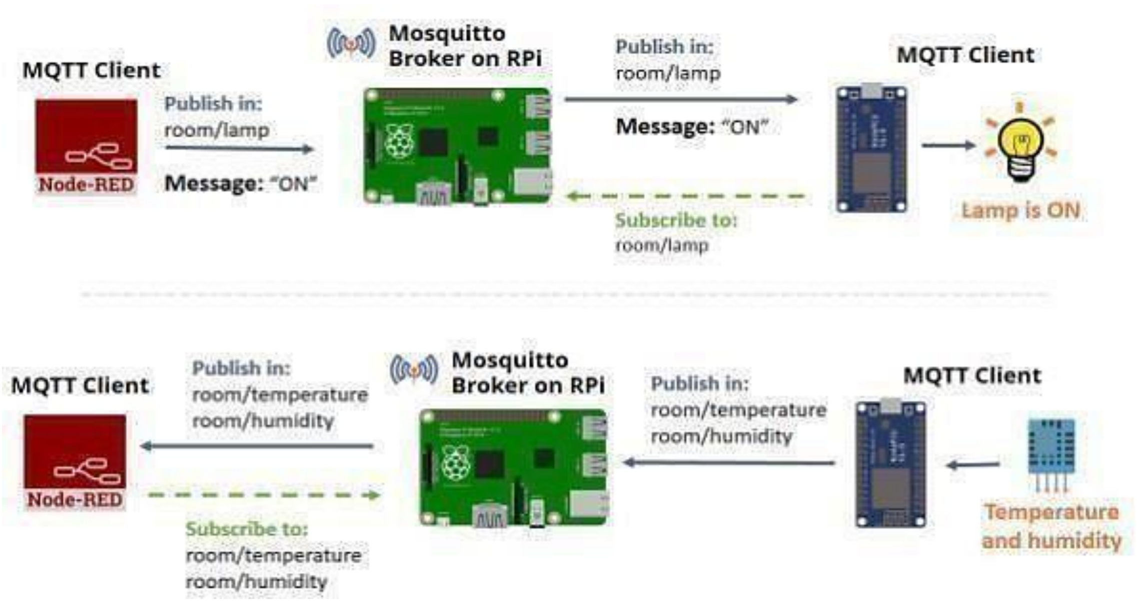


Figure III.2 Controlling light by sending 'on' command & reading data from dht11 sensor [17]

At the beginning of this chapter, we started with a general description of the system hardware to be implemented, also we gave a description, plus the characteristics and installation process of the software used. In addition, we've set an example of how things communicate illustrated in figure 1 above.

Next, we will look into, let's call "what is behind the scenes", the i2c concept that we are going to discuss now in chapter 4. Besides discussing project's result, their flowchart and circuits diagrams for better illustration of the system.

# CHAPTER IV: PROJECT STRUCTURE AND RESULTS DISCUSSION

# CHAPTER IV: PROJECT STRUCTURE AND RESULT DISCUSSION

## IV.1. Introduction

After we have talked in previous chapters about the software and hardware used and how the interaction between the both was done, using Raspberry Pi as the main server within two other important servers: node red and MQTT broker. We may summarize each role as fellow:

- Node-red is like the web viewer, where we view the reported information of our controlled devices thanks to the node-red dashboard in the goal of triggering orders quickly
- The mosquito server to establish a communication between things so they can talk and this interaction is done via MQTT protocol over Wi-Fi.

In this Chapter, we are going to discuss and see other different technologies seen named I2C. Where the esp8266 is going to play the role of the master that control the slaves, which in our case are the arduino's by giving them certain orders to do specific tasks, like opening/closing garage, turning lights on/off, controlling the house temperature, receiving notifications in case of detecting intruders and so on…, and this idea of the master slave communication is done via the I2C which help us to minimize the prototype connection between the components.

Overall, we are going to overview the concept of the I2C communication, how it works, its features, the concept of the slave and master interaction, how we are going to connect these, how the esp8266 is going to control the other arduino's. It's basically, like the CEO of home automation company, is the Raspberry Pi "The boss" who everything works under, Then the vice president ESP8266 who take the head orders to be then given to the workers (the arduino's) to complete the tasks, then reviewed again by esp8266 to check if each one is doing his job as it should be, to report it back to the Raspberry pi. Where everything is well organized. Then, we will present the process of each project's part in the form of flowchart.

The following synoptic diagram will allow us to better understand the overall functioning of the studied system:

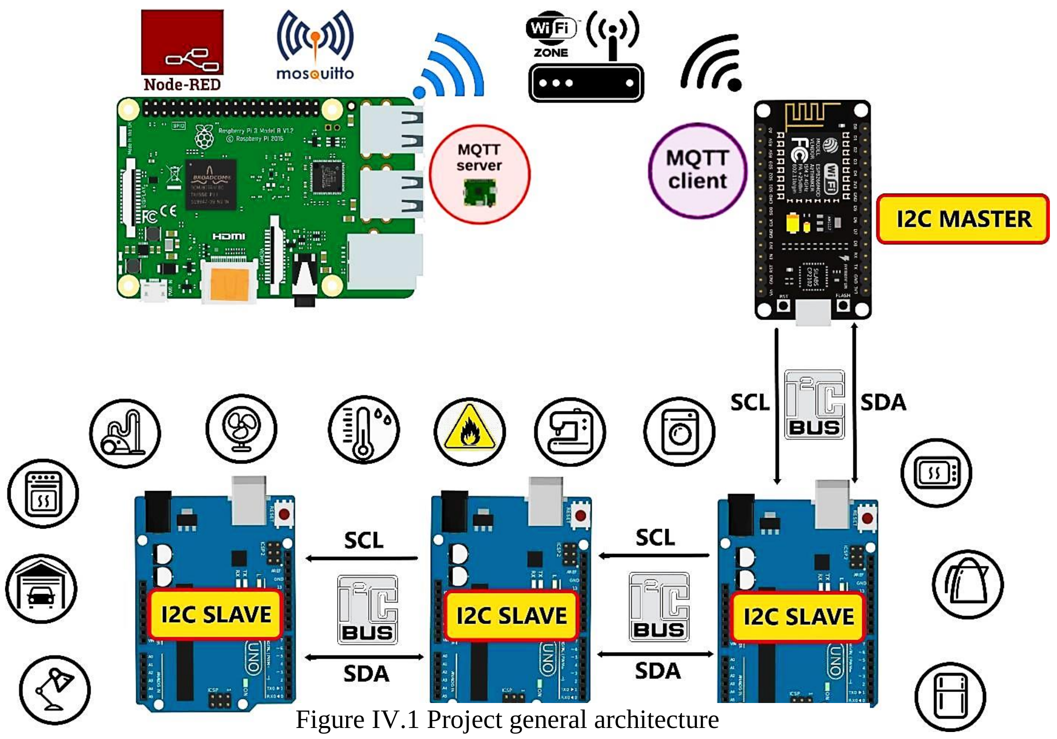


Figure IV.1 Project general architecture

Our project architecture is based on MQTT Broker (server) with one client (subscriber / publisher), as mentioned in the fig1, the raspberry plays the role of MQTT broker (mosquitto) and user interface (Nodered dashboard). The raspberry is connected to the WLAN network and defined by its IP which means that the services are accessible only in local Network therefore the MQTT client (esp8266) must be connected to the same network in order to subscribe and publish data using defined topics. In our project we use our client to subscribe to topics that are related to the things we want to control, in addition, we publish data with the topics that are related to sensors values and status. Moreover, our MQTT client (esp8266) is an I2C Master used to control the connected Arduino's (slaves) that are responsible to interact with appliances.as result, the Arduino's will process the sensor values and appliances status and exchange that data with ESP8266 in order to send and synchronize that data with server.

## IV.2. I2C (inter-integrated circuits)

Like SPI, I2C is a synchronous serial communication protocol. It means that data bits are transferred one by one at regular intervals of time along a single wire (the SDA line), and the output of bits is synchronized to the sampling of bits by a clock signal (the SLC line), shared between the master and the slave [18].

➢ **Important Features**

- Only two common bus lines (wires) are required to control any device/IC on the I2C network
- No need for prior agreement on data transfer rate like in UART communication. So the data transfer speed can be adjusted whenever required
- Simple mechanism for validation of data transferred
- Uses 7-bit addressing system to target a specific device/IC on the I2C bus
- I2C networks are easy to scale. New devices can simply be connected to the two common I2C bus lines

➢ **Good reasons to use**

- Only uses two wires
- Supports multiple masters and multiple slaves
- ACK/NACK bit gives confirmation that each frame is transferred successfully
- Hardware is less complicated than with UARTs
- Well known and widely used protocol

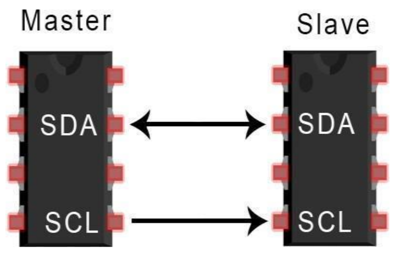
➢ **Limitation**

- Slower data transfer rate than SPI
- The size of the data frame is limited to 8 bits
- More complicated hardware needed to implement than SPI

## The physical I2C Bus

I2C Bus (Interface wires) consists of just two wires and are named as Serial Clock Line (SCL) and Serial Data Line (SDA). The data to be transferred is sent through the SDA wire and is synchronized with the clock signal from SCL.

- **SDA (Serial Data)**: The line for the master and slave to send and receive data.
- **SCL (Serial Clock)** : The line that carries the clock signal, and it is always controlled by the master.



All the devices/ICs on the I2C network are connected to the same SCL and SDA lines as shown below:

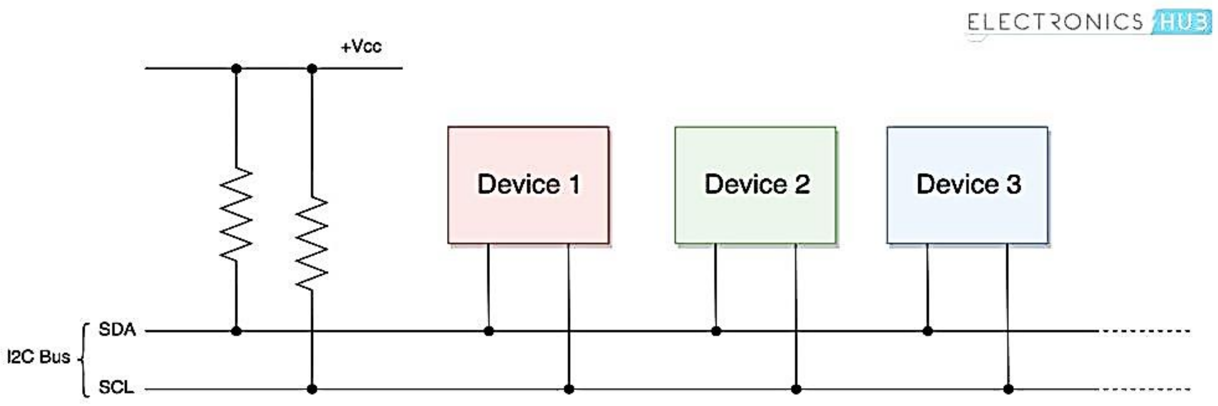


Figure IV.2 Master Slave wiring through i2c bus lines (sda & scl)

## IV.2.1. Master and Slave Devices

### 1. Master-Slave

The devices connected to the I2C bus are categorized as either masters or slaves. At any instant of time only a single master stays active on the I2C bus. It controls the SCL clock line and decides what operation is to be done on the SDA data line.

All the devices that respond to instructions from this master device are slaves. For differentiating between multiple slave devices connected to the same I2C bus, each slave device is physically assigned a permanent 7-bit address when a master device wants to transfer data to or from a slave device, it specifies this particular slave device address on the SDA line and then proceeds with the transfer. So effectively communication takes place between the master device and a particular slave device [20]. For example:

❖ **The Master- ESP8266:** This board is responsible for asking the slave for all the information like the temperature, the door status, the light or the gas level. Other functions of this board is to post the data on the web or read data from the web because this board has Wi-Fi built-in on it.

On startup, this board reads the Wi-Fi connection information from the microSD card and use it to connect to Adafruit and establish a bidirectional MQTT connection; reading and writing using MQTT

❖ **The Slaves -the Arduino's Uno:** The program inside this micro controller runs many STATE machines to get information about all the sensors, calculate and maintain floating averages of the temperature, the light, reading from Gas, fire sensors and store all the values in variables ready to be sent to the master when needed.

As a slave, it is getting ready to answer the requests from the masters such as the value for the door state or the temperature in the bedroom.

**2. Single Master with Multiple Slaves**

Because I2C uses addressing, multiple slaves can be controlled from a single master. With a 7 bit address, 128 ($2^7$) unique addresses are available. Using 10 bit addresses is uncommon, but provides 1,024 ($2^{10}$) unique addresses. To connect multiple slaves to a single master, wire them like this, with 4.7K Ohm pull-up resistors connecting the SDA and SCL lines to Vcc.

**3. Multiple Masters with Multiple Slaves**

Multiple masters can be connected to a single slave or multiple slaves. The problem with multiple masters in the same system comes when two masters try to send or receive data at the same time over the SDA line. To solve this problem, each master needs to detect if the SDA line is low or high before transmitting a message. If the SDA line is low, this means that another master has control of the bus, and the master should wait to send the message. If the SDA line is high, then it's safe to transmit the message. To connect multiple masters to multiple slaves, use the following diagram, with 4.7K Ohm pull-up resistors connecting the SDA and SCL lines to Vcc.

All the other slave devices don't respond unless their address is specified by the master device on the SDA line

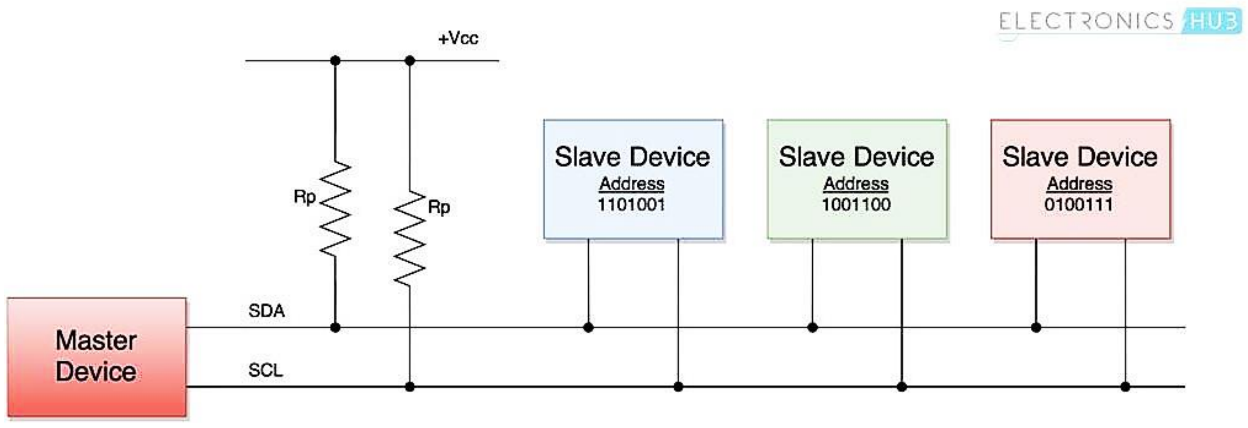


Figure IV.3 Slave recognition by master through the 7 bit address

In short, Data is transferred between the master device and slave devices through a single SDA data line, via patterned sequences of 0's and 1's (bits). Each sequence of 0's and 1's is termed as a transaction and the data in each transaction is structured as below [18]:

### IV.2.2. How I2C Works

With I2C, transactions are initiated by a master device either to send data to a slave device or to receive data from it [20].

Data is transferred in *messages.* Messages are broken up into *frames* of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame

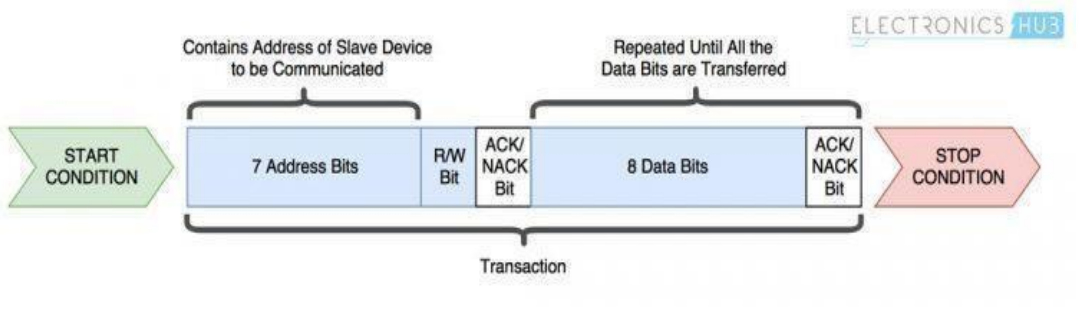


Figure IV.4 Master slave data transfer concept

1. **Start Condition**

Whenever a master device/IC decides to start a transaction, it switches the SDA line from high voltage level to a low voltage level before the SCL line switches from high to low. Once a start condition is sent by the master device, all the slave devices get active even if they are in sleep mode, and wait for the address bits

2. **Address Block**

To let the slave know that data is being sent to it, and not another slave. It does this by addressing. Each slave device consists of a frame of 7 unique sequence bits to/from which the master device needs to send/receive data. The master sends the address of the slave it wants to communicate with to every slave connected to it.
Each slave then compares the address sent from the master to its own address. If the address matches, it sends a low voltage ACK bit back to the master. If the address doesn't match, the slave does nothing and the SDA line remains high.

All the slave devices on the I2C bus compare these address bits with their address. In short, it identifies the slave when the master wants to talk to it.

3. **Read/Write Bit**

A single bit at the end of the frame specifying the direction of data transfer, which informs the slave whether the master wants to write data to it or receive data from it. If the master device/IC need to send data to a slave device, this bit is set to '0' (low voltage level). If the master needs to receive (request) data from the slave, this bit is set to '1' (High voltage level).

4. **ACK/NACK Bit**

Each frame in a message is followed by an Acknowledged (data frame was successfully received, an ACK bit '0' is returned to the sender from the receiving device.)/ Not-Acknowledged bit. If the physical address of any slave device coincides with the address broadcasted by the master device, the value of this bit is set to '0' by the slave device. Otherwise it remains at logic '1' (default).

**5. Data Block**

After the master detects the ACK bit from the slave, the first data frame is ready to be sent. The data frame is always 8 bits long set by the sender. Each data frame is immediately followed by an ACK/NACK bit to verify that the frame has been received successfully. Set to '0' by the receiver if it successfully receives data.
Otherwise it stays at logic '1'. The ACK bit must be received by either the master or the slave (depending on who is sending the data) before the next data frame can be sent. This combination of data block followed by ACK/NACK bit is repeated until the data is completely transferred

**6. Stop Condition**
After all of the data frames have been sent, the master can send a stop condition to the slave to halt the transmission. The stop condition is a voltage transition from low to high on the SDA line after a low to high transition on the SCL line, with the SCL line remaining high.

## In short:

1. When a master device tries to send data to a particular slave device through the I2C bus, it sends the start condition to every connected slave.

2. The master sends each slave the 7 bit address of the slave it wants which corresponds to the slave device to be targeted, along with the read/write bit '0', which signifies a write.

3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the SDA line low for one bit(ACK/NACK bit to'0', which signals the master device that a slave device is successfully targeted). If the address from the master does not match the slave's own address, the slave leaves the SDA line high (ACK/NACK bit stays at '1' (default). i.e, this signals the master device that the slave device identification is unsuccessful.).

4. If a slave device is successfully targeted, it will receive 8 bits of data by the master (which present the order), then the slave will set the ACK/NACK bit to '0', which signals the master device to continue. Other remaining slaves are not concerned with what this data means.

5. The master sends or receives the data frame.

6. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame.

7. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high.

### IV.2.2.1. Concept of clock stretching (SCL)

Let say the master device started a transaction and sent address bits of a particular slave device followed by a Read bit of '1'. The specific slave device needs to send an ACK bit, immediately followed by data [18].

But if the slave device needs some time to fetch and send data to master device, during this gap, the master device will think that the slave device is sending some data.

To prevent this, the slave device holds the SCL clock line low until it is ready to transfer data bits. By doing this, the slave device signals the master device to wait for data bits until the clock line is released.

### IV.2.3. I2C Wiring of Arduinos to ESP8266

This is the representation of our project slave/master interaction:

- Connect the grounds of Arduinos together.
- Then connect both A5 (SCL) analog pins together.
- Connect both A4 (SDA) analog pins together. As shown in this figure.

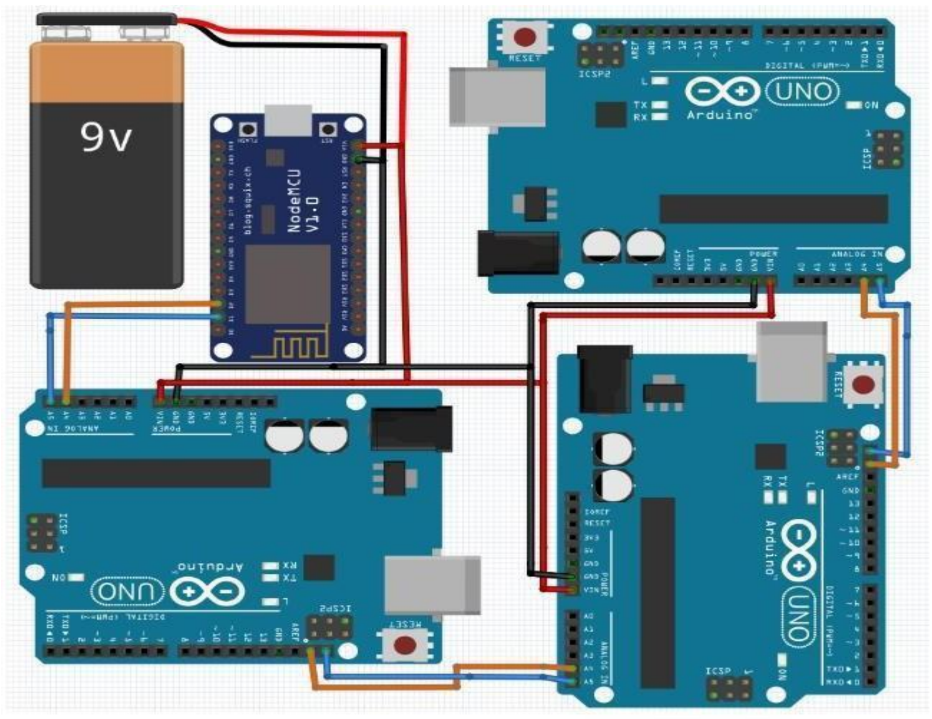


Figure IV.5 Master & Slave Wiring

## IV.3. Project's Flowchart & Circuits Diagram

To build the project, we went through many steps, tips, and searches. Hence, we are going to discuss in this section the main outcomes of our project. Furthermore, we will highlight the main results and functionalities through these flowcharts, we will see each part of the following:

- RFID door lock (scanning tag/card, entering password)
- sensing device (fire, gas, temperature & humidity)
- Garage door (schematic - how it works)

## IV.3.1. Project Research

This Organization chart explains the general idea's plan of our working system, the steps we followed in order to keep the project organized.

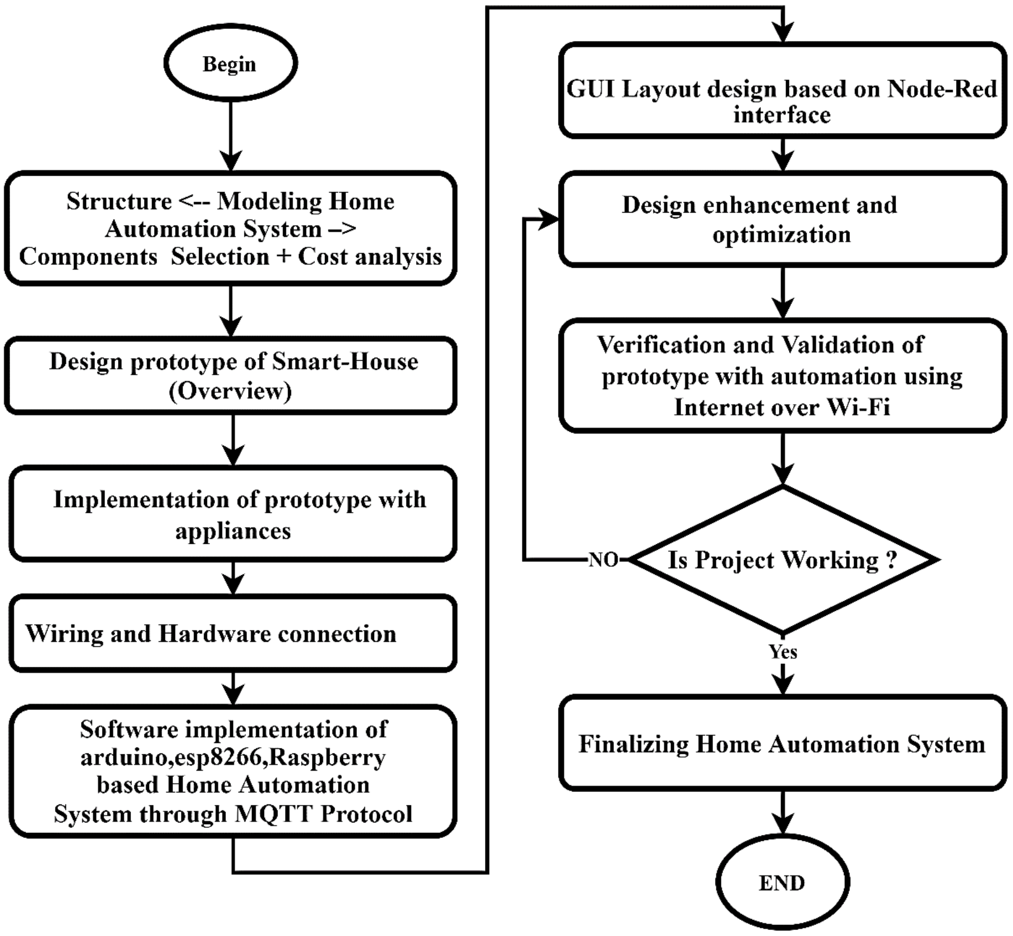

Begin
Structure <-- Modeling Home Automation System –> Components Selection + Cost analysis
Design prototype of Smart-House (Overview)
Implementation of prototype with appliances
Wiring and Hardware connection
Software implementation of arduino,esp8266,Raspberry based Home Automation System through MQTT Protocol
GUI Layout design based on Node-Red interface
Design enhancement and optimization
Verification and Validation of prototype with automation using Internet over Wi-Fi
Is Project Working ?
NO
Yes
Finalizing Home Automation System
END

Figure IV.6 Flowchart: Project Research

## IV.3.2. Raspberry Project

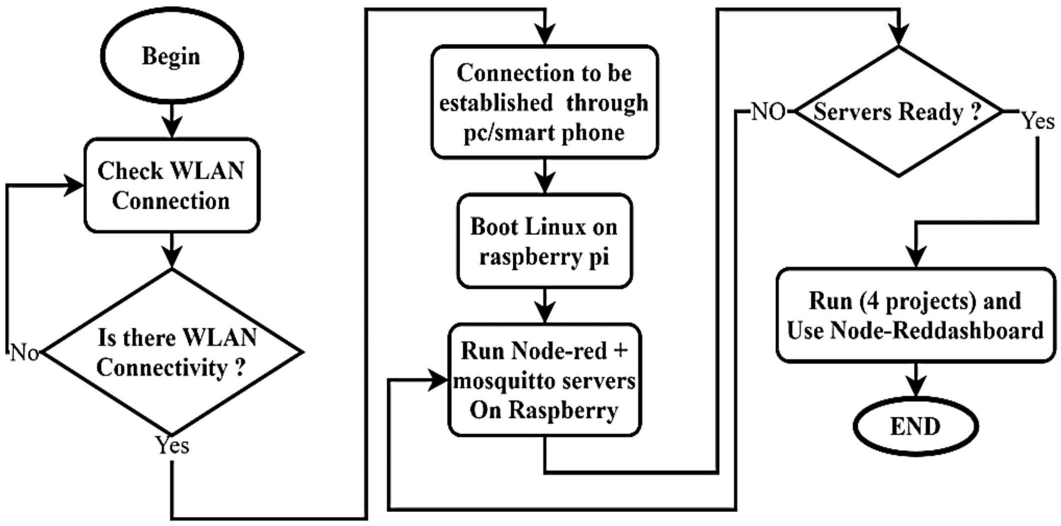

Begin
Check WLAN Connection
Is there WLAN Connectivity ?
No
Yes
Connection to be established through pc/smart phone
Boot Linux on raspberry pi
Run Node-red + mosquitto servers On Raspberry
Servers Ready ?
NO
Yes
Run (4 projects) and Use Node-Reddashboard
END

Figure IV.7 Flowchart: Raspberry Project

The Raspberry Pi organization chart explains how we used the raspberry pi. Before we use it, we make sure that our Wi-Fi is within range. We control the house using a node red interface control panel, which allows us to monitor all of the appliances. MQTT broker, on the other hand, is in charge of house communication via MQTT messaging protocol over Wi-Fi.

### IV.3.3. ESP8266 Project

The following ESP8266 flowchart depicts how NodeMcu reads the program and executes it, keeping track of all the sensors he is controlling by issuing orders so that each sensor can complete his task.

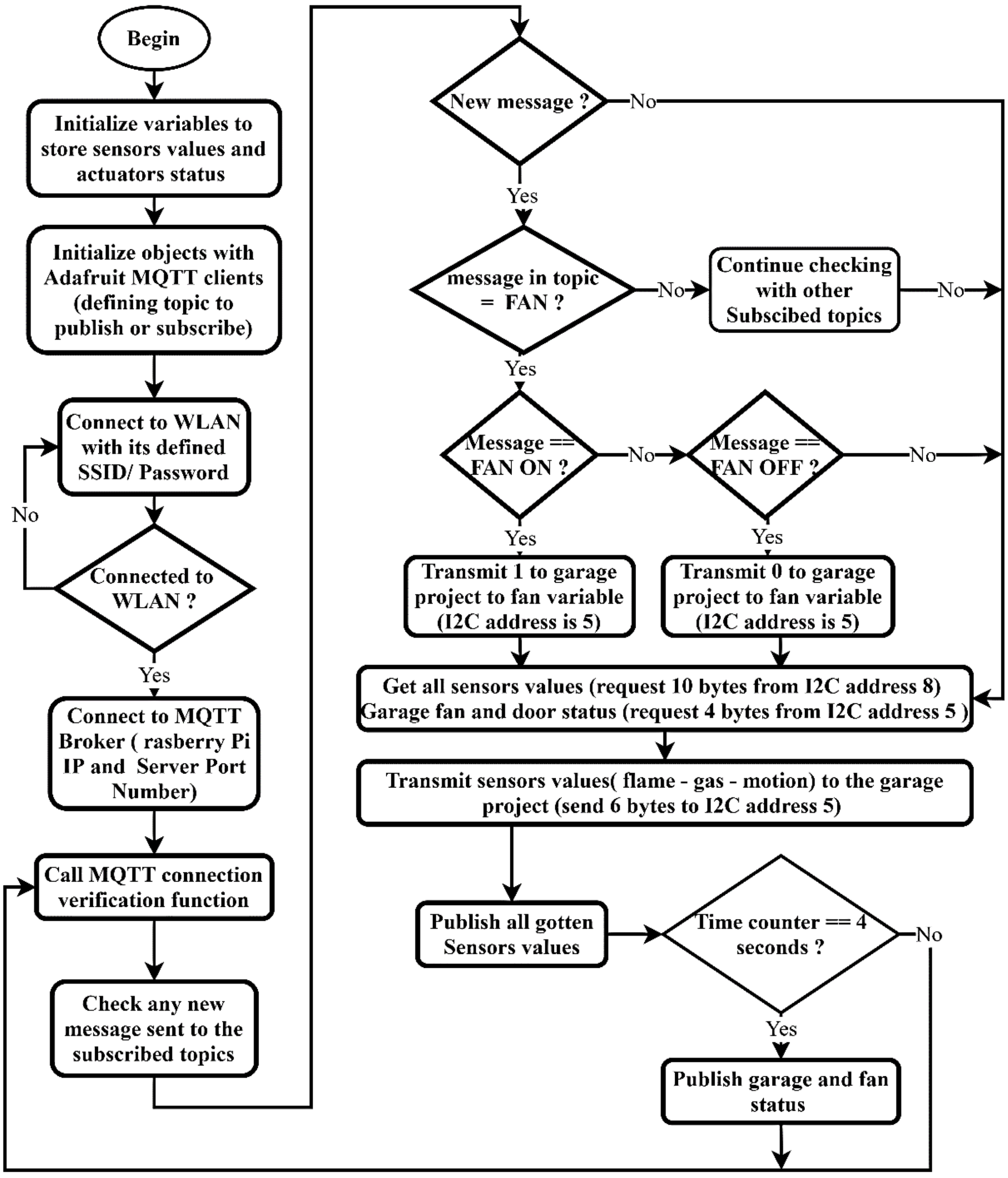
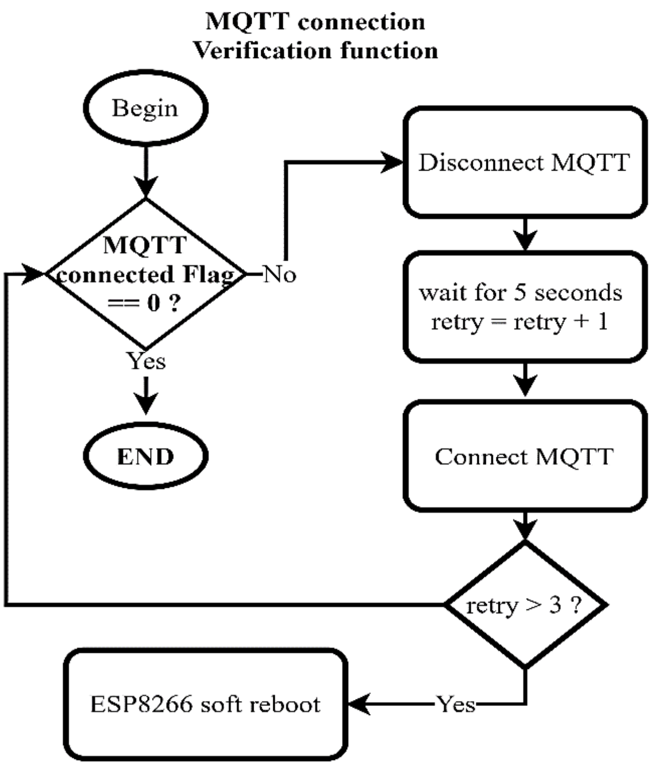

Begin
Initialize variables to store sensors values and actuators status
Initialize objects with Adafruit MQTT clients (defining topic to publish or subscribe)
Connect to WLAN with its defined SSID/ Password
No
Connected to WLAN ?
Yes
Connect to MQTT Broker ( rasberry Pi IP and Server Port Number)
Call MQTT connection verification function
Check any new message sent to the subscribed topics
New message ?
No
Yes
message in topic = FAN ?
No
Continue checking with other Subscibed topics
No
Yes
Message == FAN ON ?
No
Message == FAN OFF ?
No
Yes
Yes
Transmit 1 to garage project to fan variable (I2C address is 5)
Transmit 0 to garage project to fan variable (I2C address is 5)
Get all sensors values (request 10 bytes from I2C address 8) Garage fan and door status (request 4 bytes from I2C address 5 )
Transmit sensors values( flame - gas - motion) to the garage project (send 6 bytes to I2C address 5)
Publish all gotten Sensors values
Time counter == 4 seconds ?
No
Yes
Publish garage and fan status

Figure IV.8 Flowchart: ESP8266 Project

Since the ESP8266 is an I2C master that meant it must be always working without interruptions and that what let us do the MQTT connection verification this function is called on every loop cycle. In case we lost the connection to MQTT server and tried to connect 3 times, the ESP8266 will immediately restart in order to try the Wi-Fi connection for the next start.



## IV.3.4. Security System Using Sensors

Simply the sensors represent slaves that have their values stored, which wait for an order giving by the master, when data is requested the meant device will react either display home status, or set alarms by triggering buzzer sounds setting red light as for indicating danger or green to indicate safety

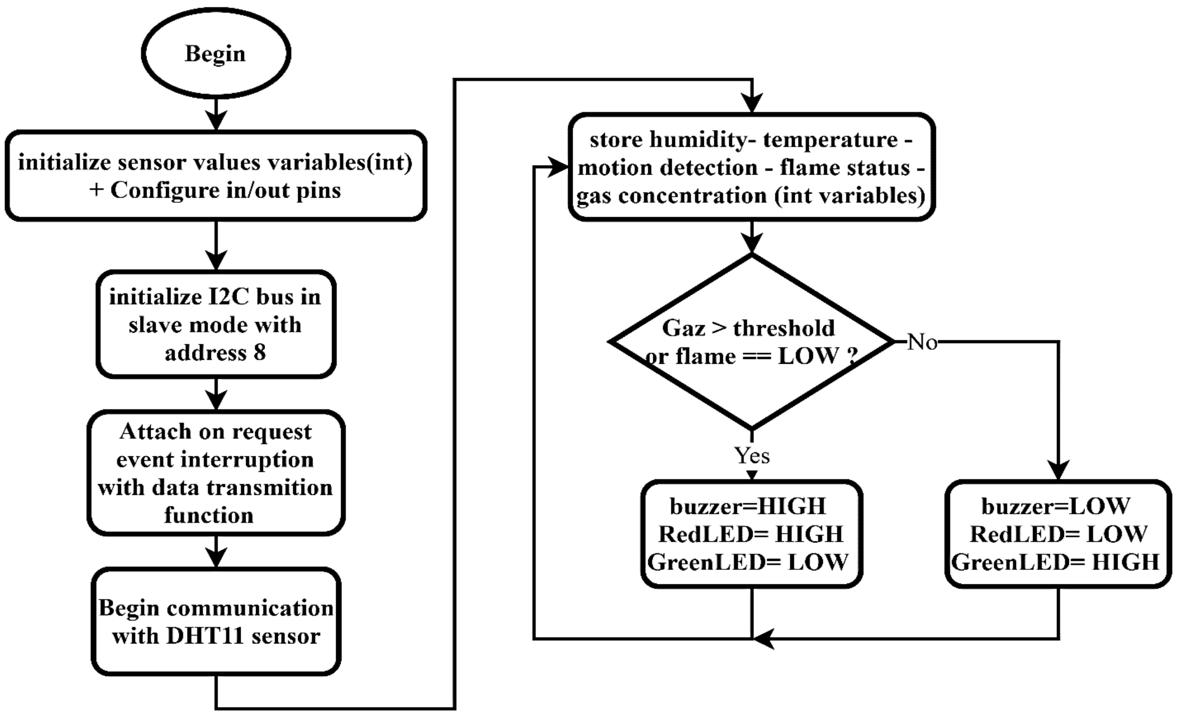

Begin
initialize sensor values variables(int) + Configure in/out pins
initialize I2C bus in slave mode with address 8
Attach on request event interruption with data transmition function
Begin communication with DHT11 sensor
store humidity- temperature - motion detection - flame status - gas concentration (int variables)
Gaz > threshold or flame == LOW ?
No
Yes
buzzer=HIGH
RedLED= HIGH
GreenLED= LOW
buzzer=LOW
RedLED= LOW
GreenLED= HIGH

Figure IV.9 Flowchart: Sensors Project

The security system project is an I2C slave so the I2C Master will interrupt this device, in this case, the data transmit function will be triggered and the slave will stop using the sensors than it transfers the last stored values from the sensors, therefore the master know how much data will be received

## IV.3.4.1. Gas detection

A gas detector can read analog (A0) / (D0) digital output voltage that is proportional to the concentration of smoke/gas. This voltage that the sensor outputs changes accordingly to the smoke/gas level that exists in the atmosphere. In other words, the relationship between voltage and gas concentration is the following:

- The greater the gas concentration, the greater the output voltage
- The lower the gas concentration, the lower the output voltage

**Note:** Results given by sensors are displayed On Serial Monitor so it can help us observe their state.

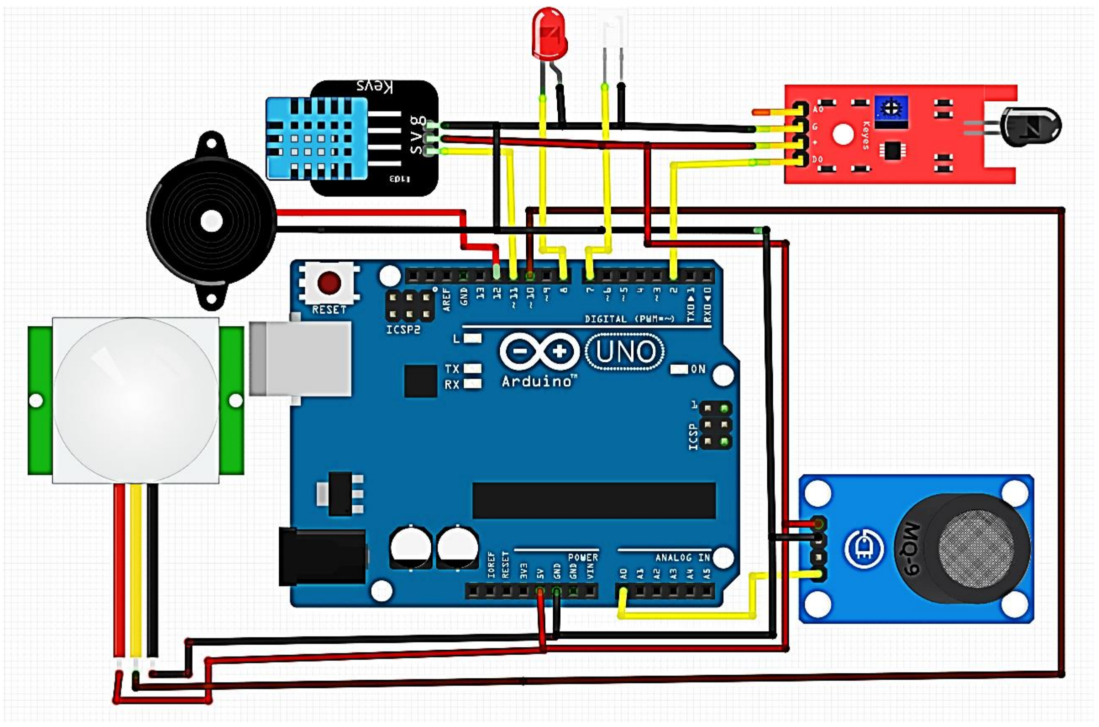
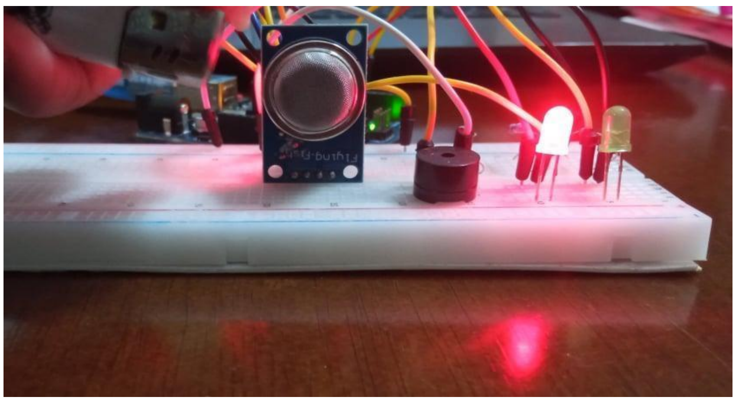
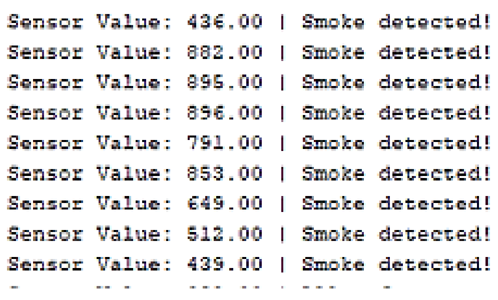


Figure IV.10 Schematic: Security System Using fire, gas, motion, humidity & temperature Sensors

1- When the smoke reaches a certain level, it will make a buzzer and a red LED will turn on. As it’s shown in figure below:

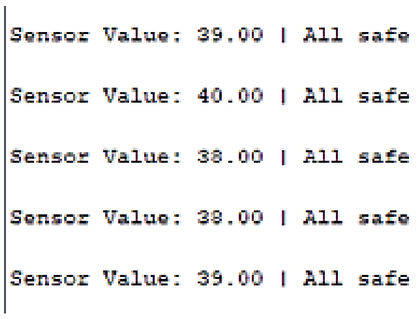




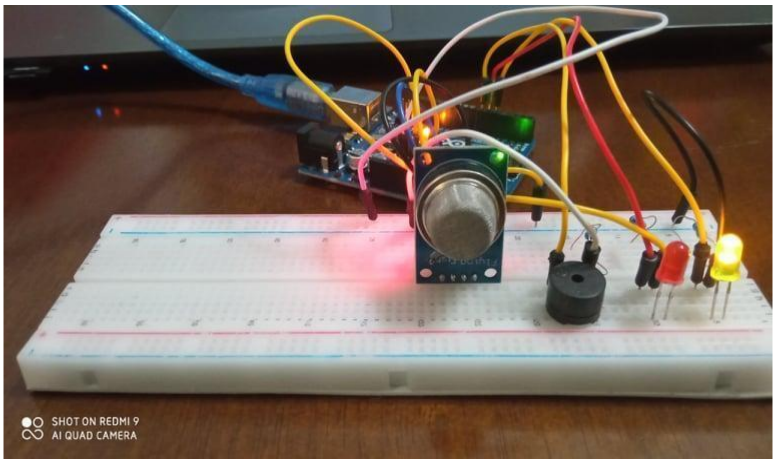
```
Sensor Value: 436.00 | Smoke detected!
Sensor Value: 882.00 | Smoke detected!
Sensor Value: 895.00 | Smoke detected!
Sensor Value: 896.00 | Smoke detected!
Sensor Value: 791.00 | Smoke detected!
Sensor Value: 853.00 | Smoke detected!
Sensor Value: 649.00 | Smoke detected!
Sensor Value: 512.00 | Smoke detected!
Sensor Value: 439.00 | Smoke detected!
```

2- When the output voltage is below that level, a green LED will be on.

Notifying operators in the area where the leak is occurring, giving them the opportunity to leave.



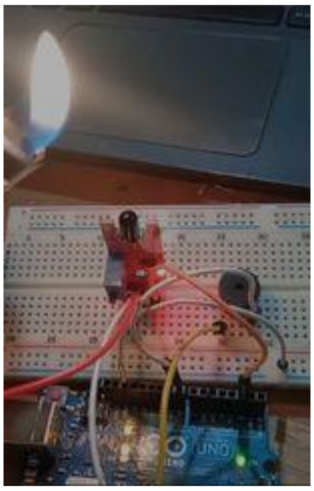
```
Sensor Value: 39.00 | All safe

Sensor Value: 40.00 | All safe

Sensor Value: 38.00 | All safe

Sensor Value: 38.00 | All safe

Sensor Value: 39.00 | All safe
```



## IV.3.4.2. Flame sensor

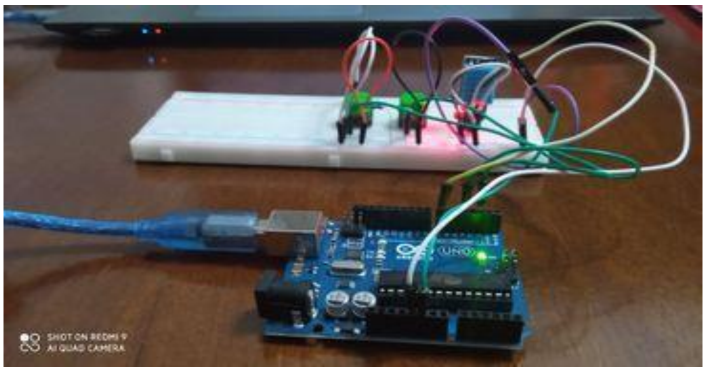
It will read the analog values given by the flame sensor (0-1024).

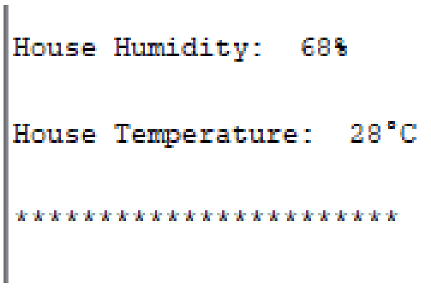
If flame is detected in front of the sensor, it will be activated, the buzzer will make a sound and a warning message will be displayed. Else, if no flame is detected in front of the sensor, a safety message will be displayed.



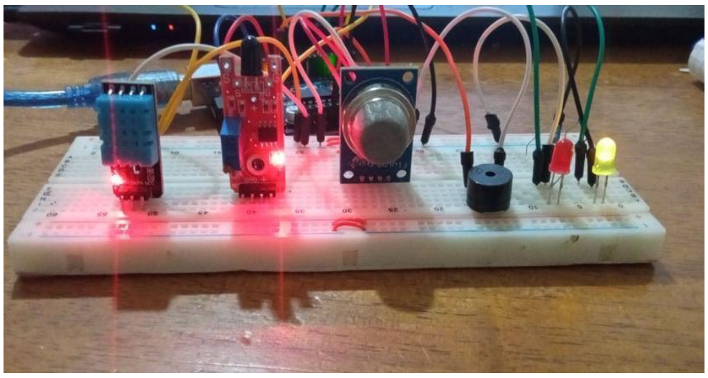
## IV.3.4.3. DHT11 Temperature & Humidity

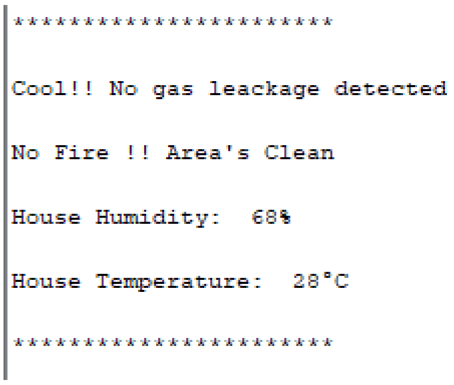
In this project, we will use the DHT11 Temperature/Humidity sensor with three LEDs that will indicate if the temp is too hot, perfect or too cold. And displaying information into the LCD and also the data of the sensor should be displaying itself and updating every 5 seconds on the serial monitor.





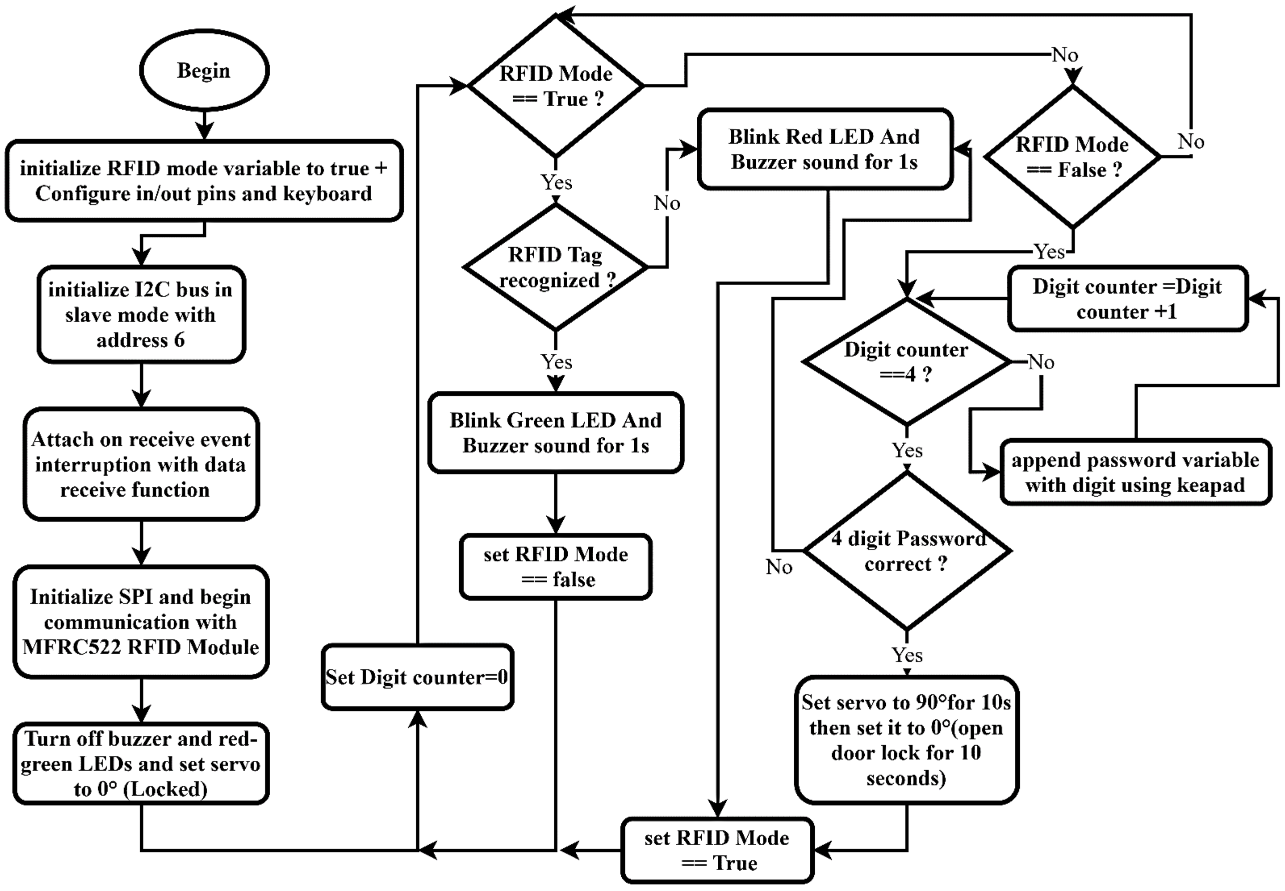
```
House Humidity:  68%

House Temperature:  28°C

***********************
```





```
**********************

Cool!! No gas leackage detected

No Fire !! Area's Clean

House Humidity:  68%

House Temperature:  28°C

**********************
```

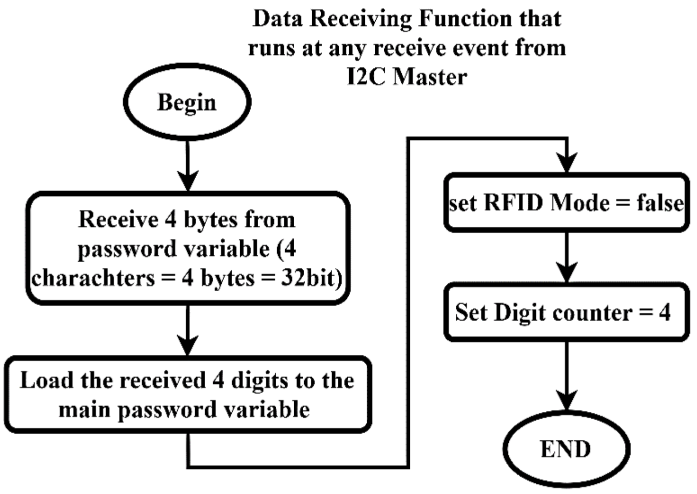
## IV.3.5. RFID and Keypad Based Door Lock



Begin
initialize RFID mode variable to true + Configure in/out pins and keyboard
initialize I2C bus in slave mode with address 6
Attach on receive event interruption with data receive function
Initialize SPI and begin communication with MFRC522 RFID Module
Turn off buzzer and red-green LEDs and set servo to 0° (Locked)
Set Digit counter=0
RFID Mode == True ?
Yes
No
RFID Tag recognized ?
Yes
No
Blink Green LED And Buzzer sound for 1s
set RFID Mode == false
Blink Red LED And Buzzer sound for 1s
RFID Mode == False ?
No
Yes
Digit counter =Digit counter +1
Digit counter ==4 ?
No
Yes
append password variable with digit using keapad
4 digit Password correct ?
No
Yes
Set servo to 90°for 10s then set it to 0°(open door lock for 10 seconds)
set RFID Mode == True

Figure IV.11 Flowchart: RFID Door Lock Project

The RFID project is also an I2C slave but it receives 4 digit password so when we receive the password from the I2C master we don't need RFID tag so we disable the RFID mode and set the keyboard digits counter to 4 in order to start comparing the transferred digits from the I2C master


Data Receiving Function that runs at any receive event from I2C Master
Begin
Receive 4 bytes from password variable (4 charachters = 4 bytes = 32bit)
Load the received 4 digits to the main password variable
set RFID Mode = false
Set Digit counter = 4
END

## Circuit Explanation

- First thing to do, is to retrieve the information of the tags on the serial monitor by uploading the "dumpinfo" from the
- examples in the Arduino.
- The 4x4 keypad has 8 connections but we only used 3 columns for the password, the ones that require numbers.
- In case of entering the wrong password, tag or intrusion the red led will light up indicating danger. Else the green led indicates safety.
- The buzzer will beep anytime we scan tag or for security purposes.
- The servo motor is responsible for closing or opening the door.

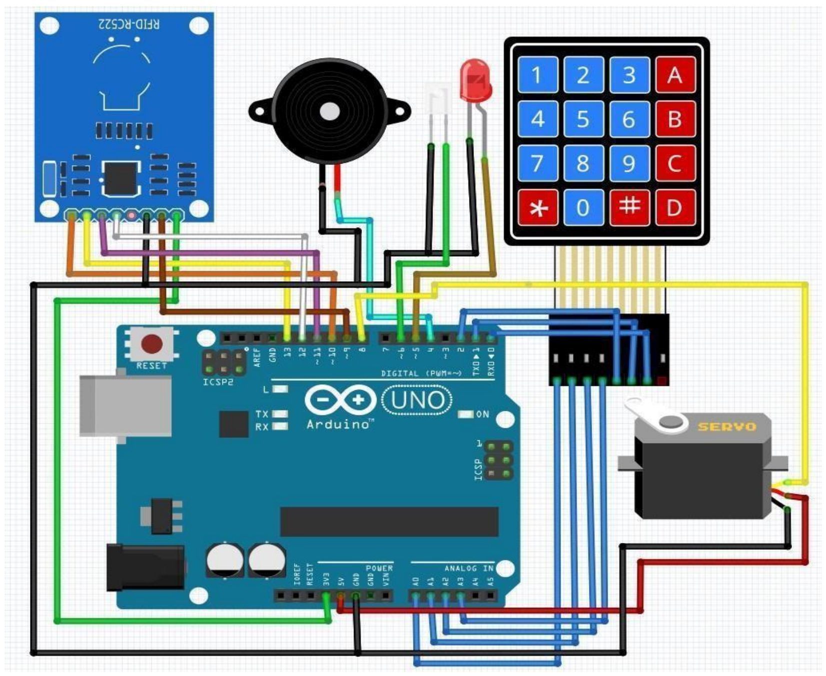
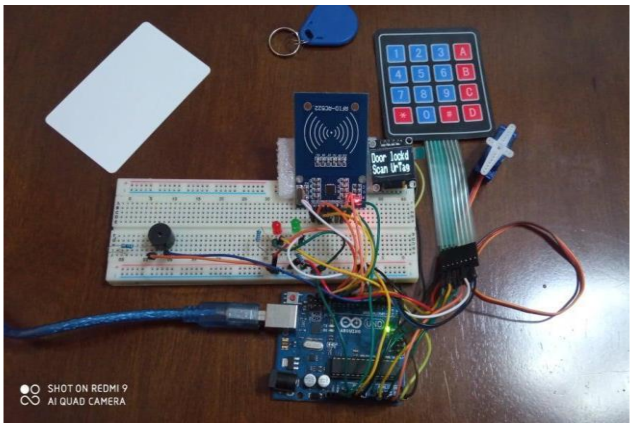
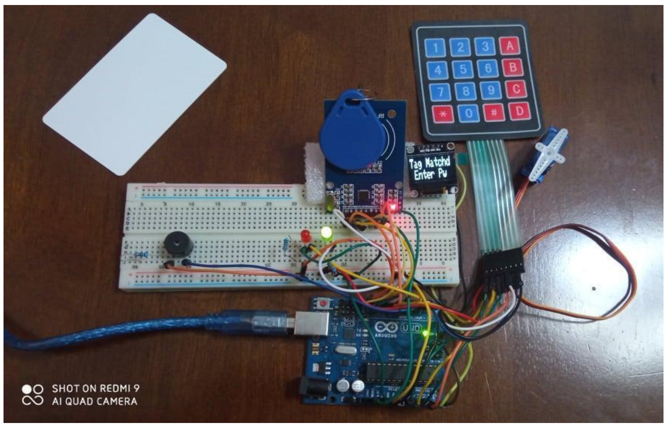


Figure IV.12 Schematic: Door lock system

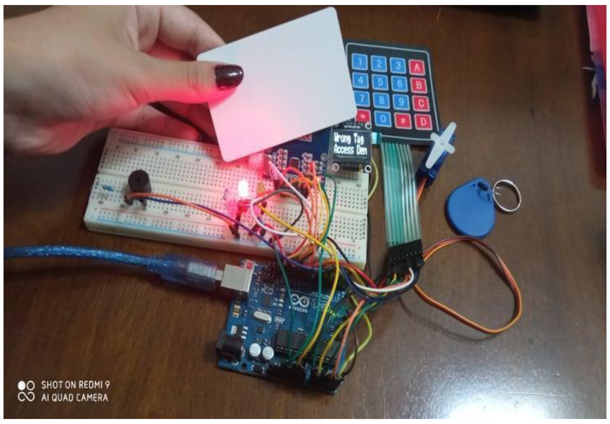
To open the door, the user will have to first scan the right tag and then he will have to enter the correct password. On scanning the wrong tag or on entering the wrong password, the system will deny access.
The main goal is to monitor the door manually or from a smartphone by typing their unique pin numbers into a special website which gives the advantage to interact in case we forget the door is open.

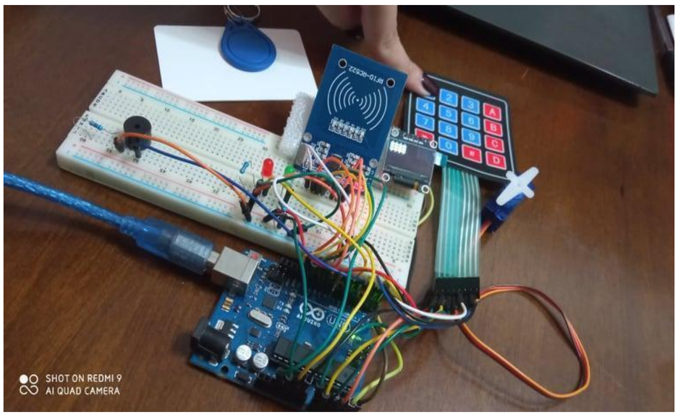




**1-** It displays door locked, scan your tag

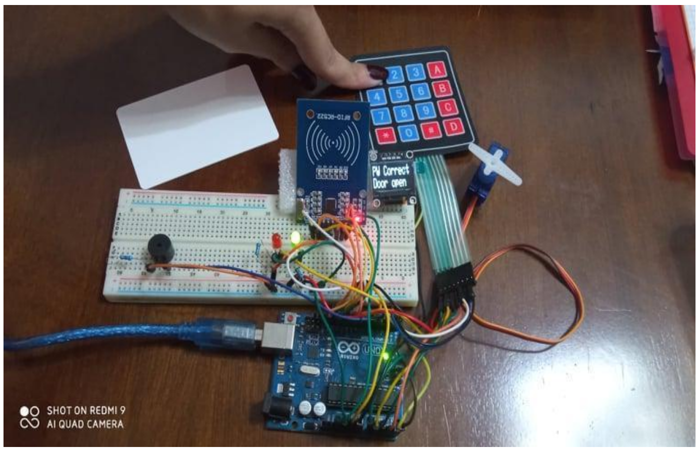
**2-**Scanning the right tag, it will ask for password





3- Entering four digits password

4- if pw is correctly entered, the door shall open



5- Else access will be denied, either wrong tag or wrong password

### IV.3.6. Garage door Project

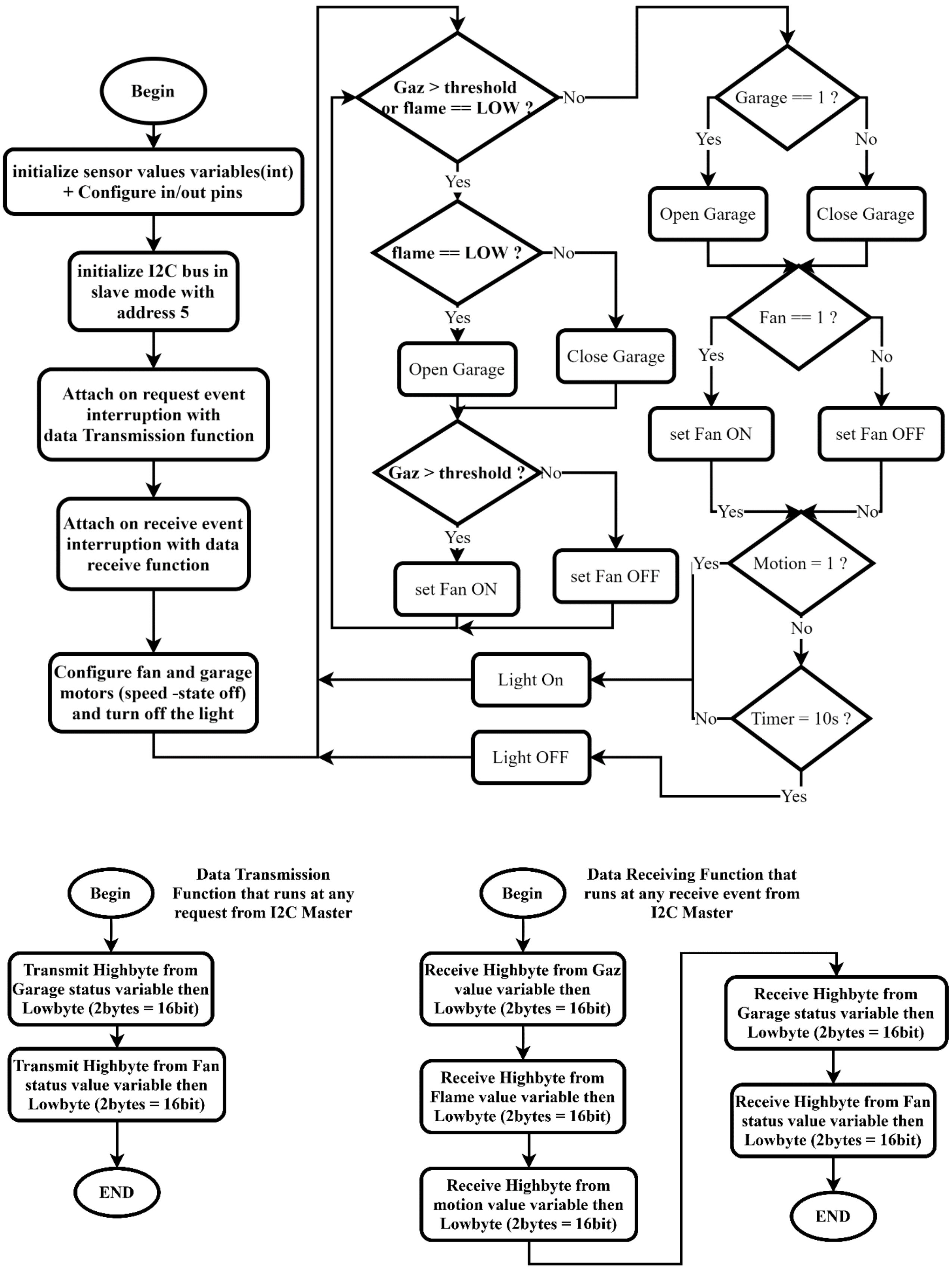

Begin
initialize sensor values variables(int) + Configure in/out pins
initialize I2C bus in slave mode with address 5
Attach on request event interruption with data Transmission function
Attach on receive event interruption with data receive function
Configure fan and garage motors (speed -state off) and turn off the light
Gaz > threshold or flame == LOW ?
No
Yes
flame == LOW ?
No
Yes
Open Garage
Close Garage
Gaz > threshold ?
No
Yes
set Fan ON
set Fan OFF
Garage == 1 ?
Yes
No
Open Garage
Close Garage
Fan == 1 ?
Yes
No
set Fan ON
set Fan OFF
Yes
No
Motion = 1 ?
Yes
No
Timer = 10s ?
No
Yes
Light On
Light OFF
Begin
Data Transmission Function that runs at any request from I2C Master
Transmit Highbyte from Garage status variable then Lowbyte (2bytes = 16bit)
Transmit Highbyte from Fan status value variable then Lowbyte (2bytes = 16bit)
END
Begin
Data Receiving Function that runs at any receive event from I2C Master
Receive Highbyte from Gaz value variable then Lowbyte (2bytes = 16bit)
Receive Highbyte from Flame value variable then Lowbyte (2bytes = 16bit)
Receive Highbyte from motion value variable then Lowbyte (2bytes = 16bit)
Receive Highbyte from Garage status variable then Lowbyte (2bytes = 16bit)
Receive Highbyte from Fan status value variable then Lowbyte (2bytes = 16bit)
END

Figure IV.13 Flowchart Garage Project

The Garage project in every cycle of the master it receives sensors data and node-red buttons status in order to know whether it uses emergency case (control door and fan depending on sensors) or just control them depending on user interface variables. Also, if master asks it receive data it transmits the door and fan status then the master sync it to the user interface.

The main goal is to monitor the garage door and to interact in case we forget the door open. Members could open the garage door using an android app on their mobile phones, in our case the node-red dashboard is our user interface where we can trigger the opening or closing of the garage door by selecting the open/close button.

- The controller will take the action of closing and opening the door.
- Else it will open automatically in case of gas or fire detection, at the same time evacuate the air by turning the fan on.

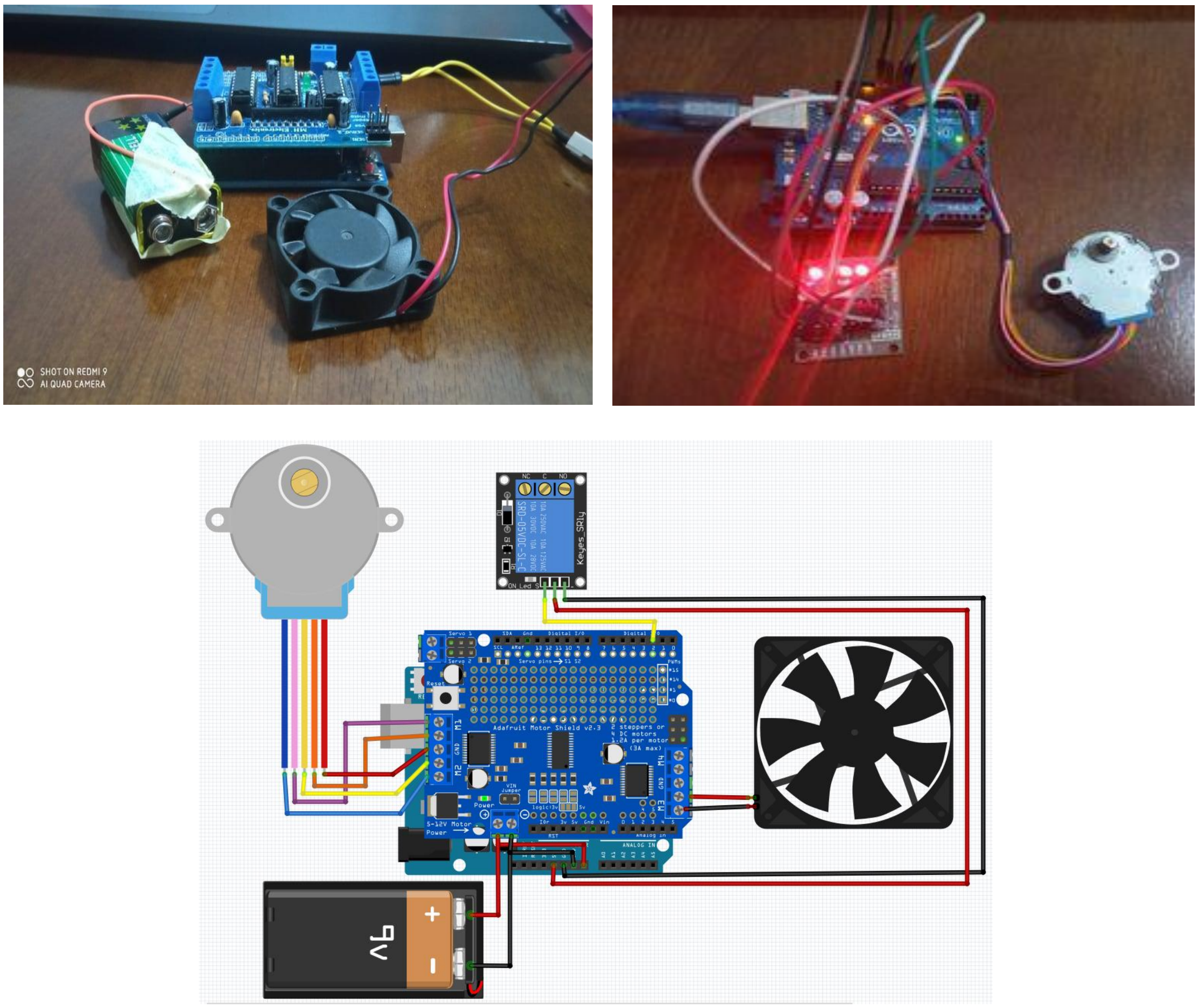


Figure IV.14 Schematic: Garage Project

## IV.3.7. Master-Slave Concept

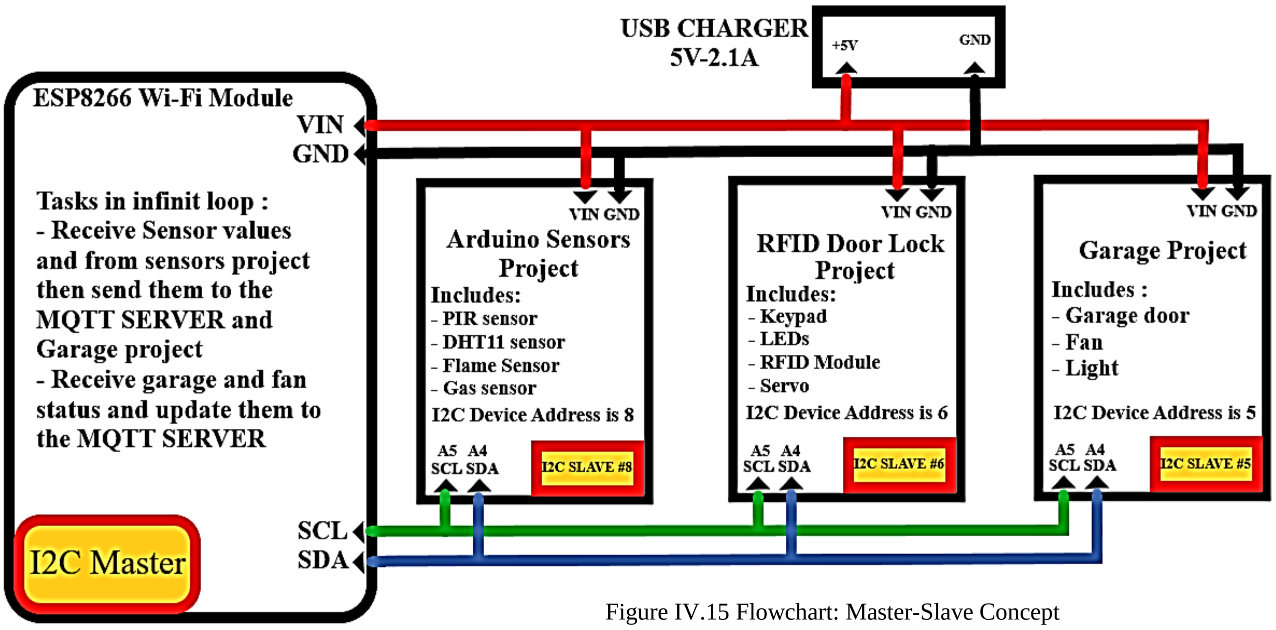
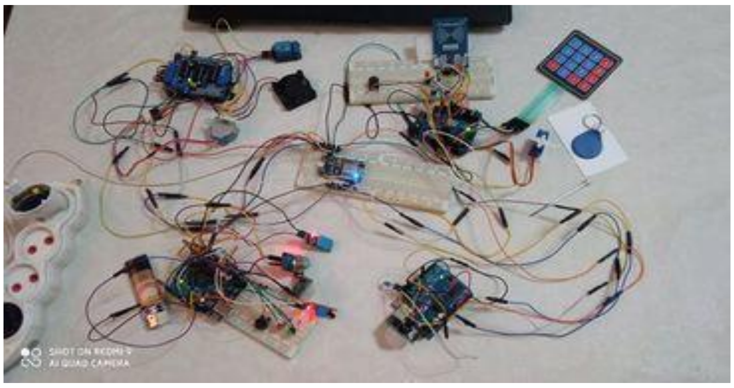
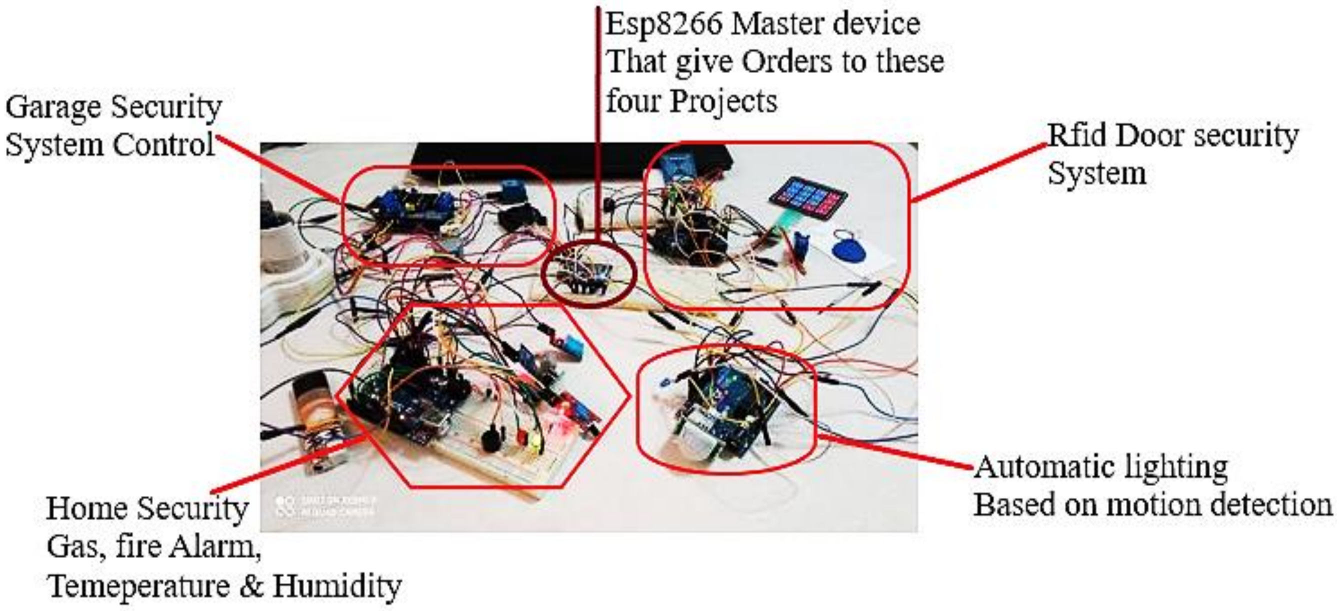


Figure IV.15 Flowchart: Master-Slave Concept

## IV.3.8. Completed Project Preview

Those are the project’s main important part in building a secure Automated House.




Esp8266 Master device That give Orders to these four Projects
Garage Security System Control
Rfid Door security System
Automatic lighting Based on motion detection
Home Security Gas, fire Alarm, Temeperature & Humidity

# General Conclusion

To create a well-designed Smart Home, we needed to integrate IOT hardware, equipped with actuators and sensors. Depending on the sensors values, we had to train house to intelligently interact not only between appliances but with home owners too, by integrating them in computing machines that are included into the IOT system. Moreover, this will let the home interact more smartly.

Practically, creating separated devices that can interact with each home appliance gives a high flexibility and compatibility for home customization, especially due to the fact that they are easy to install, providing an easy usage experience. Consequently, after testing the home, we definitely can say that integrating Mqtt protocol “which is responsible mostly for devices communication”, was a good choice for the fluidity of data transmission he provides due to the light weight message the protocol offers, as well as the reliability that quality of service integration within the protocol affords. And the sensors that interacted with each other in order to design the monitoring and control of home devices, based on the microcontrollers NodeMcu, Raspberry Pi, and arduino's modules, which worked flawlessly.

Despite the complexity and difficulty of this research, implementing this project gave us the opportunity to learn new skills and gain new knowledge in the field of smart automation. In addition, our method of thinking has improved. Furthermore, one of project’s main originality relies on a secure platform that combines online and offline services to allow users more privacy and home independence (Offline Mode can be used for peoples who don’t want to have their houses Online), as well as being sustainable (track user behavior), reliable (not expensive Hardware devices), adaptable (on one needs), scalable, fully controlled (easy interface) and Secure.

We can invest in the field of home appliances by developing fully functional appliances that are compatible with this platform. Furthermore, we may be able to work out deals with companies that already have extensive expertise making home appliances, allowing us to integrate and generalize our controlling interface to their products. Our objective is to empower our users to transform their homes into smart homes by leveraging IoT and AI technologies to achieve maximum comfort with minimal effort.

In conclusion, the contribution of this project is mainly summed up in the discovery of a new field called home automation, which is a very interesting and very vast and innovative field. Though, we took the risk to engage with it and it was really challenging for us, to study something new that we wasn’t sure if it will work out or not, but it was worth it. It gave us the chance to hit many different areas to understand too in order to hook up everything. Though, we find it difficult but after a hard work we make it through. Also, we had to learn electronics (even if it’s not our specialty we took the initiation). In addition, to studying the working concept of hardware, to the implementation of the protocol that we wasn't familiar with at all, since it was new to us. However, we aim to expand our knowledge and experience in this domain, considering it as a future research. Then, we sincerely hope that this effort will serve as a springboard for further in-depth studies.

# Annex A: Trouble Shooting

The period of the realization of this project was an educational period, despite home automation, we have penetrated several areas such as: the internet of things, electronics…, But it wasn’t that easy. Such an accomplishment is not without its difficulties. It's worth noting that we run into a number of issues ranging from hardware circuit simulation, to libraries compatibility, and particularly with the Raspberry Pi in terms of network stability. As a result, we've included this troubleshooting. In hope of helping future students interested in this field to process information faster, by avoiding these main obstacles we’ve faced.

So based on our experience during this journey, here are some examples of what we faced, and fortunately solved:

- If you ever find yourself unable to connect via SSH, you can always check the IP address if it’s the raspberry pi’s or if you made any changes in your raspberry’s pi uwf (firewall) setting, which took us a while to figure it out in our case, it’s good to be curious but be always be careful.
- Considering alarms sound, if you were to use buzzers. Then, if anything happens like starting to buzz from the start without pre-orders, then check your code settings, go to setup and where you are defining the buzzer as an output [pinMode(buzzer,OUTPUT)], make sure to declare its status as low, [digitalWrite(buzzer,LOW)].
- Need to be careful when installing libraries, you can find so many within same name but different creators, to avoid troublesome errors, use only what you need.
- To interact with MQTT broker in LAN network in raspberry we must disable its firewall.
- We must choose good power source and ground all hardware together in order to let the project function correctly.
- In our project, we must run the raspberry first, then run the NodeMCU with Arduino’s. However, we must reset the NodeMCU in order to establish communication with all Arduino’s after they get sensors data via I2C.

# Annex B: Project Summary

This project was created in an attempt to see how we could implement the latest technology for home automation using different components.

Also, we wanted to see how can communicate with some of the devices installed in this project from the distance using Raspberry pi over Mqtt protocol, where most of our sensors are controlled by Esp826, which is needed to control and monitor some parts of the home using Node-red dashboard over Wi-Fi.

➢ <u>The components that are connected to the Arduino Uno board are:</u>

- MQ5 gas sensor: used to detect gas leakage in the house;
- Flame sensor: used to detect a fire inside the house;
- Buzzer: to alert user when there is a gas leakage, intrusion,, or a fire inside house;
- DHT22 temperature and humidity sensor: used to monitor the
- temperature and humidity inside the house;
- Fan or a 5V Stepper motor: used as a fan to reduce the temperature inside the house;
- PIR sensor: used to detect movement inside the house;
- Light sensor: used to detect the level of light inside the house;
- LED: used to make light inside the house when is dark;
- 16x2 LCD & Oled with I2C interface: used to show some messages;
- RFID: used to open or close the entrance door, depending on the access level of the user;
- MG90 S servo motor: used to open the entrance door, if a valid access card is used near the RFID,

plus a valid password for extra security.

➢ <u>The components that are connected to ESP32 development board are:</u> Are the three arduino's showed in figures 3, 4, 5 above that are responsible for the following applications:

1. Application to detect a gas leakage using MQ 5 gas sensor and the buzzer.
2. Application to maintain a certain temperature inside the house using DHT11 sensor and a 5V dc motor.
3. Application to detect fire inside the house using infrared sensor (fire sensor module), and a buzzer.

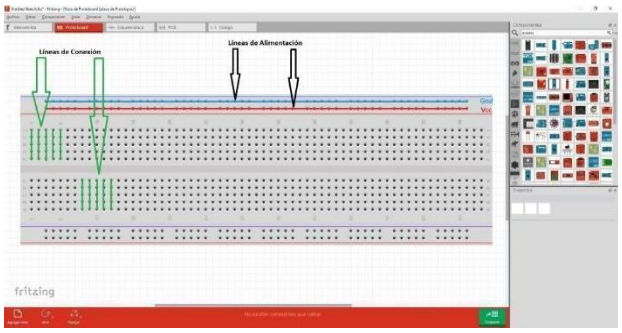
4. Application to turn on the lights inside the house automatically if there is movement detected and is dark inside the house, using PIR sensor, and a light sensor.
5. Application to have access inside the house using an RFID module, Oled module with i2c interface and an MG 90 S servo motor.

Using all these applications, anyone can build a smart home to make their life better and more comfortable. All of the components used with esp8266 and Raspberry pi help us to monitor and control the house from any place as long as we are connected to the internet.

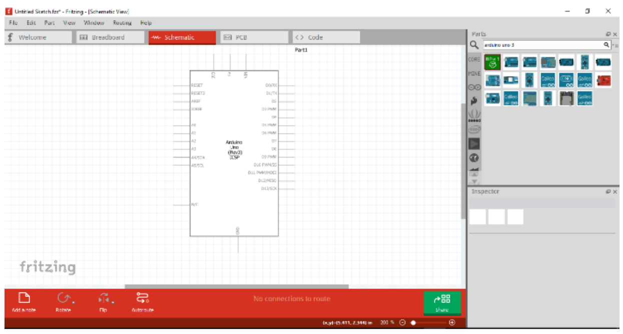
# Annex C: Fritzing

Is the Software we used in creating our circuit's Schematic. It's an innovative Electronic Design Automation software that has an extensive library of electronic components with a very simple user interface, where we can view these three following modes:

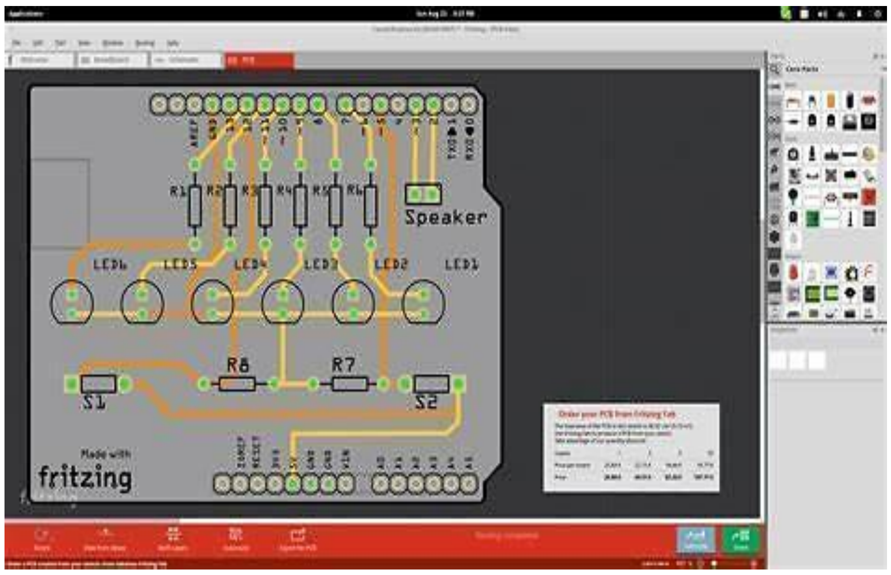
1. **Breadboard:** is where virtual electronic components can be wired and placed on a virtual breadboard. Parts that do not exist can be created as well.



2. **Schematic:** is where the former representation of the schematic can be viewed and edited. Changes made in one view instantly affect all other views.



3. **PCB:** allows us to place parts on a printed circuit board. An auto-router generates the traces and the final pcb layout can be exported to the necessary production formats.



# Annex D: Abbreviation List

## General Terms

- IOT: Internet of Things
- API: Application Programming Interface
- CEO: Chief Executive Office
- DIY : Do It Yourself
- IDE: Integrated Development Environment
- GUI: Graphical User Interface
- ACK: Acknowledgment
- NACK: Negative ACKnowledge
- SCL: Serial Clock Line
- SDA: Serial DAta
- PC: Personal Computer
- AC: Air Conditioner
- URI: Universal Resource Identifier
- URL: Uniform Resource Locator
- UWB: Ultra Wide Band
- USB: Universal Serial Bus
- GHz: GigaHertz
- JMS: Java Message Service
- RFID: Radio Frequency IDentification
- LWT: Last Will and Testament.
- PWM: Pulse Width Modulation
- COM: Communication Port
- JSON: JavaScript Object Notation
- PCB: Printed Circuit Board

## Area Network

- WLAN: Wireless Local Area Network
- WWANs : Wireless Wide Area Networks
- WIMAX: Worldwide Interoperability for Microwave Access
- WANs: Wide Area Networks
- MANs: Metropolitan Area Networks
- LAN: Local Area Network
- PANs: Personal Area Networks
- BANs: Body Area networks
- UMTS: Universal Mobile Telephone System
- EDGE: Enhanced Data for GSM Evolution
- GPRS: General Packet Radio Service

## Transport Protocols

- IP: Internet Protocol
- TCP: Transmission Control Protocol
- UDP: User Datagram Protocol
- SCTP: Stream Control Transmission Protocol

## Security Protocols

- TLS: Transport Layer Security
- SSL:  Secure Sockets Layer
- SSH: Secure Shell
- DTLS: Datagram Transport Layer Security
- IPsec: Internet Protocol Security
- QOS: Quality of Service

## Network & messaging protocols

- Wi-Fi: Wireless Fidelity
- ZigBee: ZigBee End-Device and ZigBee Coordinator
- ZWave: ZWave
- RF: Radio Frequency
- BLE: Bluetooth Low Energy
- RESTful: Representational State Transfer
- MQTT: Message Queuing Telemetry Transport
- HTTP: Hypertext Transfer Protocol
- AMQP: Advanced Message Queuing Protocol
- CoAP: Constrained Application Protocol
- XMPP: Extensible Messaging and Presence Protocol
- EXL: Efficient XML Interchanges
- DDS: Data Distribution Service
- IM: Instant Messaging
- M2M: Machine-to-Machine
- WSN: Wireless Sensor Networks

## PINS Input/output

- ADC: Analog to Digital Converter
- GPIO: General-Purpose Input/output
- DIO: Digital I/O Pins
- ADC: Analog Input Pins

## Serial Communication

- I2C: Inter-Integrated Circuit
- I2S: Inter-IC Sound
- SPI: Serial Peripheral Interface
- DSI : Display Serial Interface
- UARTs: Universal Asynchronous Receiver/Transmitter
- CLK: Clock pin

- Serial Monitor baud rate: rate at which information is transferred in a communication channel

## Current

- AC: Alternative Current
- DC: Direct Current
- IC: Integrated Circuit
- Power jack: a power source

## PORTS

- USB: Universal Serial Bus
- HDMI: High-Definition Multimedia Interface
- CSI: Camera Serial Interface port
- Power Micro USB: for powering stuff or charging

## Memory & storage

- EEPROM: Electrically Erasable Programmable Read-Only Memory
- GDDRAM: Graphics Double Data Rate Synchronous Dynamic Random-Access Memory
- SRAM: Static Random-Access Memory
- Flash Memory: Electronically Erasable Programmable Read Only Memory
- RAM: Random Access Memory
- SD-card: Secure Digital Card for storing data
- TF- card: Trans Flash Card

## Microcontroller

- ATmega328P: AVR family microcontroller
- MCU: Microcontroller Unite
- SOC: System on Chip
- BCM43438: Wi-Fi chip

## Organizations

- IBM: International Business Machines, a leading US computer manufacturer
- Facebook: Free Expression, a messaging application
- Eurotech: strategic partnership of leading European universities of science and technology
- Cisco: US Technology Company that is best known for its networking products.
- Red Hat: an IBM subsidiary and software company that provides open source software products to enterprises.
- Software AG: a business infrastructure software provider with products in various technologies
- Tibco: The Information Bus Company creates software and hardware products around business solutions based on Service Oriented Architecture or SOA.
- ITSO: smart ticketing technology (Operates smart cards)
- M2Mi: Machine-To-Machine Intelligence (computer Software Company)
- AWS: Amazon Web Services (a cloud service from Amazon)
- InduSoft: Web Studio is collection of automation tools to develop modern Human Machine

Interfaces (HMI), Supervisory Control and Data Acquisition (SCADA) systems, and embedded instrumentation and control applications.

- Fiorano: a global leader in Integration middleware, API Management and Peer-to-peer distributed systems.
- Microsoft: a leading global vendor of computer software; hardware for computer, mobile and gaming systems; and cloud services.
- JP Morgan: a global leader in financial services
- Bank of America: one of the world's leading financial institutions
- Eclipse Foundation: a European-based international not-for-profit association, an open innovation processes, and community-building events.
- Adafruit: Open-source Hardware Company, The company designs, manufactures and sells electronics products, electronics components, tools and accessories.
- Espressif: a Chinese Bluetooth\WiFi SoC manufacturer - its products are found in varied SmartThings (IoT) including Security Systems.
- OSI: International Organization for Normalization

## Standards:

- OASIS: Organization for the Advancement of Structured Information Standards
- IEEE: Institute of Electrical and Electronics Engineers
- IETF: Internet Engineering Task Force
- IBM: International Business Machines
- W3C: World Wide Web Consortium
- OSI: Open Systems Interconnection

## Units to Be Aware of:

- Gas Concentration unit: PPM (Parts per million)
- Temperature unit: °C (Celsius)
- Humidity unit: % (percentage)
- Quiescent current: 2mA
- Data Rate: Mbps
- Frequency: MHz/GHz
- Relay voltage: 250VAC or 30VDC
- Servo Rotation : 0°-180°
- Insulated Power: 600VAC/1mA/1s
- IR wavelength :760nm ~1100nm
- Motor Speed : rpm

# Annex E: Main (Master) Code

**Program Code of ESP8266 MASTER SLAVE INTERACTION MAIN CODE**

*************** Esp8266 interaction with mqtt over raspberry pi ***************

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <Wire.h>
/************************* WiFi Access Point *********************************/
#define WLAN_SSID "shadowBroker"
#define WLAN_PASS "@1lasy!@"
#define MQTT_SERVER "192.168.43.99" // give static address
#define MQTT_PORT 1883
#define MQTT_USERNAME ""
#define MQTT_PASSWORD ""
char message;

 int Gaz_val=0;
 int flame=0;

 int humid=0;
 int temperature=0;
 int motion=0;
 int garag=0;
 int garagelight=0;
 int fan=0;
unsigned long previousMillis = 0;
const long interval = 5000;
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, MQTT_PORT, MQTT_USERNAME,
MQTT_PASSWORD);

Adafruit_MQTT_Publish gas = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "gas");
Adafruit_MQTT_Publish fire = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "fire");
Adafruit_MQTT_Publish temp = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "temp");
Adafruit_MQTT_Publish hum = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "hum");
Adafruit_MQTT_Publish GL = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "GL");
Adafruit_MQTT_Publish GD = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "GD");
Adafruit_MQTT_Publish FN = Adafruit_MQTT_Publish(&mqtt, MQTT_USERNAME "FN");
// Setup a feed called 'esp8266_led' for subscribing to changes.
Adafruit_MQTT_Subscribe esp8266_led = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME
"/leds/esp8266");
Adafruit_MQTT_Subscribe doorpass = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME
"doorpass");
Adafruit_MQTT_Subscribe garage = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME
"garage");
Adafruit_MQTT_Subscribe garageLIGHT = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME
"garageLIGHT");
Adafruit_MQTT_Subscribe FAN = Adafruit_MQTT_Subscribe(&mqtt, MQTT_USERNAME "FAN");
```

```
/**************************** Sketch Code ************************************/
void MQTT_connect();
void setup() {
 Wire.begin();
 pinMode(LED_BUILTIN, OUTPUT);//12313123
 //pinMode(D4, INPUT);
 Serial.begin(115200);
 delay(10);
 Serial.println(F("RPi-ESP-MQTT"));
 // Connect to WiFi access point.
 Serial.println(); Serial.println();
 Serial.print("Connecting to ");
 Serial.println(WLAN_SSID);
 WiFi.begin(WLAN_SSID, WLAN_PASS);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
 Serial.println();
 Serial.println("WiFi connected");
 Serial.println("IP address: "); Serial.println(WiFi.localIP());
 // Setup MQTT subscription for esp8266_led feed.
 mqtt.subscribe(&esp8266_led);
 mqtt.subscribe(&doorpass);
 mqtt.subscribe(&garage);
 mqtt.subscribe(&garageLIGHT);
 mqtt.subscribe(&FAN);
}
uint32_t x=0;
void loop() {
 MQTT_connect();
 Adafruit_MQTT_Subscribe *subscription;
 while ((subscription = mqtt.readSubscription())) {
  if (subscription == &esp8266_led) {
  Serial.print(F("Got: "));
  Serial.println((char* )esp8266_led.lastread);
  String message = (char* )esp8266_led.lastread;
    if(message=="LED ON"){
    Serial.println("yes");
    digitalWrite(LED_BUILTIN, LOW);   // turn the LED on (HIGH is the voltage level)
    }
   if((String)message=="LED OFF"){
    Serial.println("no");
    digitalWrite(LED_BUILTIN, HIGH);    // turn the LED off by making the voltage LOW
   }
  }
  if (subscription == &doorpass) {
  Serial.print(F("Got doorpass: "));
  Serial.println((char* )doorpass.lastread);
  String message = (char* )doorpass.lastread;
send_password();
  }
  if (subscription == &garage) {
    String message = (char* )garage.lastread;
     if(message=="GARAGE OPEN"){
     garag=1;
     send_data();
    }else if((String)message=="GARAGE CLOSE"){
    garag=0;
    send_data();
   }
    }
 if (subscription == &FAN) {
    String message = (char* )FAN.lastread;
      if(message=="GARAGEFN ON"){
     fan=1;
     Serial.println("yes");
     send_data();
    }else if((String)message=="GARAGEFN OFF"){
    fan=0;
    Serial.println("no");
    send_data();
   }
   }
   }
  recive_Arduino_data();
  //send_data();
  gas.publish(Gaz_val);
  fire.publish(flame);
  temp.publish(temperature);
  hum.publish(humid);
  unsigned long currentMillis = millis();
  if (currentMillis-previousMillis >= interval){
  previousMillis=currentMillis;
  if(fan==0){FN.publish("GARAGEFN OFF");}else{FN.publish("GARAGEFN ON");}
  if(garag==0){GD.publish("GARAGE CLOSE");}else{GD.publish("GARAGE OPEN");}
  }
   }

  // Function to connect and reconnect as necessary to the MQTT server.
  void MQTT_connect() {
 int8_t ret;
 // Stop if already connected.
 if (mqtt.connected()) {
  return;
 }
 Serial.print("Connecting to MQTT... ");
 uint8_t retries = 3;
 while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
  Serial.println(mqtt.connectErrorString(ret));
  Serial.println("Retrying MQTT connection in 5 seconds...");
  mqtt.disconnect();
  delay(5000); // wait 5 seconds
  retries--;
  if (retries == 0) {
  // basically die and wait for WDT to reset me
  while (1);
  }   }
 Serial.println("MQTT Connected!");
 }
 void recive_Arduino_data(){
   Gaz_val=0;
   flame=0;
   humid=0;
   temperature=0;
   motion=0;
   fan=0;
    //Serial.println(c);
    Wire.requestFrom(5,4);
    garag += Wire.read()<< 8; // receive a byte as character
    garag += Wire.read();
    //garagelight += Wire.read()<< 8; // receive a byte
    //garagelight += Wire.read();
    fan += Wire.read()<< 8; // receive a byte
    fan += Wire.read();
    Wire.requestFrom(8,10);    // request 8 bytes from slave device #8
    Gaz_val += Wire.read()<< 8; // receive a byte as character
    Gaz_val += Wire.read();
    flame += Wire.read()<< 8; // receive a byte
    flame += Wire.read();
    humid += Wire.read()<< 8; // receive a byte
    humid += Wire.read();
    temperature += Wire.read()<< 8; // receive a byte as character
    temperature += Wire.read();
    motion +=Wire.read()<< 8; // receive a byte
    motion += Wire.read();
    //motion=digitalRead(D4);  }
 void send_data(){
  Wire.beginTransmission(5); // transmit to device #5
  Wire.write(highByte(Gaz_val)); //I2C is byte oriented;
  Wire.write(lowByte(Gaz_val));
  Wire.write(highByte(flame)); //I2C is byte oriented;
  Wire.write(lowByte(flame));
  Wire.write(highByte(motion)); //I2C is byte oriented;
  Wire.write(lowByte(motion));
  //Wire.write(highByte(garagelight)); //I2C is byte oriented;
  //Wire.write(lowByte(garagelight));
  Wire.write(highByte(garag)); //I2C is byte oriented;
  Wire.write(lowByte(garag));
  Wire.write(highByte(fan)); //I2C is byte oriented;
  Wire.write(lowByte(fan));
  Wire.endTransmission(); // stop transmitting }
 void send_password(){
  Wire.beginTransmission(6); // transmit to device #6
  Wire.write((char* )doorpass.lastread); // sends x
  Wire.endTransmission(); // stop transmitting }
```

# References

[1] Prof. Prashant Rathod, Syed Khizaruddin, Rashmi Kotian, Shubham Lal.

[2] "The Smart Home Concept: our immediate future", Vincent et Al, IEEE International Conf 2006,doi:10.1109/ICELIE.2006.347206,[Online].Available:https://www.researchgate.net/publication/224696459_The_Smart_Home_Concept_our_immediate_future

[3] L. D. S. al., "State of the art of smart home", vol. 25, pp. 1313-1321, 2012.

[4] R. Harper, "Smart Homes: Past, Present and Future," in Inside the Smart Home, Bristol,UK, Springer Science & Business Media, 2006, pp. 17-39.

[5] Samuel, "A Review of Connectivity Challenges in IoT-Smart Home," in International Conference on Big Data and Smart City, Muscat, Sultanate of Oman, 2016.

[6] "Choice of Effective Messaging Protocols for IoT Systems", Nitin Naik et Al, Conf 2017, IEEE,International,Systems,Engineering,Symposium,(ISSE),doi:10.1109/SysEng.2017.8088251,[Online],Available: https://ieeexplore.ieee.org/abstract/document/8088251/ .

[7] Kelltontech, "internet-of-things-protocols- standards",[Online],Available: https://www.kelltontech.com/kellton-tech-blog/internet-of-things-protocols- standards

[8] "Comparison with HTTP and MQTT In Internet of Things (IoT)", ICIRCA, Conf 2017 International Conference on Control, Electronics, and Communications (ICCEREC), doi: 10.1109/ICCEREC.2016.7814989,[Online].Available:https://ieeexplore.ieee.org/abstract/document/7814989.

[9] "Design_and_Implementation_of_Push_Notification_S stem_Based_on_the_MQTT_Protocol", Yong-Wang, Conf 2013 Researchgate, doi: 10.2991/isca-13.2013.20, [Online]. Available: https://www.researchgate.net/publication/266650239_Design_and_Implementation_of_Push_Notification_System_Based_on_the_MQTT_Protocol

[10] "A Servey On MQTT Protocol Of  Internet Of Things", Dipa Soni, Conf 2017 Researchgate,Chandubhai,SPatel,Institute,of,Technology,[Online].Available: https://www.researchgate.net/profile/Dipa- Soni/publication/

[11] HiveMq,"what,is,mqtt-protocol-an-introduction-about-mqtt",[Online].Available: https://www.hivemq.com/mqtt-5/

[12] "Smart home automation with MQTT", Andrei Cornel et AL, Conf 2019 54th International Universities Power Engineering, doi: 10.1109/UPEC.2019.8893617, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8893617

[13] "Specifying an MQTT Tree for a Connected Smart Home", Van den Bossche A. et Al, ICOST 2018 Lecture Notes in Computer Science, vol 10898. Springer, Cham. https://doi.org/10.1007/978-3-319-94523-1_21

[14] "Secure MQTT for Internet of Things (IoT) " ,singh et Al, Conf 2015 Fifth International Conference on Communication Systems and Network Technologies, doi: 10.1109/CSNT.2015.16,[Online].Available:https://ieeexplore.ieee.org/abstract/document/7280018,

[15] Raspberry.Community,"Raspberry-PIDocumentation",[Online].Available:

https://www.raspberrypi.org

[16]	NodeRed Community "Node-Red Documentation", [Online]. Available: https://nodered.org

[17]	RandomNerdTutorials,"MQTTServerWorkConcept",[Online].Available:https://randomnerdtutorials.com/MQTT-broker

[18]	Electronicshub,"Master,slave,interaction's,workconcept",[Online].Available: https://www.electronicshub.org/basics-i2c-communication/

[19]	Vinay.S, Kusuma.M, "Home Automation Using Internet of Things", International Research Journal of Engineering and Technology, 2015, Vol. 2, No. 3.
[20]	Science Soft – Professional Software Development, " How IoT architecture,works",[Online].Available:https://www.scnsoft.com/blog/iot-architecture-in-a-nutshell-and-how-it-

[21]	Henderson Electric, "How Does Smart Home Automation Work?" Published 2019, [Online]. Available: https://hendersonelectric.com/how-does-smart-home-automation-work/

[22]	Build, "what smart wiring" No Date, [Online]. Available: https://build.com.au/what-smart-wiring/

[23]	Boudellal.M,"Smart, homeHabitat, connecté, 361installations, domotiques, et, multimédia". Dunod, 2014.

[24]	Jatin.P, "WiFi's evolving role in IoT", MAY 11, 2017, P 802.

[25]	Mahmoud.E,Seyed.S,Hon.C ,"Emerging Wireless Technologies in The Internet of Things: A Comparative Study ",International Journal of Wireless & Mobile Networks (IJWMN) , 2016,Vol. 8, No. 5.

[26]	Dinusha.R,"Evaluation, of,wireless,home,automation,technologies", Conference: Digital Ecosystems and Technologies Conference 2011.

[27]	Roy Delgado. A, Picking. R, Grout. V, "Remote-Controlled Home Automation Systems with Different Network Technologies", Paper presented to the 6th International Network Conference (INC 2006) 2006, pp360.

[28]	Rajeev. P," Internet of Things: Ubiquitous Home Control and Monitoring System using Android based Smart Phone "2013.

[29]	Stontronics - Chancerygate Business, "Raspberry Pi 3 Model B ", [Online]. Available: https://www.alliedelec.com/m/d/4252b1ecd92888dbb9d8a39b536e7bf2.pdf

[30]	LastMinuteEngineers,"Insight Into ESP8266 NodeMCU Features & Using It With Arduino IDE", [Online]. Available: https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/

[31]	Blogspot,"Arduino-uno-r3-datasheet", [Online].Available: https://udatasheet.blogspot.com/2019/05/download-arduino-uno-r3-datasheet-pdf.html.

[32]	DFROBOT- drive the future, "Esp32-Cam Datasheet", [Online]. Available: https://media.digikey.com/pdf/DataSheets/DFRobotPDFs/DFR0602_Web.pdf

[33]	Adafruit, "DHT11_Humidity_TempSensor ", [Online]. Available: https://media.digikey.com/pdf/DataSheets/Adafruit PDFs/DHT11_Humidity_TempSensor pdf.

[34]	SeeedTechnology Bendable El Wire, "Grove Gas Sensor MQ-5 ", [Online]. Available:

https://media.digikey.com/pdf/DataSheets/SeeedTechnology/Grove_Gas_Sensor_MQ5_Web.pdf

[35] F. E. Egypt, "Flame Sensor Module Datasheet," [Online]. Available: http://rogerbit.com/wprb/wp-content/uploads/2018/01/Flame-sensor-arduino.pdf

[36] Adafruit, "pir-passive-infrared-proximity-motion-sensor", [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf

[37] Components101, "Rfid RC522 datasheet", [Online]. Available: https://components101.com/sites/default/files/component_datasheet/4x4Keypad Module Datasheet.pdf

[38] Components101, "5V Relay 4x4Keypad datasheet", [Online]. Available: https://components101.com/sites/default/files/component_datasheet/4x4Keypad Module Datasheet.pdf

[39] RandomtutorialsNerdTtutorials, Ruis Sentos, "Oled datasheet", [Online]. Available: https://randomnerdtutorials.com/guide-for-oled-display-with-arduino/

[40] Components101, "4x4Keypad datasheet", [Online]. Available: https://components101.com/sites/default/files/component_datasheet/4x4Keypad Module Datasheet.pdf

[41] LastMinuteEngineers, "Insight Into l293d-motor-driver-shield-arduino ", [Online]. Available: https://lastminuteengineers.com/ l293d-motor-driver-shield-arduino/

[42] LastMinuteEngineers,"Stepper Motor 28-BYJ48 Guide line", [Online]. Available: https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/

[43] LastMinuteEngineers,"Servo Motor Guide line", (june, 2018) [Online]. Available: https://lastminuteengineers.com/servo-motor-arduino-tutorial/

[44] BMC, "OSI Model: The 7 Layers of Network Architecture", [Online]. Available: https://blogs.bmc.com/osi-model-7-layers/?print=pdf

[45] The It Guys,"Smart wiring in smart houses",[Online].Available: https://www.itguyswa.com.au/smart-wiring/

[46] Devopidia,"MQTT Structure and message flow", [Online].Available: https://devopedia.org/mqtt.

[47] Juniper-Research,"Smart-home-statistic",[Online].Available: https://www.juniperresearch.com/infographics/smart-home-statistics

[48] AvSystem, "IoT Standards and protocols guide_protocols of the Internet_of_Things",[Online].Available:https://www.avsystem.com /blog/iot-protocols-and-standards/

[49] SmartBear, "AMQP Protocol", [Online].Available: https://support.smartbear.com/readyapi/docs/testing/amqp.html

[50] Geeksforgeeks, "Internet of things layers Architecture", [Online].Available: https://www.geeksforgeeks.org/architecture-of-internet-of-things-iot/