



TALLINNA TEHNIKAÜLIKOOL  
INSENERITEADUSKOND

Elektroenergeetika ja mehhatroonika instituut

**Traadita võrkude kasutamine koduautomaatika  
süsteemi ja asjade interneti jaoks**

MAGISTRITÖÖ

MEHHATROONIKA ÕPPEKAVA

Üliõpilane:

Admir Hoxha

Üliõpilaskood:

A144754

Juhendaja:

Vu Trieu Minh

Tallinn, 2017



TALLINN UNIVERSITY OF TECHNOLOGY  
SCHOOL OF ENGINEERING

Department of Electrical Power Engineering and Mechatronics

**Wireless Network for Home Automation System  
and Internet of Things**

MASTER THESIS

MECHATRONICS PROGRAM

Student:

Admir Hoxha

Student code:

A144754

Supervisor:

Vu Trieu Minh

Tallinn, 2017

# AUTHOR'S DECLARATION

Hereby I declare, that I have written this thesis independently.

No academic degree has been applied for based on this material.

All works, major viewpoints and data of the other authors used in this thesis have been referenced.

Thesis is completed under the supervision of **Prof. Vu Trieu Minh.**

25 May 2017

Author: Admir Hoxha

  
.....  
/signature/

Thesis is in accordance with terms and requirements

“.....” ..... 201...

Supervisor: Prof. Vu Trieu Minh

.....  
/signature/

Accepted for defense

“.....” .....2017

Chairman of ..... theses' defense commission: .....

/signature/

TTU

School of Engineering

### THESIS TASK

Student: Admir Hoxha (A144754)  
Study program, MAHM02/13  
Specialty: Mechatronics  
Supervisor: Professor, Vu Trieu Minh  
Consultants: None

#### THESIS TOPIC:

(In English) Wireless Network for Home Automation System and Internet of Things

(In Estonian) Traadita võrkude kasutamise koduautomaatika süsteemi ja asjade interneti jaoks

#### Work tasks to be performed as per timetable:

No.	Description of tasks	Completion date
1.	Research on Wireless home automation systems and hardware use, learning about sensors used and to build connection with the microcontroller.	01/03/2017
2.	Learning in deep about microcontrollers and their application on the system.	20/03/2017
3.	Building a webserver control interface compatible with smartphones, PC's and tablets.	03/04/2017
4.	Building the real system and simulating real data of the system.	17/04/2017
5.	Building a data-storage internet database and preparing the thesis report.	22/05/2017

#### Engineering and economic problems to be solved:

The Wi-Fi module which powers and controls the sensor is powered using a power line. It would be less complex, more flexible and space saving if it were powered via battery. The module is not very battery friendly, therefore it needs low-power mode program configuration.

**Language of the thesis:** English

Deadline for submission of the application for defence: 15/05/2017

**Deadline for submission of the thesis: 25.05.2017**

Student: Admir Hoxha

  
.....  
/signature/

“25” May. 2017

Supervisor: Vu Minh Trieu

.....  
/signature/

“25” May. 2017

## Contents

Preface .....	7
1 INTRODUCTION.....	9
2 MOTIVATION.....	10
3 BACKGROUND AND RELATED WORK .....	11
3.1 Choosing the Wireless Technology.....	11
3.1.1 Wi-Fi Technology .....	12
3.1.2 ZigBee Technology .....	12
3.1.3 Radio Frequency Identification (RFID) Technology.....	13
3.1.4 GSM based Home Automation System .....	13
3.1.5 Bluetooth.....	15
3.1.6 Power consumption.....	15
3.1.7 Security .....	16
3.1.8 Error control and reliability .....	18
3.1.9 Conclusion.....	19
3.2 Choosing the Architecture .....	21
4 HOME AUTOMATION SYSTEM ARCHITECTURE .....	23
4.1 MQTT .....	24
4.2 Node-RED.....	26
4.3 Hardware used on the System.....	27
4.3.1 Sensors.....	27
4.3.2 Other tools.....	29
5 CONSTRUCTING THE HOME AUTOMATION SYSTEM.....	33
5.1 Configuring RPI .....	33
5.2 Configuring esp-e12 NodeMCU.....	35
5.2.1 Controlling power outlets using esp-e12.....	38
5.2.2 Controlling temperature sensor .....	46
5.3 Controlling the Gas sensor and synchronization with the buzzer and LED light .....	50
5.4 Establishing motion detection system .....	57
5.5 Smart home event triggering.....	61
5.5.1 Triggering email notification events.....	61
5.5.2 Controlling outlets based on the temperature sensor.....	64
5.5.3 Controlling outlets based on the luminosity sensor.....	65
5.5.4 Triggering time-based events.....	68
6 ACCESSING THE SYSTEM OUTSIDE THE HOME NETWORK.....	70

6.1 RPI Static IP .....	70
6.2 Port forwarding our RPI .....	70
7 STORING AND ANALYSING THE SYSTEM DATA.....	72
7.1 Storing data using Node-Red.....	72
7.2 Storing data using cloud .....	74
7.3 Storing data using Matlab .....	79
8 FINAL PROTOTYPE .....	82
9 KOKKUVÕTE .....	86
10 SUMMARY .....	87
11 REFERENCES .....	88
12 APPENDICES.....	91

## **Preface**

The general idea of the thesis was taken from Professor Vu Trieu Minh thesis topic that was declared on the website of the University. The topic consisted on wireless networks and applications were the student task was to study the wireless network applications from remote sensors, microcontrollers that can be monitored and controlled from PC and smartphones. Then I shaped the topic into a more concrete system.

I would like to thank Mr. Vu that supervised me on the topic with motivation, practical ideas and support during the entire period of the thesis work that helped me to achieve the finalization of the thesis.

I would also like to thank the Mechatronic department staff especially Professor Mart Tamre, Chair of Mechatronics who was always available and supportive during the whole period of the master studies. I appreciate his will and satisfaction of listening the student's ideas and always trying to give the best solutions for them.

And I would like to thank Tallinn University of Technology that gave me the opportunity to experience wonderful times of seeking knowledge, wonderful student life, and a descend study with European standards.

## EESSÖNA

Käesoleva töö üldine idee on võetud professor Vu Trieu Minh poolt antud lõputöö teemast. Antud töö koosnes juhtmevabadest võrkudest ja nende rakendustest. Ülesandeks oli uurida juhtmevabadete võrkude rakendusi kasutades kaugsensoreid ja kaugelt juhitavaid ning jälgitavaid mikrokontrollereid kasutades arvutit ja nutiseadmeid. Uurimusest saadud info põhjal arendati välja ka konkreetne süsteem. Soovin tänada hr. Vu-d, kes juhendas mind antud töö tegemisel. Lisaks soovin tänada hr. Mart Tamre-t toetuse eest õpingute ajal. Soovin tänada ka Tallinna Tehnikaülikooli saadud teadmiste, suurepärase tudengielu ja Euroopa standarditele vastava hariduse eest.



# 1 INTRODUCTION

With the continuous advancement of technology, the society approaches to more wide opportunities. And we notice this starting from our homes, where technology and Internet of Things (IoT) offer us possibilities to make use of many kinds of systems that ease our life and save us time. Knowing the fact that nowadays our society has reached a point where almost every person is equipped with a smartphone or a PC, it has facilitated the development and improvement of many home automation systems. The system that will be treated on this thesis consists of building a smart home system that will be able to monitor the risk factors of the home and also control many home equipment when the owner is outside. For instance, the system will detect if there will be any fire in the house, smoke or suspicious gases using a gas sensor, it will measure the temperature and humidity level of the environment for any possible abnormalities caused from many factors like short circuit or any electric equipment. The system will detect motion in cases of intrusions and make it possible to trigger a buzzer alarm event to scare the intruder and alarm the neighbors. It can detect the vehicle garage whether it is open or closed. Also the light intensity of the room will be measured for security and other reasons.

In principle, the system will ensure that all the important information will be gathered in real time and it will trigger events in cases of any unusual detections. The system user will be able to monitor the home using any web-browsing enabled device (smartphone, PC, Tablet, etc.) and will be notified via email in cases of any suspicious detection. The system offers also a possibility to turn on/off any desired home electrical equipment by using wirelessly controlled power sockets, including home lights etc.

## 2 MOTIVATION

Nowadays the technology is developing so fast that the electronics are getting faster and cheaper every day. There are for sure many already developed home security automation systems able to easily cover all the features of the system we mentioned above but most of leader companies that offer such systems are using proprietary technologies and not open source, for instance using their own servers, hardware etc., some use even their own wireless technology thus increasing the cost of these systems.

With the fast grow of wireless home automation technologies and the Internet of things, more and more work is addressed on these systems, and nowadays there are already many free open-source ways to build such kind of systems and using cheap market technology, thus reducing considerably the cost of these systems, and this is where the work of this thesis is going to focus on, and this is the reason I got motivated.

The aim of the thesis is to build a home security automation system which could fulfill all the requirements to keep a home safe and easily controllable. The system should be cheaper to build and simple than most of the already developed systems. The aim is to save money and time on building it and to make it as flexible as possible. One detail that increases the flexibility of the system is the wireless technology. Using wireless sensors instead of cable wirings reduces significantly the cost of installation, and also wireless allows placing sensors where cabling is not aesthetic, conservatory or safety reasons. The exact physical location of the sensor is no longer crucial for a connection, as long the sensor is in reach of the network.

Using only a small Wi-Fi chip like esp8266 that costs less than four dollars, we can connect to it many sensors, and it allows us even to program the chip and process the sensor data, and send them to the main server. We can also connect many sensors in just one of those chips and send all of the information to the main serve, thus reducing the cost of expensive hardware use.

The thesis project will use many sensors as mentioned above, the data will be sent wirelessly using esp8266 Wi-Fi chip to the Raspberry pi (RPI), the latter will serve as the main server. The RPI will be accessible all-over the world from any web-browsing enabled devices thus making it flexible to access. A user interface will be designed for the system making it user friendly for the client, and accessible by using just an IP address thus making it simple to access for any portable device, including android smartphones and iPhones. Also the electric-

equipment commands will be reflected in the user-interface dashboard, thus making our home remotely accessible from anywhere.

### **3 BACKGROUND AND RELATED WORK**

*Keywords:* Wireless automation security systems, smart home using node-red and mqtt, Smart home using Wi-Fi, Home automation esp8266, Home automation raspberry pi, home automation systems.

Wireless home automation (WHA) in general are networks composed of wireless sensors that are interconnect with each other through a wireless architecture. Recently there have been developed many wireless technologies targeting WHA, hence selecting the optimal technology is challenging. We will make a general overview of the developed wireless technologies used in WHA and discuss the advantages and challenges.

#### **3.1 Choosing the Wireless Technology**

Smart home devices nowadays can be accessed, monitored and controlled using different built-in wireless communication technologies such as ZigBee, Wi-Fi, RFID, Bluetooth, GSM with optional general packet radio service (GPRS) and Z-wave with its series like INSTEON, Wavenis, EnOcean and UWB.

One of the most important criteria of Choosing the right wireless connectivity technology for our IoT application is the final cost of the technology, and since the Z-wave and its series including INSTEON, EnOcean and Wavenis are proprietary technologies and not opensource, this automatically increases the costs of using such systems, therefore we will exclude this technologies from our scope of research.

According to many scientific articles [2] [3] [4] [5] [8], the predominant wireless connectivity technologies in the market are ZigBee, Wi-Fi, Bluetooth, RFID and GSM, which are most widely used wireless communication protocols in home area network.

### 3.1.1 Wi-Fi Technology

Until recently, using Wi-Fi technology to home appliances with little processing power was not cost effective nor possible. Today, most of devices that are coming out on the market embed the Wi-Fi technology. These technologies enable wireless Internet connectivity with the smallest microcontroller (MCU).

*Advantages of Wi-Fi in HANs:*

- Highly secured connection which uses 128-bit AES encryption [26].
- Wi-Fi technology can be adopted to power lines, coaxial cables and phone lines [26].
- Because it supports TCP/IP thus it does not require special gateway [27].
- Wi-Fi has a short delay latency less than 3ms.

*Disadvantages of Wi-Fi in HANs:*

- Wi-Fi consumes significant power when compared to ZigBee.
- The battery lifetime can reach till 5 days without charging [28].
- It gets affected by electromagnetic radiation caused from household appliances which affects the speed of transmission [29].

### 3.1.2 ZigBee Technology

*Advantages:*

- Highly secured connection using the 128-bit AES encryption [26].
- The battery lifetime can reach a range from 100-1000 days without charging [26]-[31].
- ZigBee network is reliable since it avoids collision, conflict and competition between nodes [13].
- ZigBee has a delay latency ranging from 15ms to 30ms
- ZigBee protocol has self-organizing networking capabilities [27].

*Disadvantages:*

- Setting up a ZigBee network requires additional devices which increases cost.
- Appliances running ZigBee are incompatible with other network protocols such as Wi-Fi.

- ZigBee has low data transmission rates and lacks Internet Protocol support [27] which means that it has no build-in TCP/IP stack which makes it necessary to use additional devices for internet connections.

### 3.1.3 Radio Frequency Identification (RFID) Technology

RFID is a bi-directional radio frequency identification system which is used in forms of tags and readers that can be interfaced to handheld computing devices or personal computers. It can coexist with other technologies such as ZigBee and Wi-Fi. RFID operates under frequencies from 120 kHz - 10 GHz and the detection distance can vary from 10 cm - 200 meters [2]. RFID can be used in home area network applications like locking doors and controlling lighting.

*Advantages of using RFID in HANs:*

- RFID has high transmission speed and a latency of around 100ms.
- RFID allows bidirectional communication.

*Disadvantages of using RFID in HANs:*

- RFID chips are expensive.
- RFID supports relatively low bandwidth as compared to Wi-Fi; hence it cannot be used for multimedia and download applications.
- Collisions can occur when two tags in the same area try to communicate to the reader. Both tags might be unable to respond to the reader.
- RFID's accuracy depends on line-of-sight communication.

### 3.1.4 GSM based Home Automation System

In the figure one it is shown a common system based on GSM network via SMS [4]. It is used to control the home appliances using an Arduino board. It uses certain peripheral drivers and relays to achieve this interfacing. The smart phone is the user interface device. The system uses the 'App Inventor' visual programming tool to develop the android application. The application generates SMS messages based on the user commands and sends them to the GSM modem attached to the Arduino board. This way the user can control the home appliances.

Sure there is many kinds of GSM architecture systems that are used on home automation but in general they show this characteristics results.

*The advantages:*

GSM is a SMS based system and it can establish connection in places where there may not be proper internet connectivity.

The control of home appliances is done primarily through SMS codes. AT commands can be sent through the GSM network and this controls the home devices. Messages are sent by the device to the user through SMS as well.

*Disadvantages*

The SMS service can incur additional costs which is a considerable drawback. This system has the drawback of not being able to program the devices in same way that an esp8266 WI-FI chip can do. Also SMS depends on the networks and there is a possibility of delayed delivery, so basically it relies heavily on the SMS, which is not very fast and dependable.

In most cases the GSM communication requires a password based authentication system to be used. The text messages sent through GSM module will contain the password which is used to ensure the message is sent from a recognized trusted source therefore the security of the system is compromised since passwords are sent freely over the network.

The GSM technology is also not flexible in the sense that it cannot easily fit into an existing system.

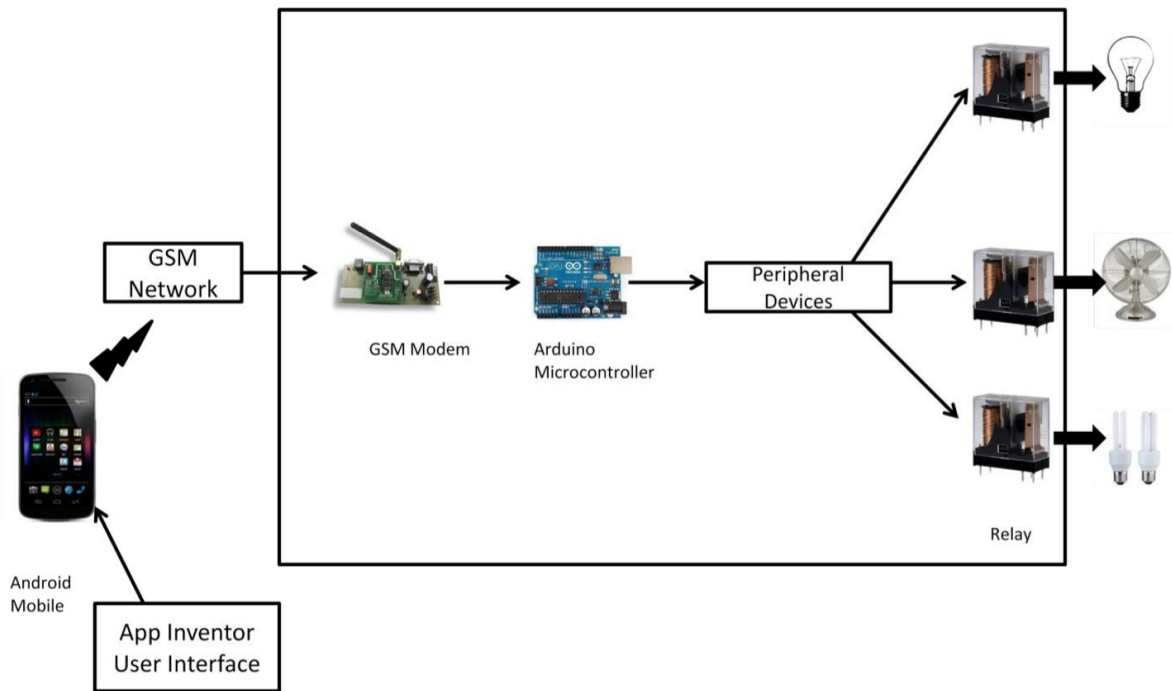


Figure 2. GSM based home automation system [4]

### 3.1.5 Bluetooth

Bluetooth is a cheap connection technology and has the capabilities of being able to fit onto an existing system.

Some of the drawbacks of this technology consist on not providing energy conservation tips. Real time access cannot be achieved. Anywhere access to the devices cannot be achieved. This system also suffers from the drawback of the range of Bluetooth being around 10 meters only.

### 3.1.6 Power consumption

The energy saving is one of the main reasons for the emergence of smart home automation concepts. Most wireless autonomous devices are usually battery-powered. Therefore it's essential to manage the smart devices to best utilize the scarce power resources over long time.

Our project won't require battery-powered devices since the availability of the home electrical power sockets will be used, therefore the cost of the power consumption of the system will be almost inconsiderable [8].

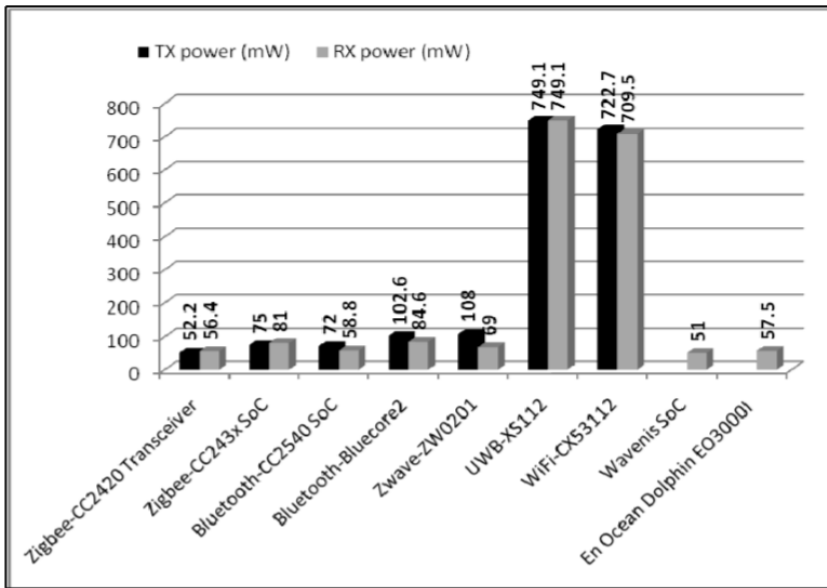


Figure 3. Power consumption of the many Wireless communication technologies [5]

### 3.1.7 Security

Talking about the security of the smart home system we should first describe the general architecture of the system in order to specify the vulnerable points of it.

In practice, a smart home system is a combination of sensors or devices controlled by a small computer that one installs on his house and connects it with the home router. After that we can remotely control the devices using smartphones or PC via internet. While we could use such setup in many areas, the most common ones are:

- Light Control
- House kitchen equipment's
- Alarm systems etc.

The common architecture of the smart home systems is shown in Figure 4.



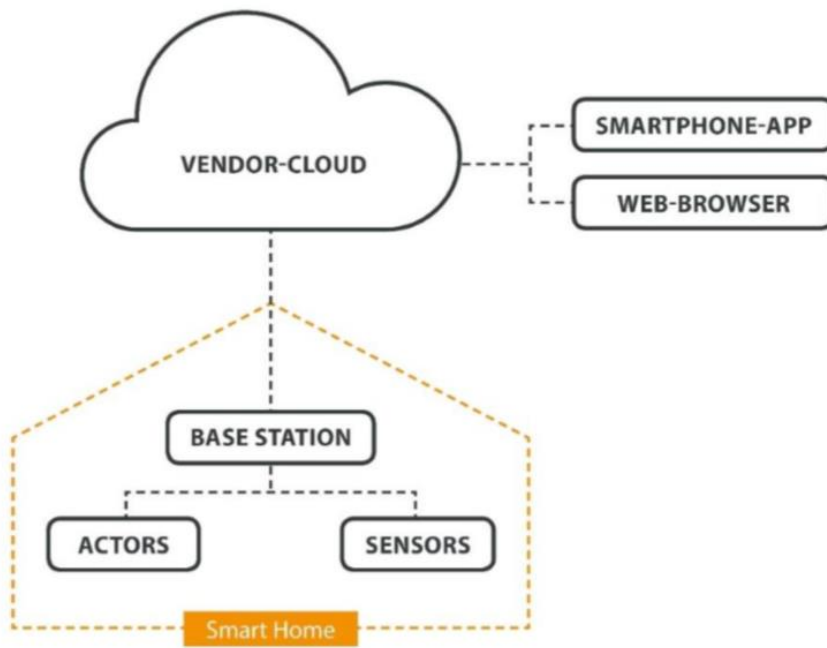


Figure 4. Basic elements of a smart home system [6]

There are sensors with specific tasks and there are also devices which can change the condition of the home. This sensors and actors are connected to a central base station which manages the information. The base is accessible remotely from a smartphone or a PC. This connection can either be established within the home network or using a cloud-interface provided by the vendor in order to enable remote access to the system from anywhere.

- **Protocols in use**

The architecture mentioned above can be divided into two parts.

The first part is the communication between the main station and the control client, in most cases realized via the cloud. This is the interface between the smart home system and the user. The web-area, is the area in which many web protocols e.g. HTTP, Servers WebSockets etc, are used to establish connection between the base station and smartphones or web-browsers.

The second part is the communication between the base station and the sensors and other devices. Many smart home communication protocols exist in this area, which can be wireless or cable-based and also mixed. This area is referred as the “local area”.

The web area is vulnerable from the traditional well-known web threats, which any one of us are exposed to by using our web-browsing devices and smartphones therefore we will consider this section of the security risks as understandable and not unusual vulnerability to the smart home system.

The vulnerabilities depended on our wireless technologies are focused on the “local area” of the system.

We will mention some of the wireless technologies that are used to the local area that use the Advanced Encryption Algorithm (AES). AES is a symmetric key encryption algorithm which offers three key sizes [5]. When we talk about the security methods of wireless solutions,

- Bluetooth performs encryption using E0 algorithm (Bluetooth-BR/EDR), and 128bit AES (Bluetooth low-energy).
- The Z-Wave200/300 series chips do not support security, the 400 series chip offer 128-bit AES encryption.
- WiFi, UWB and Zigbee use AES block ciphers

Since the WI-FI and ZigBee are using AES, the security mechanism is considered quite good [6].

- **MQTT, Protocol the safest protocol**

The existing protocols e.g. HTTP, CoAP were previously used the most in smart homes and proved to be less reliable considering security of data [32].

MQTT protocol requires low bandwidth, battery and less resources compared to HTTP. It also provides security for the transmitted data, faster response and throughput making it very suitable for home automation.

Data transmission using MQTT will become more reliable and prominent is usage of source.

Furthermore, MQTT has the advantage of overcoming firewalls.

### **3.1.8 Error control and reliability**

Reliable delivery of information is critical in smart home networks to obtain a steady hime automation control. Error detection and correction enable identification and correction of the corrupted data in unpredictable communication channels. Zigbee, WiFi, and Bluetooth use cyclic redundancy check (CRC) [24, 25] to validate the packets. Here a certain number of bits (checksum) are appended to the message being transmitted. Different technologies use 16, 8, 32 checksum and higher the checksum, more powerful error controls.

WiFi has the highest CRC error control proficiencies by offering 23 checksum, whereas Zigbee, and Bluetooth use 16 checksum.

### **3.1.9 Conclusion**

UWB and Bluetooth technologies exhibit lower range transmission coverage of 10 meters, hence not very suitable for WHA networks where the appliances could be located in long distances. They also support only 8 nodes of network protocol size thus limiting the automation system.

The ZigBee technology is one that can compete the most with the WI-FI technology but since it suffers from [5] disadvantages we mentioned above that disqualify it which consist on higher costs to setup a similar network because it requires additional devices.

For instance, in order to achieve a transceiver system between a sensor and the internet, we would need to use two ZigBee transmitter/receiver modules because they don't support TCP/IP stack thus making necessary another device to achieve that function.

Besides, the ZigBee modules cost a lot more, compared to WI-FI modules on the market.

Another drawback of ZigBee technology is that it has low data transmission rates with maximum size of 250 kb/s.

We also excluded RFID technology for the reasons of high cost of the RFID chips, and low bandwidth. And also a major drawback of RFID is that when tags in the same area try to communicate to the reader they can collide thus both of them might be unable to respond to the reader.

After we mentioned all the important wireless technologies used in today's home automation systems, we came to a conclusion that for our project requirements we consider WI-FI technology as more suitable due to the ready availability of Wi-Fi networks in a house-hold setup.

- It's an open publicly accessible standard, therefore a lot of support and projects can be found on the web, thus also leading to an enormous available cheap technology on the market.

- It has a high range of data transfer that can reach a bandwidth starting from 54 - 300 Mb/s which suits our needs for future camera surveillance option.
- It is flexible and can be easily adapted to adding additional features.
- It does not require a special gateway because it inherits the Internet protocol support capability and features.
- It has short delay
- It is highly secured connection
- It has a high range data transmission coverage up to 100 meters.
- It is reliable and trusted since it has the highest error control proficiencies.

We considered the fact that WI-FI suffers from some disadvantages which are related to high power consumption and interference but for our project this disadvantages represent no considerable importance.

- **High power consumption:**

It poses a problem in cases of powering the WI-FI system via battery, but since our project will be a home project, the availability of permanent power source won't be a problem.

Our chosen WI-FI technology will be the esp8266 e12 Wi-Fi module which costs less than four dollars on the market and it also has three low power modes (Light sleep, modem sleep and deep sleep) features [7] of reducing the power consumption making it also suitable for battery-powered projects.

- **Interference**

Interference may be caused from presence of microwave ovens, codeless phones and other wireless technologies. Based on some studies [9] the interference of our system will not be a serious problem because of the non-presence of interference devices in the home because the system will be working when the owners will be outside, and the presence of interferers outside the home will be neglected and considered harmless.

## 3.2 Choosing the Architecture

While we chose the WI-FI technology as a WHA connection tool, we are aware that many kinds of WHA architectures are already implemented, and we will make a general review of some researched articles and discuss about the advantages and disadvantages.

The WHA architecture mentioned in the [10] and [13] are able to control lights and other electrical equipment using relays that are controlled using wire connection with arduino microcontroller. The disadvantage of this system is that it is using wired connection to control equipment's, which makes the system not flexible and increases the costs whereas in the article [11] gas and fire sensors are connected directly to arduino mini, and from arduino the data is sent using esp8266 standard module in online server. The idea of our project is to reduce costs by removing the need of using an arduino or similar microcontroller technology for processing the sensor data, instead we use a more integrated esp8266 e12 module on NodeMCU board [12] which could easily replace the need of using the arduino mini board, thus reducing the cost of the project and making it more compact. This practice would reduce considerably the cost of the project when we want to include many sensors.

The most considerable work from cost perspective is the [33] article in which MQTT protocol is used to send temperature signal to the web-server through a cheap esp8266 Wi-Fi module, making it one of the most cost effective systems, but since it is using a server to store data and host a web domain for user interface, it makes the system less flexible when it comes to more complex home automation systems that require many tasks and smart event triggering based on sensors and actors. For such complex systems Node-RED comes in hand as it gives a wide area of possibilities to trigger events and such. More on Node-RED will be explained in another section.

The architecture of the WHA that we have decided to build is motivated based on the cost perspective, flexibility and complexity. It is inspired from a book [1], which discusses many project related to IoT and smart home systems using RPI and other technologies.

The WHA will be in detail explained in the next section.

Protocol	Zigbee	Z-Wave (Zensys Corp)	EnOcean	UWB	Bluetooth	Insteon (SmartLabs, Inc)	Wavenis (Coronis System)	Wi-Fi	6LoWPAN
IEEE Standard	802.15.4	-	-	802.15.3a*	802.15.1	-	-	802.11a/b/g	802.15.4
Frequency Band	2.4 GHz, 915 MHz, 868 MHz	868/908MHz, 2.4 GHz (400 series only)	868 MHz	3.1-10.6 GHz	2.4 GHz	904 MHz	433 /868/ 915	2.4 GHz; 5 GHz	2.4 GHz, 915 MHz, 868 MHz
Data Rate	20/40/250 Kb/s	9.6Kbps/40Kbps, 200 kb/s	125 kbit/s	110 Mb/s	1 Mb/s	38.4 Kb/s	4.8/19.2/100	54 Mb/s	20/40/250 Kb/s
Modulation	BPSK/BPSK/O-QPSK	FSK /GFSK	ASK	BPSK, QPSK	GFSK	FSK	GFSK/PSK	B/QPSK, COFDM, QAM	BPSK/BPSK/O-QPSK
Spreading	DSSS	No	No	DS-UWB, MB-OFDM	FHSS	No	FHSS	DSSS, CCK, OFDM	DSSS
Communication Range(m)	10-100	30 (in) 100 (out)	30 (in) 300(out)	10	10	45 (out)	200 (in) 1000 (out)	100	10-100
Security	AES	AES-128	Basic	AES	E0 Stream AES-128	Rolling codes, public-key	3 DES 128AES	RC4 Stream / AES Block	AES
Error Control/Reliability	16-bit CRC, ACK, CSMA-CA	8-bit CRC, ACK, CSMA-CA	-	32-bit CRC CSMA-CA	16-bit CRC	8-bit checksums	BCH (32,21) FEC	32-bit CRC	16-bit CRC, CSMA-CA, ACK
Network Size	64000	232	2 <sup>32</sup>	8	8	256	NA	2007	2 <sup>64</sup>
Internet connection	Gateway Required	Gateway Required	Gateway Required	Gateway Required	Gateway Required	Gateway Required	Gateway Required	Gateway Required	Gateway NOT required
Logistic	Standard	Proprietary	Proprietary	Standard	Standard	Proprietary	Proprietary	Standard	Standard

Table 1. The characteristics of wireless technologies in WHA [5]

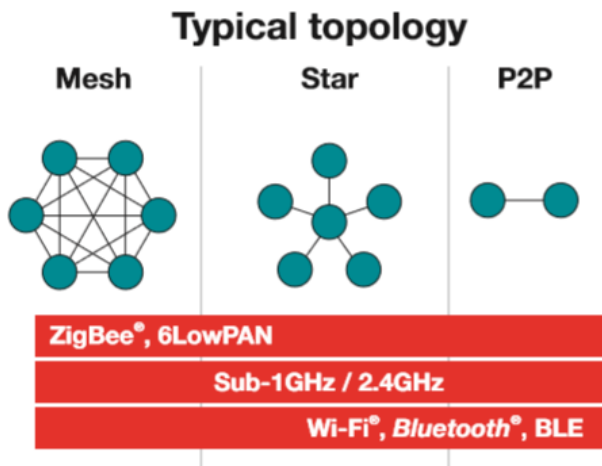


Figure 5. The topology of the wireless technology connections [3]

## 4 HOME AUTOMATION SYSTEM ARCHITECTURE

The data of the sensors that we will use, will be gathered on the esp8266 e12 WI-FI module (esp-e12) and from there they will be wirelessly transferred through home WI-FI router to the internet as shown in Figure 6. In the same time we have a RPI connected through the same router to the internet, thus the data of the esp-e12 is going to be transferred to the RPI using MQTT communication protocol. The RPI will be the main server where the data will be stored and managed, and the user interface dashboard will be built in RPI also using open-source Node-Red tool which will be accessible from the use of the web-browser.

The system will be a two-way communication, meaning that it will give the opportunity to the user to remotely control the sensors. Our system will offer the opportunity to turn on and off home electrical equipment through wirelessly controllable power sockets, and also the system will be able to notify the user by email in case of any sensor unusual detection. More of the system in detail will be explained later on this report.

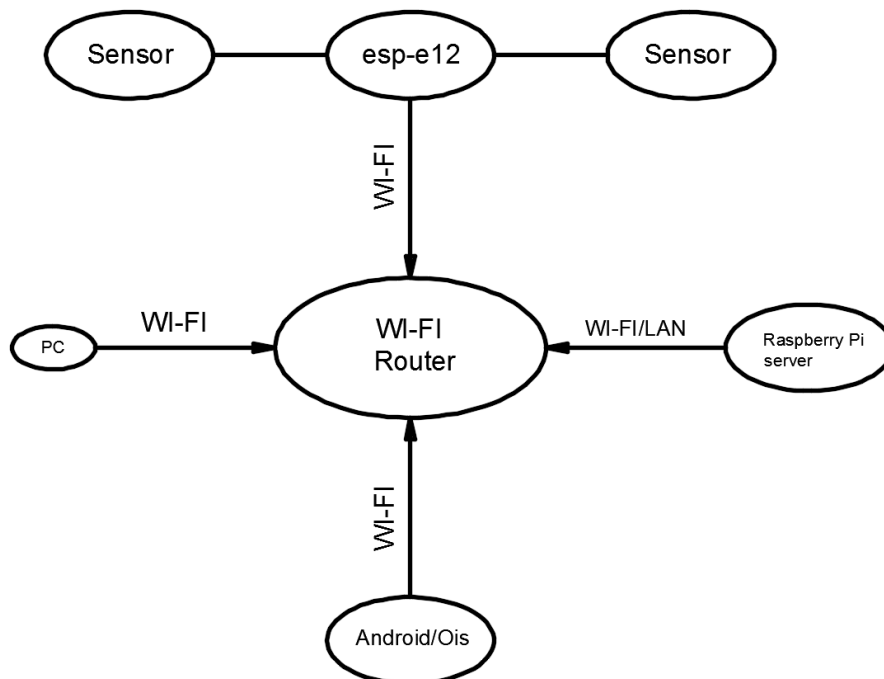


Figure 6. The schematics of WHA

## 4.1 MQTT

MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) context where a low bandwidth is required.

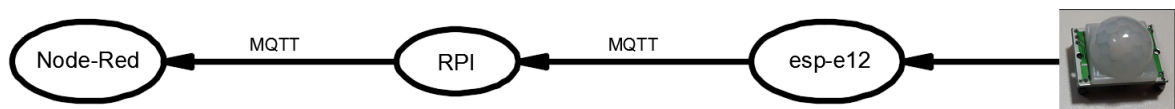
The protocol runs over TCP/IP, or over other bi-directional connections.

Basically, it is a protocol with which you can publish and receive messages as a client and it facilitates connection between multiple devices.

With MQTT we can send a command to control an output:



Or we can read data from a sensor and publish it:

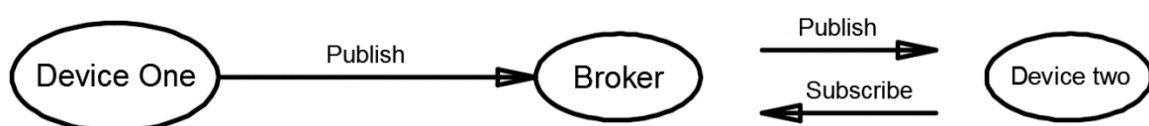


There are four basic MQTT concepts that we should mention:

- Use of the publish/subscribe message pattern

Here a client communicates directly with an endpoint. Pub/Sub decouples a client who is sending a particular message (the publisher) from another client (or clients), who is receiving the message (called subscriber). This means that the publisher and subscriber don't know about the existence of one another. There is a third component, called broker, which is known by both the publisher and subscriber, which filters all incoming messages and distributes them accordingly.

Basically, a device can publish messages to our device. Or our device can subscribe to a particular topic to receive the messages.





Therefore, the device two should be subscribed to the same broker where the device one has published a topic, only this way the message is received.

- Messages

Are basically the information we want to send between devices, like data or commands.

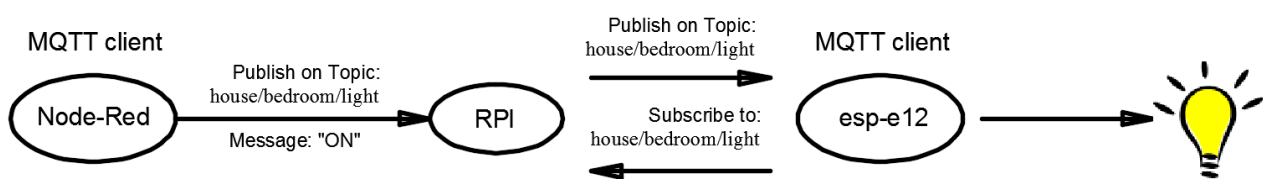
- Topic based filtering of the messages

Topics are the method we want to specify where we want to publish our messages. Topics in Node-Red are shown with strings separated by slashes, the latter indicate the topic level.

For instance, if we want to turn the home light ON, assuming we are using our project, we should publish an “ON” message to a topic on Node-RED:

“house/bedroom/light “ where “house” and “bedroom” are topic levels.

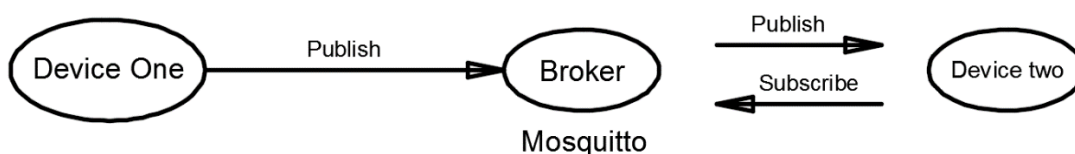
And in order for the light to turn ON, our esp-e12 should be subscribed to that topic in order to receive the “ON” message.



- MQTT Broker

The broker does the management role on the connection, it has the responsibility of receiving all the messages, to decide who is interested in the message, and to publish the message to all subscribed clients.

There are many MQTT brokers, and for the RPI the most common broker used is the Mosquitto broker. It is open source and simple to install.



## 4.2 Node-RED

Node-red is an IBM open-source tool that allows us to wire together different devices and build Internet of Things (IoT) applications with the focus on simplifying the programming part of the system application. It provides a browser based interface that makes it easy to wire together flows using the wide range of nodes in the palette that can be dragged and connected as desired.

The RPI runs Node-Red perfectly [15] [16]. The advantage of Node-Red is that it simplifies the things by allowing the user build complex systems using a basic interface, avoiding the unnecessary time-consuming code programming. With the node-red we could have access to our RPI GPIO pins to control inputs/outputs, establishing MQTT connection with other boards like arduino, esp8266 etc., receiving data from the web, create time-triggered events, store and receive data from a database [15].

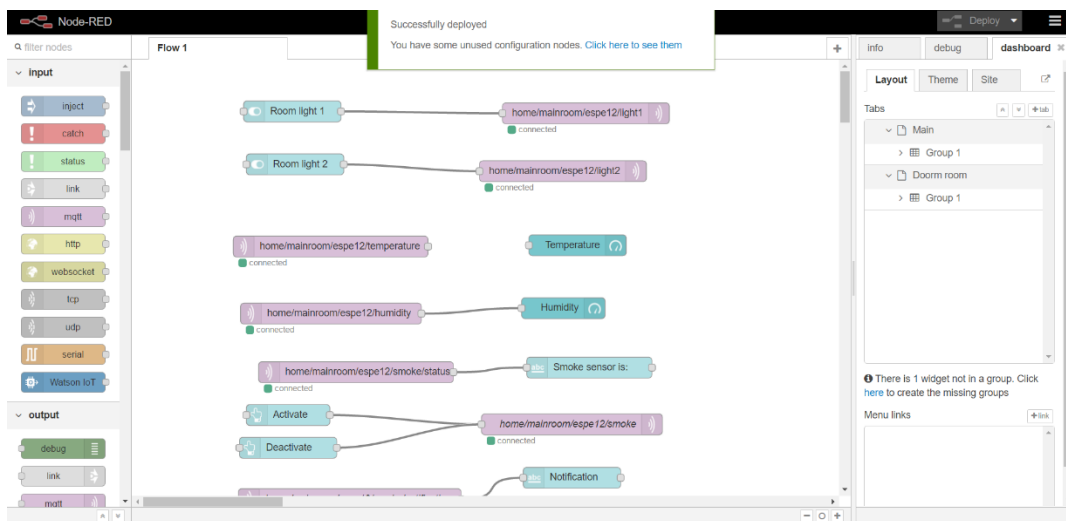


Figure 7. A preview of the Node-Red web-interface.

## 4.3 Hardware used on the System

### 4.3.1 Sensors

- **Gas sensor (MQ2)**

The Grove - Gas Sensor(MQ2) module is useful for gas leakage detection (home and industry). It is suitable for detecting H<sub>2</sub>, LPG, CH<sub>4</sub>, CO, Alcohol, Smoke or Propane. Due to its high sensitivity and fast response time, measurement can be taken quickly.

The sensor will be used to detect possible electrical short circuit incidents that could happen in the house that could cause fire, or any kind of smoke casing phenomenon.



Figure 8. MQ-2 Gas sensor

- **PIR Motion sensor**

PIR stands for Passive Infrared. This motion sensor consists of a lens, an infrared detector, and supporting detection circuitry. The lens on the sensor focuses any infrared radiation present around it towards the infrared detector. Our bodies generate infrared heat which serves as a detection source for PIR sensor. The sensor outputs a 5V or 3.3V (based on the pin we use for supplying power) for a period of one minute as soon as it detects the presence of a person. It usually comes with one pin for 5V and one for 3.3V power supply which we are going to use. It offers a tentative range of detection of about 6-7 m and is highly sensitive.



Figure 9. PIR motion detector

- **DHT11 Temperature and Humidity Sensor**

The sensor will be used to track humidity and temperature in our home.

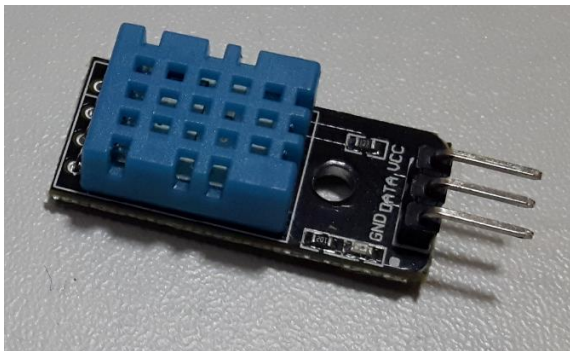


Figure 10. DHT11 Temperature and Humidity sensor

- **Light-dependent Resistor (LDR)**

The resistor will be used to track the light intensity on the house. The light sensor can be used to trigger different events e.g. turning on the lights when sunset time reaches, or any other task of this kind.

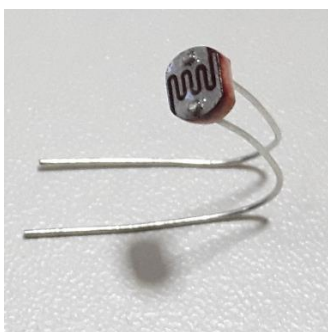


Figure 11. LDR sensor

### 4.3.2 Other tools

- **Wirelessly Controlled Power Socket**

The power socket is able to be controlled remotely using radio frequency (RF) of 433MHz. There are also other options of similar technology like “Lamp Bulb Holders” that are controlled similarly making possible to directly control the lights of the house.



Figure 12. Radio controlled switch set

- **One 433MHz RF Receiver and two 433MHz RF Transmitter**

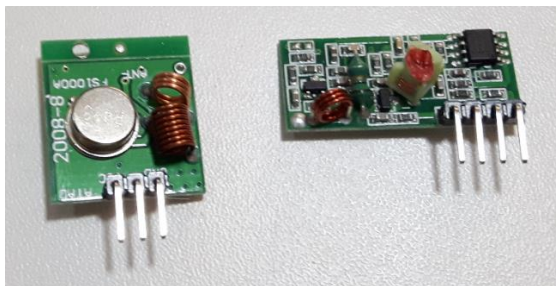


Figure 13. 433 MHz Radio Frequency Transceivers

- **Buzzer**



Figure 14. Buzzer alarm

- **Raspberry Pi 3 model B**

The RPI in our case is considerable expensive but we can use for our project a much cheaper version like the first model (RPI model B) which we can find it on e-bay for less than 20 dollars.

- A 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Camera interface (CSI)
- Bluetooth 4.4
- Bluetooth Low Energy (BLE)
- 1GB RAM
- 4USB ports
- Full HDMI port
- Ethernet port
- Audio jack 3.5mm
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core



Figure 15. Raspberry PI 3 model B

- **The ESP 8266 E-12 (Node MCU)**

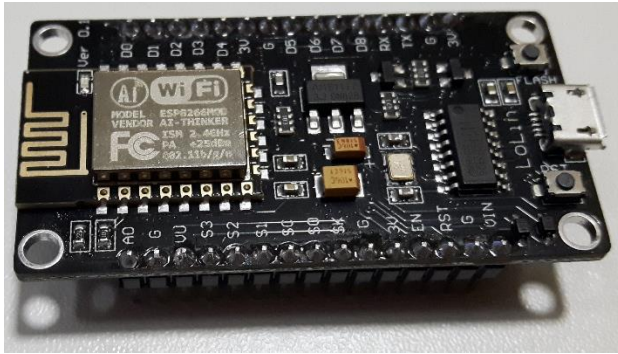
The esp-e12 [14] is a WI-FI offers a complete and self-contained Wi-Fi networking solution; it can be used to host the application or to offload Wi-Fi networking functions from another application processor, it allows to create web-servers, sent HTTP requests, establish MQTT communication, control outputs, send e-mails, etc. And all this with minimal costs and energy used. It costs less than six dollars and since it is not board friendly [12], we will use the esp-e12 integrated in NodeMCU model, which is an interface firmware and open source hardware board widely implemented in esp modules. It uses a CP2102 TTL to USB chip for programming and debugging the module, it is breadboard friendly and it can power the module directly via its micro USB port. Our esp-e12 NodeMCU costs around 6 dollars.

One important advantage of this module is that it can be programmed and debugged the same way as Arduino boards using open soure Android integrated development environment (IDE).

- 11 b/g/n protocol
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Built-in low-power 32-bit CPU
- SDIO 2.0, SPI, UAR



*Figure 16. Esp8266 E12 Wi-Fi module*



*Figure 17. Esp8266 E12 Wi-Fi module incorporated in NodeMCU V3 board*

We will going to use only one of this module since it has 11 GPIO pins and it allows us to use all the sensors of our project.

- **Role in the project**

The esp-e12 will play the role of a microcontroller meaning, it will gather data from different sensors, and it will process them by making comparison to the set-values of the program that we will upload in it using Android IDE, after it will transfer this data through Wi-Fi on the internet.



## 5 CONSTRUCTING THE HOME AUTOMATION SYSTEM

### 5.1 Configuring RPI

We are going to install the raspbian Lite as an operation system (OS) in the sdcard of RPI.

In [17] we can download the latest OS for RPI, and we chose Raspbian lite OS because it is the lightweight version of the Raspbian, and it does not have a graphical user interface (GUI) installed. Therefore it does not contain any unnecessary installed software for our project, making it lightweight solution for our project. The OS installation is done using a “Win32 Disk Imager” application and OS image file as described in [17].

- **Enabling remote communication SSH**

SSH will make our RPI remotely controllable, for this we download in [18] the SSH blank file and we move this file to our prepared sdcard.

Now it is time to boot up our RPI. We firstly insert our prepared microSD card to it and we connect our RPI to the internet using Ethernet cable. We power on the RPI using its power adapter, and the RPI will boot up.

Now we need to find the IP address of our RPI in order to remotely control it. To do this we log-in to our Router settings, and there we can find the IP by checking the user client’s area where all users of the network are listed.

- **Connecting to RPI via SSH**

SSH stands for secure shell, which establishes a remotely secured control connection, and all data sent through SSH is encrypted. For our case, we are using a windows operated machine thus we have to install the PuTTY software (putty.org).

Basically, we open PuTTY, we select SSH as connection type, and we insert the IP of our RPI which we found previously. We let everything else as it is and press open. Then a command prompt interface will show up asking for login and password for logging in to RPI. The default login username: pi, and password: raspberry, are standard for Raspbian lite OS.

- **Installing Node-RED (NRED) on RPI**

On the command prompt we type:

\* sudo apt-get update && sudo apt-get upgrade – (we update the RPI OS)

\* sudo apt-get install nodered – (we install Node-RED)

Now we make the NRED to run automatically when the RPI boots up:

\* sudo systemctl enable nodered.service

And then we restart the RPI:

\* sudo reboot

Now to access the RPI NRED interface, we use our PC web-browser by typing:

\* http://OUR\_RPi\_IP\_ADDRESS:1880

In our case it is:

\* 192.168.1.197:1880

And to access the NRED dashboard we type:

\*http:// OUR\_RPi\_IP\_ADDRESS:1880/ui

- **Installing MQTT Mosquitto Broker on RPI**

In the command prompt we have to update the repository by firstly importing the repository package signing key:

\* wget <http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key>

\* sudo apt-key add mosquitto-repo.gpg.key

Now we should make the repository available to be installed:

\* cd /etc/apt/sources.list.d/

\* /etc/apt/sources.list.d \$

Then we proceed in the following command:

\* sudo wget <http://repo.mosquitto.org/debian/mosquitto-jessie.list>

than we go back to the root directory:

\* /etc/apt/sources.list.d \$ cd

\* \$

now we update the information:

```
* sudo apt-get update
```

Now we install the Mosquitto broker:

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

## 5.2 Configuring esp-e12 NodeMCU

First we download the Arduino IDE (<https://www.arduino.cc/en/Main/Software>) for windows, than after installing the IDE, we install the esp-e12 board in Arduino IDE:

1. File > Preferences , and then to the Additional Board Manager URLs field we enter the link : [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) and press “ok”.
2. Tools > Board > Boards Manager... , than we select “esp8266 platform” and install it.
3. From Tools > Board > NodeMCU 1.0

- **Installing PubSubClient Library**

It provides a client for doing simple publish/subscribe messaging with servers that support MQTT, basically allows the esp-e12 to talk with Node-RED.

First we download this [19] link and after unzipping the folder we rename it to “pubsubclient” and then we move the folder to Arduino IDE installation libraries folder and then we restart the IDE.

- **Connecting esp-e12 to Node-RED**

Then we go to “File > Examples > PubSubClient > mqtt\_esp8266” where a new sketch code will appear that enables to publish/subscribe to a topic with MQTT.

Now we will explain the sketch how it works.

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

It loads the WI-FI and PubSubClient library.

```

const char* ssid = ".....";
const char* password = ".....";
const char* mqtt_server = "broker.mqtt-dashboard.com";

```

where we have to fill in our router Wi-Fi credentials like name of the Wi-Fi (SSID) password, and RPI IP address.

```

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

```

It initializes the espClient and creates three variables.

```

void setup() {
  pinMode(BUILTIN_LED, OUTPUT); // Initialize the BUILTIN_LED pin as an output
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

```

The setup function sets the esp-e12 built-in LED to output and starts the serial communication at baudrate of 115200. It also calls the setup Wi-Fi function, it connects the esp-e12 to MQTT broker and sets the callback function.

```

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

```

The setup Wi-Fi function connects the esp-e12 to the router WI-FI using our given credentials. If it connects successfully the esp-e12 IP address is shown to the IDE serial monitor.

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    ++value;
    snprintf (msg, 75, "hello world %ld", value);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("outTopic", msg);
  }
}
```

The loop function checks if the esp-e12 is connected to MQTT broker and if not, it will try to connect/reconnect. We have also a timer that every two seconds publishes the message “hello world” to a topic called outTopic. The outTopic, is just an example of the library which we will skip.

```
void reconnect() {

  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");

    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c_str())) {
      Serial.println("connected");

      client.publish("outTopic", "hello world");

      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      delay(5000);
    }
  }
}
```

If esp-e12 loses connection it will try to reconnect by calling the reconnect function. If still can't connect, it will try every 5 seconds to do so, and when it establishes connection, it publishes a message to the "outTopic" topic and subscribes to the "inTopic" topic.

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);

  } else {
    digitalWrite(BUILTIN_LED, HIGH);
  }
}
```

The callback function will be triggered any time another device (in our case the RPI) publishes a message to our esp-e12. In this case the message received will be printed out in the serial monitor of the IDE.

### 5.2.1 Controlling power outlets using esp-e12

To control the home electrical equipment, lights, etc. remotely, we must power the equipment using radiofrequency (RF) controlled outlets, and control the sockets through our esp-e12 using a radio-frequency transmitter.

Our RF sockets [20] come with controlled remote controller and they can be used in any country.

To start with by setting our remote controller to the position I, and the outlets must be all in the I position, thus are able to control our sockets using the remote controller.

- **Installing RC switch library**

The library will enable us to control radio controlled devices using esp-e12 and it will work with the most popular low cost RF outlet sockets. We can download it here [21] and we unzip the file, after it we rename the name to “rc\_switch”, then we move the folder to the Arduino IDE installation libraries folder.



Figure 18. Radio controlled power socket

- **Decoding the RF signals to control RF outlets**

In order for the esp-e12 to control the RF outlets, we need first to decode the RF signals that the remote control (of the RF outlets) sends so that the esp-e12 can reproduce those signals.

So firstly we open the IDE example by going to: File/RC\_Switch/ReceiveDemo\_Advanced.

A new sketch will show up, then we will use an Arduino Uno clone (Fig. 19) which we bought for less than 8 dollars, but we won't need this board for other tasks than this one and some other small task, therefore we can just borrow it for a short time if possible without having to pay for it.

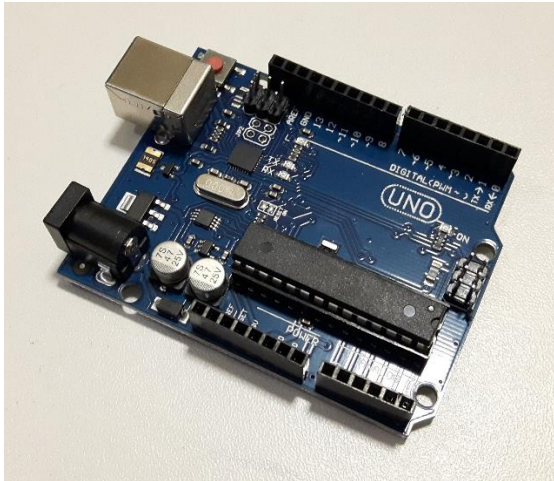


Figure 19. Arduino Uno clone

We connect our Arduino clone to our PC and to the RF receiver, then in the IDE we go to the “tools” tab, then we select “Arduino UNO” board, and then we select the “COM” port, at last we press “upload” button to upload the software to Arduino Uno.

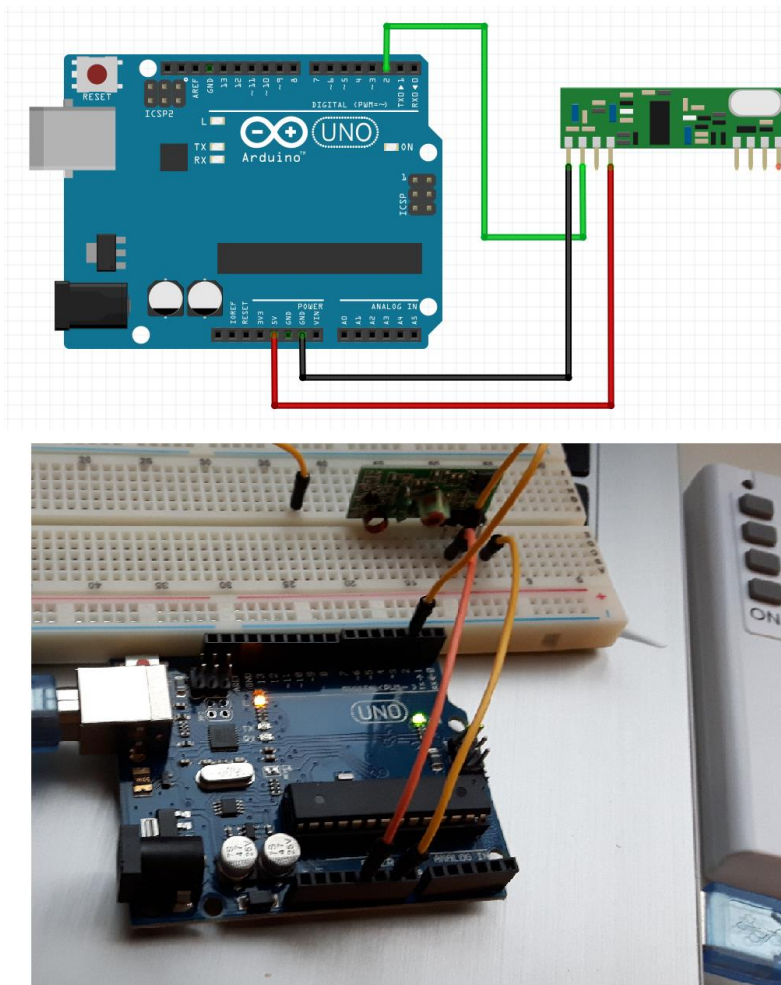


Figure 20. The Arduino Uno and RF receiver schematic



After building the circuit (Fig. 20), we open the IDE serial monitor and we start pressing the remote buttons. After pressing each button one time we get the binary code for each button (Fig. 20/1). Then we proceed by saving:

1. Binary code of outlet 1 for ON mode: 000101010001010101010101
2. Binary code of outlet 1 for OFF mode: 000101010001010101010100
3. Binary code of outlet 2 for ON mode: 000101010100010101010101
4. Binary code of outlet 2 for OFF mode: 000101010100010101010100
5. Pulse length: 435 microseconds
6. Protocol: 1

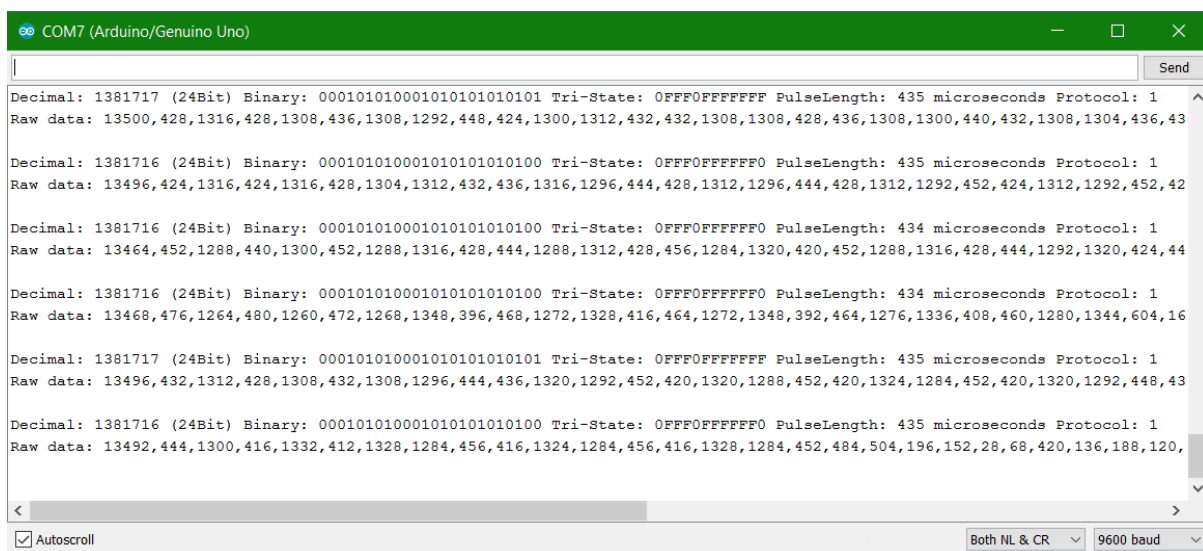


Figure 20/1. IDE serial monitor.

- **Preparing the esp-e12 program for controlling the outlets.**

We should replace our decoded RF signals settings to our code below.

We will do a brief overview of what is new to the code.

```
#include <RCSwitch.h>
```

loads the RC Switch library.

```
RCSwitch mySwitch = RCSwitch();
```

and this line creates the mySwitch variable.

```
mySwitch.enableTransmit(16);
```

the declaration of the D0 (GPIO 16) is set as transmitter inside the setup () function (Figure 20).

```
mySwitch.setPulseLength(435);
```

we set our pulse length,

```
mySwitch.setProtocol(1);
```

and here our protocol.

Now since our project will consider using many esp-e12 modules and outlets, in our case in order to identify which esp-e12 and outlet we are using, in the reconnect function we will subscribe to two topics which specify details about esp-e12 module and outlet.

```
client.subscribe("home/mainroom/espe12/light1");
```

```
client.subscribe("home/mainroom/espe12/light2");
```

Now through Node-Red dashboard we will send the binary codes to the callback function, and for this we need two “ if” statements for each outlet that sends the binary code through the topics to control the outlet ON/OFF state.

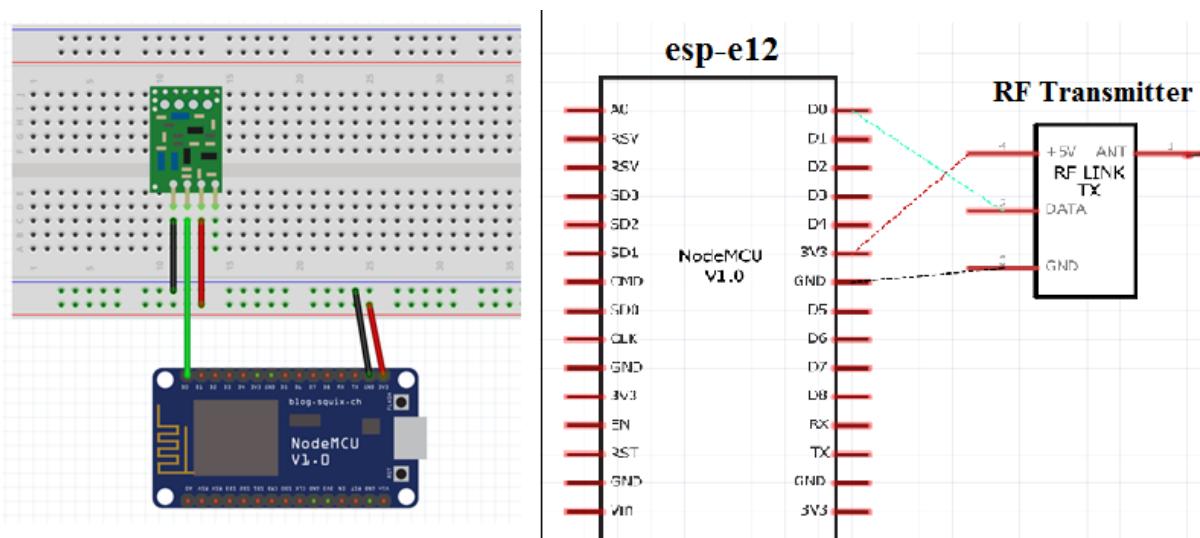


Figure 21. Schematics of the esp-e12 and RF transmitter.

```
if(topic=="home/mainroom/espe12/light1"){
    Serial.print("Mainroom light 1 changed to ");
    if(messageTemp == "1"){
        mySwitch.send("000101010001010101010101");
    }
}
```

```

        Serial.print("On");
    }

    else if(messageTemp == "0"){

        mySwitch.send("0001010100010101010100");

        Serial.print("Off");
    } }

if(topic=="home/mainroom/espe12/light2"){

    Serial.print("Mainroom light 2 changed to ");

    if(messageTemp == "1"){

        mySwitch.send("0001010101000101010101");

        Serial.print("On");
    }

    else if(messageTemp == "0"){

        mySwitch.send("0001010101000101010100");

        Serial.print("Off");
    }
}

```

And finally we upload the code to the esp-e12 in Arduino IDE.

- **Preparing the Node-Red program flow**

First we click to the settings tab of the NRED interface, view/Dashboard, and once the dashboard appears, in the layout tabs we create a new group and we name it “Main UI” group.

Then we drag two “switch” nodes and two “mqtt output” nodes into the working space of the NRED work space, and we start off by double-clicking the first switch node (fig. 22) and enter the group name which in our case is the “Main UI” group and will be for all the nodes of the esp-e12 NRED program. Then we enter the label name and the Name fields. At last, we select “string” option on both Payload’s” and enter 1 for ON and 0 for OFF payload.

Figure 22. Switch node

Figure 23. Switch node preview

We do the same for the second switch then on the mqtt node, on the server field we select “Add new mqtt-broker then we type “localhost” on the server field and leave all the rest as default. And last we come back to main window of mqtt node and enter the topic on the “Topic” field as shown in figure 23. The localhost server is the default server that NRED uses for standard tasks that we need to do. After we do the same with the second mqtt, we connect the switch nodes to the mqtt nodes as shown in figure 24. This connection causes the nodes or blocks to trigger each-other or pass the information in a message format (msg) to the connected block.

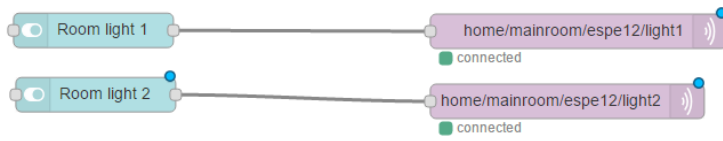


Figure 24. The NRED preview

At last we click deploy to the NRED web interface and when we access the NRED dashboard we should have the preview as shown in figure 25.



Figure 25. NRED web-browser dashboard

When we turn on and off the “light 1” and “light 2” button on NRED dashboard, we get the following messages on the IDE serial monitor as shown in figure26.

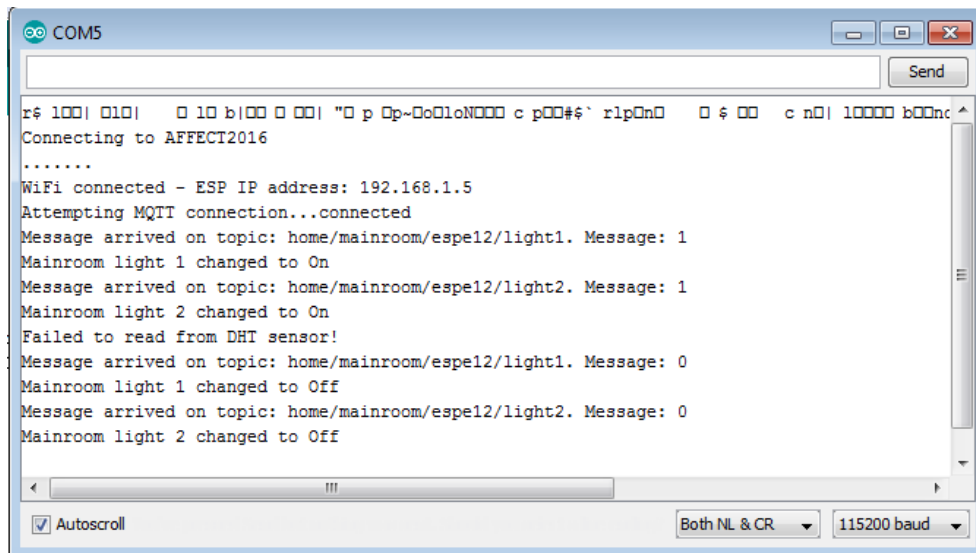


Figure 26. Receiving the binary code from NRED topic using MQTT connection

## 5.2.2 Controlling temperature sensor

- **Installing the sensor library**

We will use DHT11 sensor for temperature and humidity measurements, thus we should firstly install the sensor libraries [22] for our arduino IDE. After downloading the files, we put them in one folder and rename the folder as “DHT” and move it to the IDE installation libraries folder. We make sure we install adafruit DHT11 libraries.

- **Preparing the esp-e12 code for temperature sensor**

The code starts by loading the DHT library and the sensor model that we have:

```
#include "DHT.h"

#define DHTTYPE DHT11
```

Then we set the GPIO 5 pin of the esp-e12 (D1) as the “DHTGPIO” variable and we initialize the sensor:

```
const int DHTGPIO = 5;

DHT dht(DHTGPIO, DHTTYPE);
```

We need to keep track of the sensor values and for that we need a timer variable that will publish each period of time new values of the temperature and humidity.

```
long now = millis();

long lastMeasure = 0;
```

Then we begin the sensor in the “void setup()” function:

```
dht.begin();
```

Finally, in order to be able to publish every period of time a new value of temperature, we first need to compare the last measured value to the new measured one using a timer. For our sensor it is recommended to measure values using a minimum time of 30 seconds and above, because of the slow performance and to avoid mistakes of measurement.

```
now = millis();

if (now - lastMeasure > 30000)

{ lastMeasure = now;
```

and the following code is used to measure temperature in Celsius and check if the sensor did achieve the reading successfully, and if not, it exits early to retry reading again.

```
float h = dht.readHumidity();

float t = dht.readTemperature();

if (isnan(h) || isnan(t) || isnan(f))

{ Serial.println("Failed to read from DHT sensor!");

return; }
```

now we make the temperature conversion to Celsius,

```
float hic = dht.computeHeatIndex(t, h, false);

static char temperatureTemp[7];

dtostrf(hic, 6, 2, temperatureTemp);

static char humidityTemp[7];

dtostrf(h, 6, 2, humidityTemp);
```

and in the end, after receiving the temperature and humidity values, we need to publish them to the topics which then will be sent to NRED dashboard:

```
client.publish("home/mainroom/espe12/temperature", temperatureTemp);

client.publish("home/mainroom/espe12/humidity", humidityTemp);
```

- **Preparing the NRED flow**

We drag two input mqtt nodes and two gauge nodes. Then we fill the temperature topic to the first mqtt node (Fig. 27) and the humidity topic to the second one (Fig. 28).

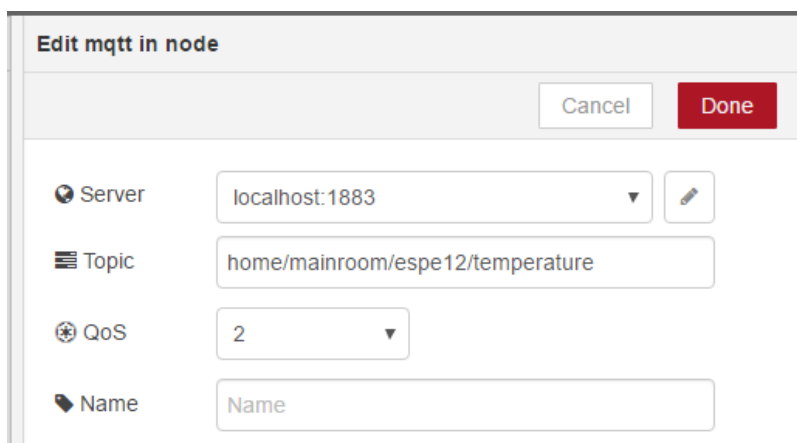


Figure 27. NRED mqtt node for temperature

The screenshot shows a dialog box titled "Edit mqtt in node". At the top right, there are "Cancel" and "Done" buttons. The main area contains four rows of configuration options:

- Server:** A dropdown menu showing "localhost:1883" and an edit icon.
- Topic:** A text input field containing "home/mainroom/espe12/humidity".
- QoS:** A dropdown menu showing the value "2".
- Name:** A text input field containing "Name".

Figure 28. NRED mqtt node for humidity

Then we configure the two gauge nodes by selecting the group for each of them and naming them (Fig. 29, 30).

The screenshot shows a dialog box titled "Edit gauge node". At the top right, there are "Cancel" and "Done" buttons. The main area contains several rows of configuration options:

- Group:** A dropdown menu showing "Sensor UI [Main UI]" and an edit icon.
- Size:** A text input field containing "auto".
- Type:** A dropdown menu showing "Gauge".
- Label:** A text input field containing "Temperature".
- Value format:** A text input field containing "{{value}}".
- Units:** A text input field containing "units".
- Range:** Two text input fields labeled "min" and "max" containing "-15" and "45" respectively.
- Colour gradient:** Three colored rectangular swatches: green, yellow, and red.
- Sectors:** A row of text input fields containing "-15", "optional", "optional", and "45".
- Name:** A text input field containing "Temperature".

Figure 29. NRED temperature gauge node



Figure 30. NRED humidity gauge node

At last, we connect the nodes with each-other (Fig. 31) and hit the deploy button.

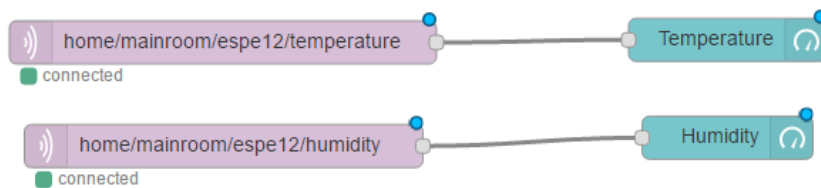


Figure 31. NRED node configuration

After deploying it, and after uploading the esp-e12 code, we navigate to the NRED dashboard, and we have the preview of our sensor feedback ( Fig. 32) and the serial monitor feedback of the sensor ( Fig. 33) using the below code on our esp-e12.

```
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(hic);
Serial.println(" *C ");
```

The code above is demonstrative only.

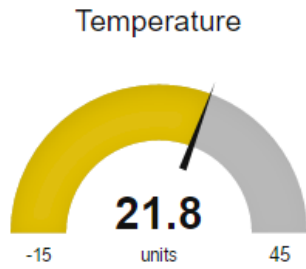


Figure 32. NRED dashboard temperature gauge

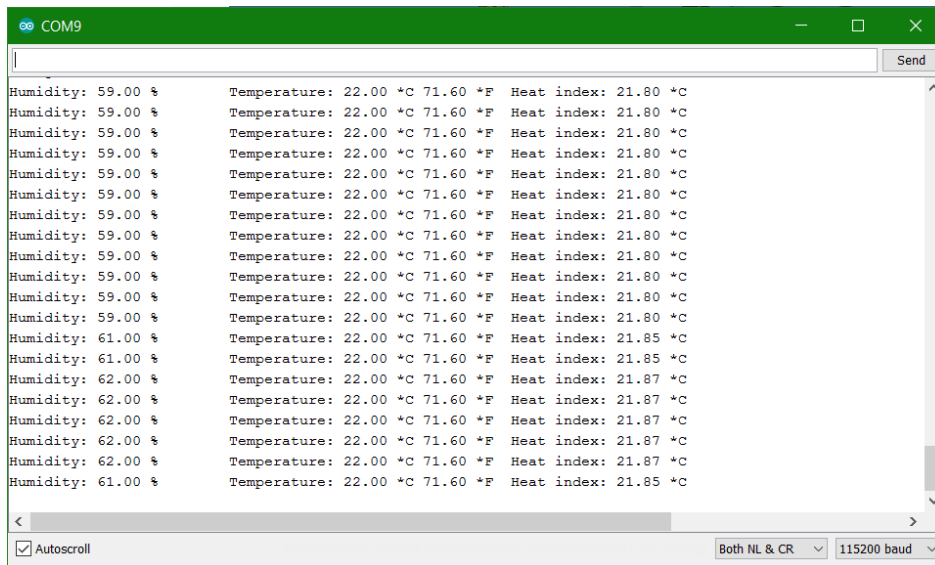


Figure 33. IDE Serial monitor

### 5.3 Controlling the Gas sensor and synchronization with the buzzer and LED light

The following sensor will serve as gas detector and will publish a notification to the NRED dashboard.

In this section we will make the sensor controllable through the NRED and we will install a LED light to indicate whether the sensor will on or not.

We will start by explaining the code of the esp-e12:

```

int smokegpio = A0;

const int buzzerGPIO = 14;
  
```

The line declares the analog pin A0 of the esp-e12 as the smoke pin and the GPIO 14 as the buzzer pin.

```
int smokeThres = 200;

boolean activateSmoke = false;

boolean smokeActivated = false;
```

We chose a smoke threshold which will serve as limit for the buzzer to be triggered.

Then we use the variable to activate or deactivate the sensor, and a variable to indicate the current status of the sensor in NRED. And in the beginning we declare it as deactivated.

```
long lastSmokeCheck = 0;
```

the variable declared is a timer which will be used to keeps checking the smoke value every few milliseconds.

```
pinMode(smokeLED, OUTPUT);

pinMode(buzzerGPIO, OUTPUT);

pinMode(smokegpio, INPUT);
```

inside the setup function we set the light and buzzer as outputs, and the sensor as input.

```
client.publish("home/mainroom/espe12/smoke/status", "Not Activated");

client.publish("home/mainroom/espe12/smoke/notification", "NO SMOKE");

client.subscribe("home/mainroom/espe12/smoke");
```

and here we publish two messages to the reconnect function so the esp-e12 will publish once it connects to mqtt broker. And we also subscribe to the topic which allows the NRED to turn on/off the sensor.

In order to actually activate or not the sensor using NRED, we will use the following code inside the callback function which provides us with the necessary information of the status of the sensor and to control it.

```
if(topic=="home/mainroom/espe12/smoke"){

    Serial.print("Change smoke sensor status");

    if(messageTemp == "1"){

        Serial.print("Smoke Sensor Activated");
```

```

    client.publish("home/mainroom/esp8266/smoke/status", "Activated");

    client.publish("home/mainroom/esp8266/smoke/notification", "NO SMOKE");

    activateSmoke = true;

    smokeActivated = false;

    digitalWrite(smokeLED, HIGH);
}

else if(messageTemp == "0"){

    Serial.print("Smoke Sensor Not Activated");

    client.publish("home/mainroom/esp8266/smoke/status", "Not Activated");

    client.publish("home/mainroom/esp8266/smoke/notification", "NO SMOKE");

    activateSmoke = false;

    smokeActivated = false;

    digitalWrite(smokeLED, LOW);
}

```

Basically the code makes possible to turn on/off the sensor and provide necessary feedback from esp-e12 of the status of the sensor.\

and lastly, the loop function makes possible for our smoke checking variable to repeat the smoke checking each 200 milliseconds and if there is an increase of the value above the smokethreshold variable, than a message which states “Smoke detected” is triggered to the NRED dashboard.

```

if (now - lastSmokeCheck > 200) {

    lastSmokeCheck = now;

int smokeValue = analogRead(smokegpio);

    if (smokeValue > smokeThres && activateSmoke){

        Serial.print("Pin A0: ");

        Serial.println(smokeValue);

        tone(buzzerGPIO, 1000, 200);

        if(!smokeActivated){

            Serial.println("SMOKE DETECTED!!!");

```

```

    smokeActivated = true;

    client.publish("home/mainroom/espe12/smoke/notification", "SMOKE DETECTED");
}
}

```

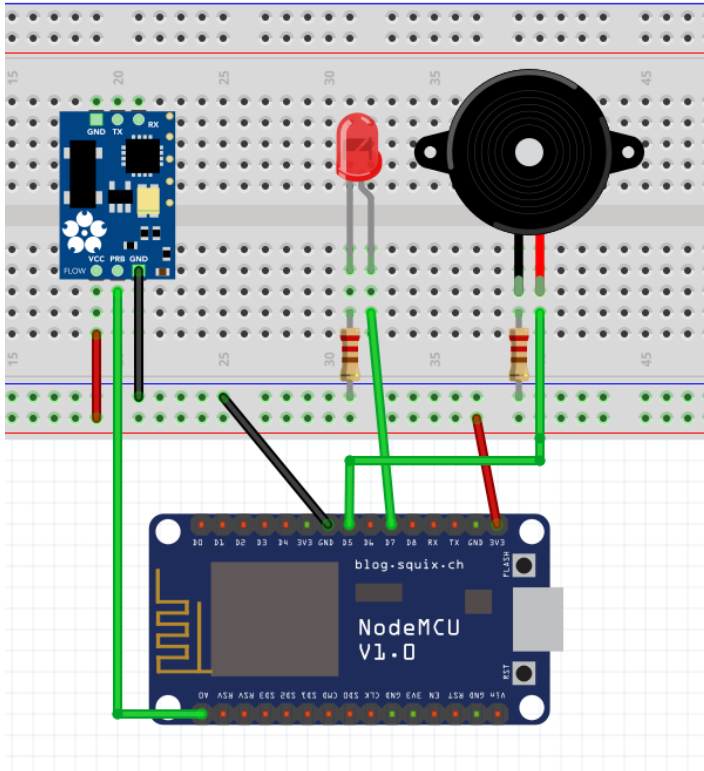


Figure 34. Connection schematics of the gas sensor with the esp-e12 module.

- **NRED configuration**

We insert two mqtt input nodes and one output one, than two text input nodes and two button nodes.

In the first mqtt input node we enter the topic that indicate the smoke status of the sensor:

```
*home/mainroom/espe12/smoke/status
```

then we connect the node to the text input node (Fig.35) which contains the status.

In the second mqtt input node we enter the notification topic:

```
* home/mainroom/espe12/smoke/notification
```

which then will be connected with the other text input node which will hold the status of the topic. Basically the notification topic will keep track of the sensor values, and notify the NRED if there is smoke or not.

The two button nodes, we will configure them as string buttons which will sent binary commands to the mqtt output node, to turn on the sensor. And the mqtt output node (Fig.36) will contain the topic related to the smoke sensor:

\* home/mainroom/espe12/smoke

Figure 35 shows the configuration for a text input node in NRED. The node is titled "Smoke sensor is:". It is configured to be a "text input" mode with a delay of 300ms. The node is set to pass through the message to the output when it arrives. The payload is set to "Current value". The topic is currently empty. The node name is "Smoke sensor is:". The node is grouped under "Sensor UI [Main UI]".

Figure 35. NRED text input node

After deploying the NRED, our configuration (Fig. 37) and the NRED dashboard (Fig. 38) are shown in below figures.

**Edit mqtt out node**

Cancel Done

Server

Topic

QoS  Retain

Name

Figure 36. NRED mqtt output node

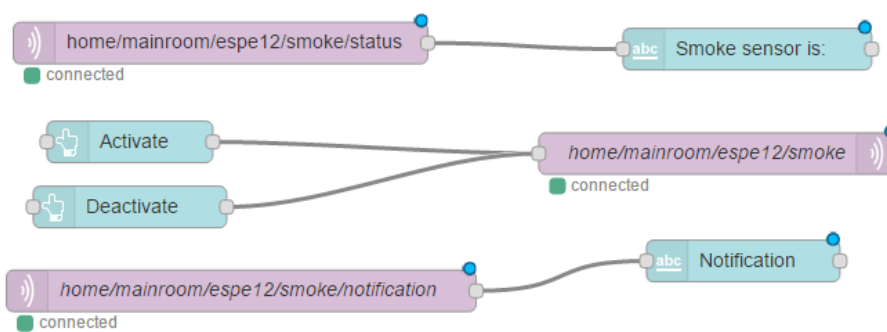


Figure 37. NRED node configuration

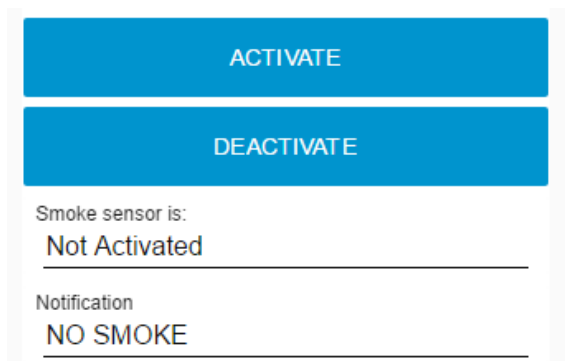


Figure 38. NRED gas sensor dashboard preview

Now after uploading the esp-e12 code we will have the following feedback from the serial monitor of the IDE and from the NRED dashboard (Fig.39).

We will press the activate button on the NRED dashboard for short time, and the sensor will start to detect smoke values, and then we hit deactivate button.

\*we should keep in mind that when we activate the sensor, the LED will blink accordingly, and the buzzer will trigger once we transcend the threshold value of the variable “smokeThres”.

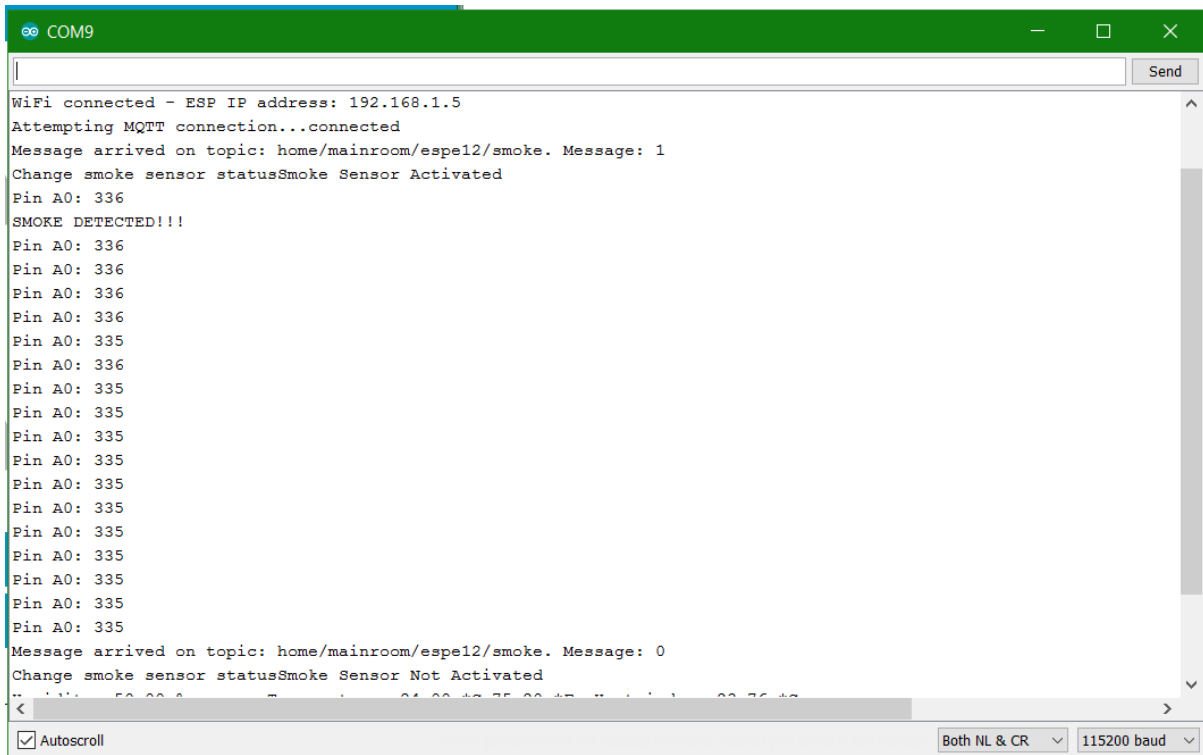


Figure 39. Serial monitor after pressing the activate button for short time and then the deactivate button.

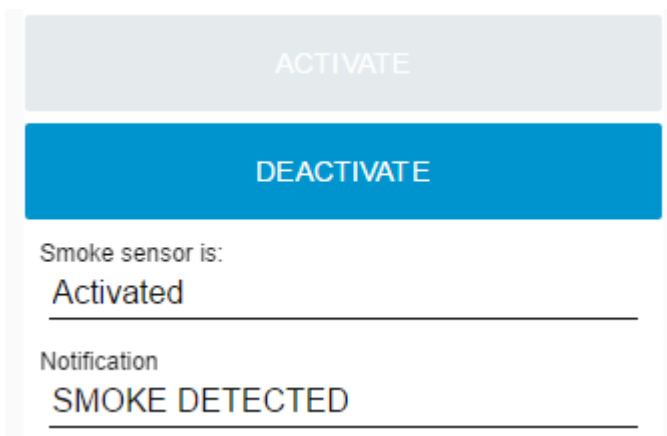


Figure 40. When activate button is pressed



## 5.4 Establishing motion detection system

We will install the PIR sensor together with a LED light to indicate the status of the motion sensor.

We will proceed by explaining the esp-e12 code:

```
const int motionSensor = 4;
```

here we assign the GPIO 4 of the board to the variable,

```
boolean activateMotion = false;
```

```
boolean motionActivated = false;
```

then we declare a variable for activating the sensor and the variable to notify the dashboard for the status of the sensor.

```
pinMode(motionLED, OUTPUT);
```

```
attachInterrupt(digitalPinToInterrupt(motionSensor), detectsMovement, RISING);
```

the motionLED is the light variable which is set as an output, and the sensor is set as an interrupt that activates in RISING mode.

```
client.publish("home/mainroom/espe12/motion/status", "Not Activated");
```

```
client.publish("home/mainroom/espe12/motion/notification", "NO MOTION");
```

```
client.subscribe("home/mainroom/espe12/motion");
```

once the esp-e12 is connected to the broker, the two topics inside reconnect function will be published in the broker, and then the esp-e12 will subscribe to the third topic mentioned above, in order to receive commands from NRED dashboard to activate the sensor.

We will use a function called “detectsMovement()” which will condition the sensor in cases that the activateMotion variable is active and motionActivated variable is triggered than there will be a “MOTION DETECTED” message published in the “home/mainroom/espe12/motion/notification” topic.

```
void detectsMovement() {  
    if (activateMotion && !motionActivated) {  
        Serial.println("MOTION DETECTED!!!");  
        motionActivated = true;  
    }  
}
```

```

    client.publish("home/mainroom/espe12/motion/notification", "MOTION DETECTED");
}
}

```

And through the callback function, we will be able to send commands using NRED dashboard to activate/deactivate the sensor thus arms/disarms the LED light also.

```

if(topic=="home/mainroom/espe12/motion"){

    Serial.print("Status change of motion sensor ");

    if(messageTemp == "1"){

        Serial.print("Motion Sensor Activated");

        client.publish("home/mainroom/espe12/motion/status", "Activated");

        client.publish("home/mainroom/espe12/motion/notification", "NO MOTION");

        activateMotion = true;

        motionActivated = false;

        digitalWrite(motionLED, HIGH);

    }

    else if(messageTemp == "0"){

        Serial.print("Motion Sensor Not Activated");

        client.publish("home/mainroom/espe12/motion/status", "Not Activated");

        client.publish("home/mainroom/espe12/motion/notification", "NO MOTION");

        activateMotion=false;

        motionActivated = false;

        digitalWrite(motionLED, LOW);

    }

}
}

```

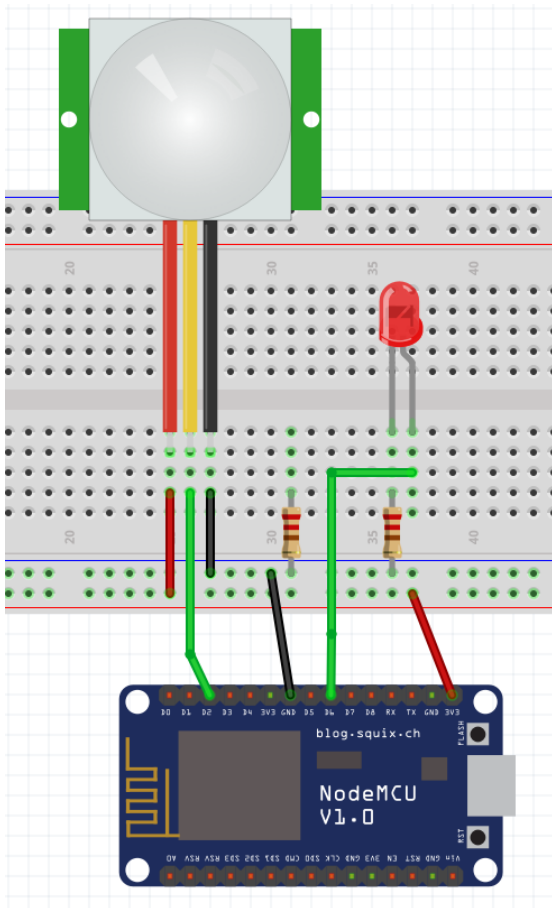


Figure 41. Connection schematics of PIR sensor

- **Configure the NRED flow**

We drag two button nodes for activating/deactivating the sensor, one output and two input mqtt nodes, and one input text node.

We fill the first input mqtt node by entering the following topic

```
*home/mainroom/espe12/motion/status
```

This node basically will subscribe to the mentioned topic, and will receive status messages from esp-e12. Then this node we will connect it to input text node, to show the messages into the NRED dashboard.

In the two button nodes we will configure them as string buttons and we will connect both with an output mqtt node, and the last will publish a topic to broker,

```
* home/mainroom/espe12/motion
```

which will make possible to activate/deactivate the sensor.



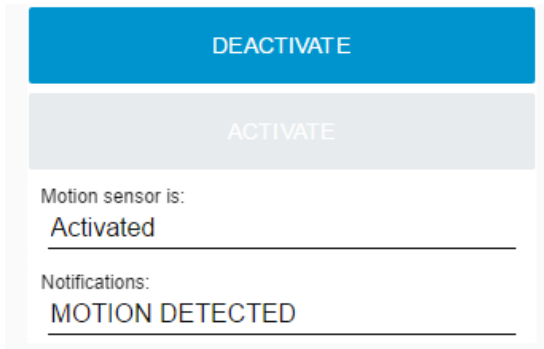


Figure 45. NRED dashboard PIR sensor activated

## 5.5 Smart home event triggering

### 5.5.1 Triggering email notification events

We can trigger an email notification event using any of the sensors that we are using like,

1. In cases of a specific reach of a temperature level
2. In cases of high levels of smoke detected from the gas sensor
3. In cases of high humidity
4. In cases of motion detection
5. In cases of high luminosity, etc.

We will use the motion sensor to trigger an email notification event when the sensor detects a movement.

In such case, we will receive an email from NRED, therefore we are aware of the situation of our home wherever we are in the world, as long as we have internet access.

We will edit the PIR NRED flow (Fig. 42). We first drag a calling function block that allows pure JavaScript code within our application, and an e-mail node.

Figure 46. Function node

Figure 47. Email node

In the function node we enter the following JavaScript code:

```

if(msg.payload == "MOTION DETECTED") {
    msg.topic="Motion detected in your home";
    msg.payload="Motion detected";
    return msg;
}
else {
    return null;
}

```

The function basically is triggering an email message based on the mqtt input node of the notification topic “home/mainroom/espe12/motion/notification”, in case the node payload receives a “MOTION DETECTED” feedback, than the payload of the function node will send a “Motion detected” message to the email node configuration.

In the email node (Fig. 47) we configure our email account which will be used to send the emails, and the email address that we will use to get the notifications.

We should make sure that we have to activate the “Allowing less secure apps” feature in the email account which will be used to send messages.

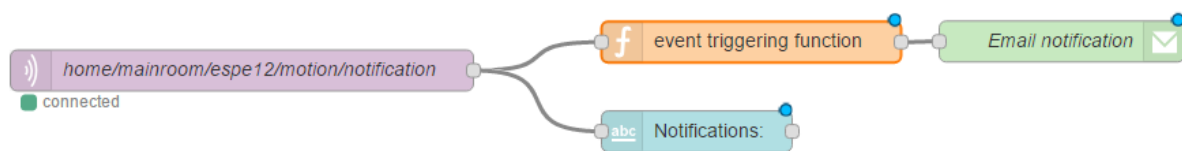


Figure 48. The NRED e-mail notification event flow

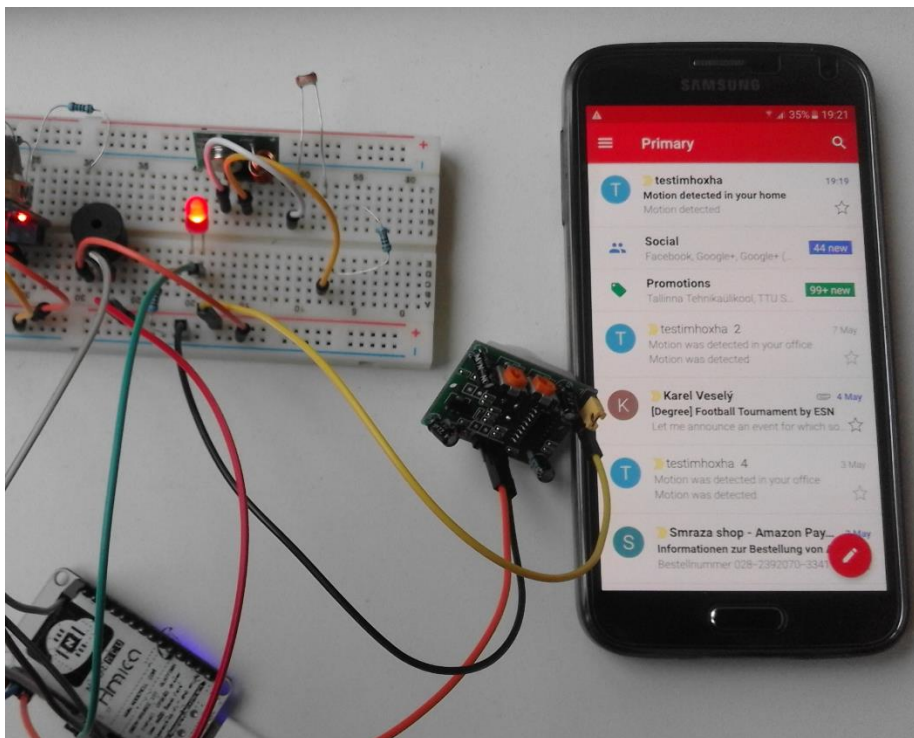


Figure 49. Email message received after motion is detected.

## 5.5.2 Controlling outlets based on the temperature sensor

We can use the temperature sensor as event trigger to control the home power outlets.

The NRED interface makes us possible to trigger an event based on the temperature sensor value. We can make it such, that if the room temperature goes over 24 degrees Celsius, than we turn off the power outlet that could power a room heater for instance, thus making it possible to control the room temperature.

For this we will use the existing NRED nodes of the temperature flow (Fig. 31) and power outlets (Fig. 24) and add a function node (Fig. 50).

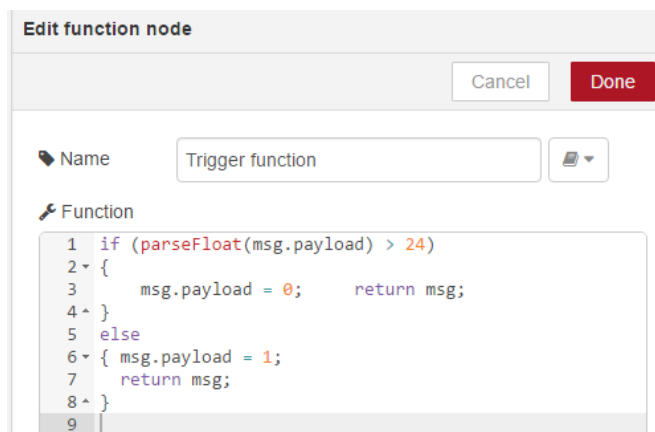


Figure 50. Trigger function node

The function we will write in the function node will make it possible to trigger a payload event which will send a 1 binary string to the “Room light 1” node which will turn off the power outlet.

```
if (parseFloat(msg.payload) > 24)
{msg.payload = 0;
return msg; } else
{ msg.payload = 1;
return msg; }
```



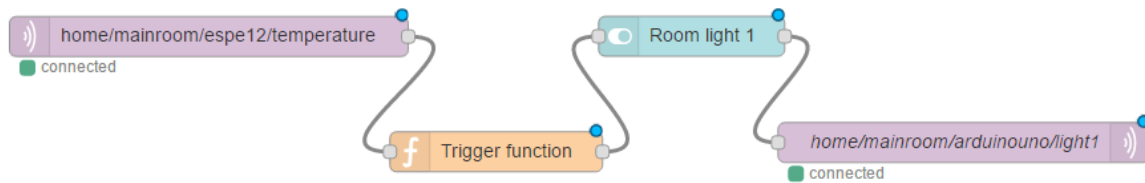


Figure 51. NRED flow of temperature trigger event.

### 5.5.3 Controlling outlets based on the luminosity sensor

The luminosity sensor will be connected to the Arduino Uno board since the esp-e12 contains only one ADC pin.

To measure the luminosity we will use the LDR sensor.

The NRED supports many devices and controllers and gives the opportunity to combine the devices with each-other to trigger different events.

In our case we are using the Arduino Uno board together with the internet shield (Fig. 52) to receive feedback from the luminosity sensor, and the NRED interface will allow us to combine the feedback of the Arduino board to control the power outlet using the esp-e12 thus combining two different devices with each-other.

- **Preparing the arduino UNO code**

The arduino code is very similar to the esp-e12 module code, since we are using the same IDE, but we will focus on the luminosity sensor code part.

```
int LDRAnalogGPIO = A1;
```

the above code defines the analog pin for the sensor,

```
long LDRCheck = 0;
```

this variable server as a timer to check the LDR value.

```
if ((now - LDRCheck) > 3000) {
```

```
    LDRCheck = now;
```

```
    int LDRValue = analogRead(LDRAnalogGPIO);
```

```
    Serial.print("LDR Value: ");
```

```
    Serial.println(LDRValue);
```

```

static char LDRTemp[7];

dtostrf(LDRValue, 5, 0, LDRTemp);

client.publish("home/mainroom/arduinouno/ldr", LDRTemp);
}

```

here in the loop function, we check each three seconds the value measured of the sensor, after reading the value it publishes it in the NRED dashboard.

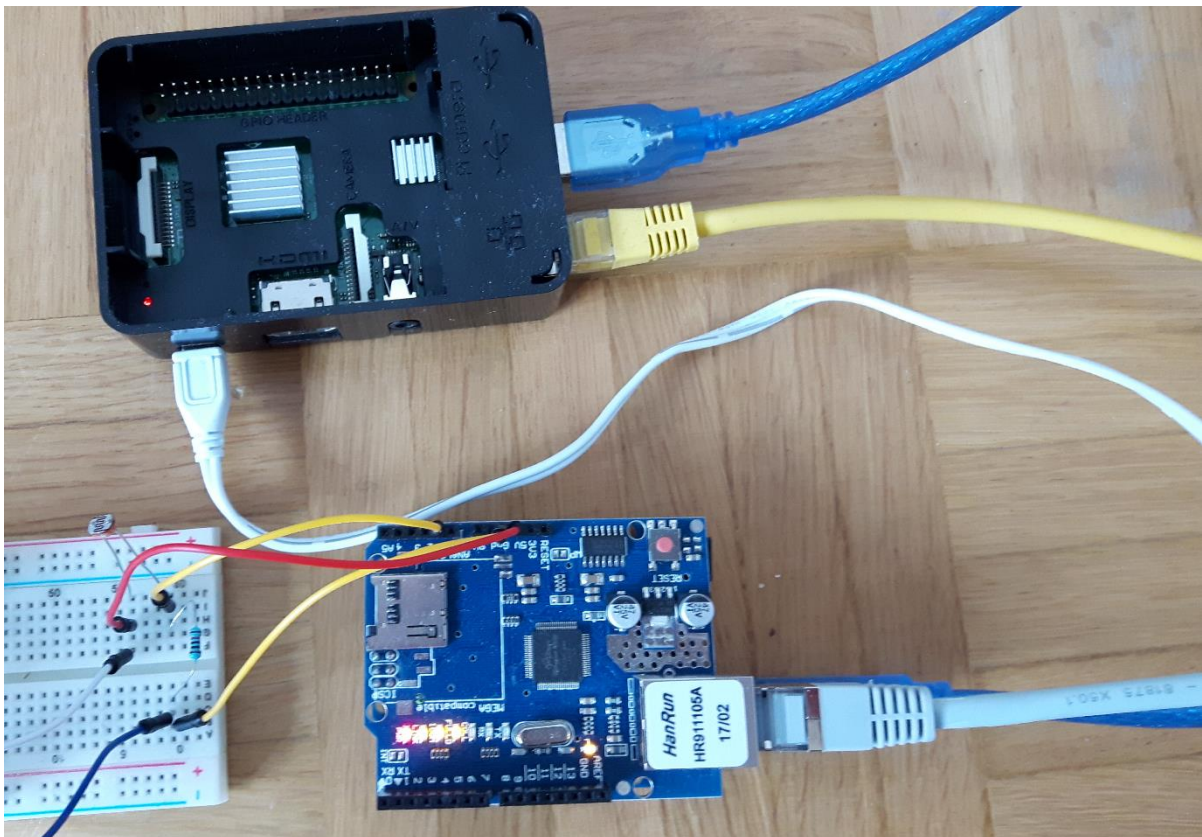


Figure. 52 Arduino UNO and Ethernet shield connected to the RF transmitter and LDR sensor

- **Building LDR event trigger on NRED**

For controlling the power outlet based on the luminosity sensor we will use the existing nodes of the power outlet 1 (Fig.22) and the mqtt output node (Fig.23). Then we add a function, mqtt input and a gauge node.

We enter the following code to the function node:

```
if (parseFloat(msg.payload)<400)
```

```
{msg.payload = 1; return msg; }  
  
else { msg.payload = 0;  
  
return msg; }
```

it triggers a 1 bit binary output towards the power outlet mqtt node to turn on the light when the luminosity intensity goes below 400 hundred units, which means that it has become almost night.

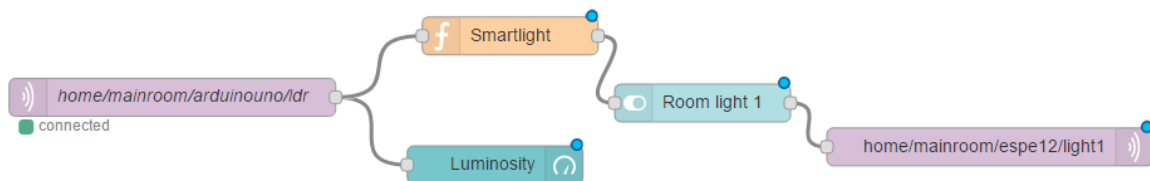


Figure 53. NRED flow

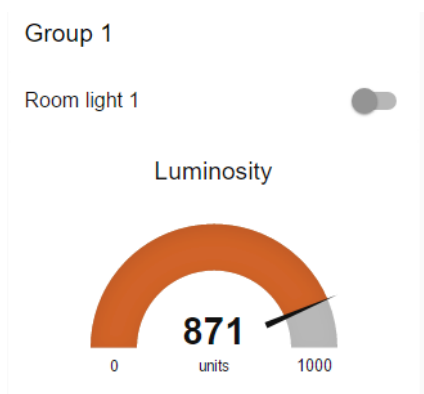


Figure 54. NRED dashboard interface

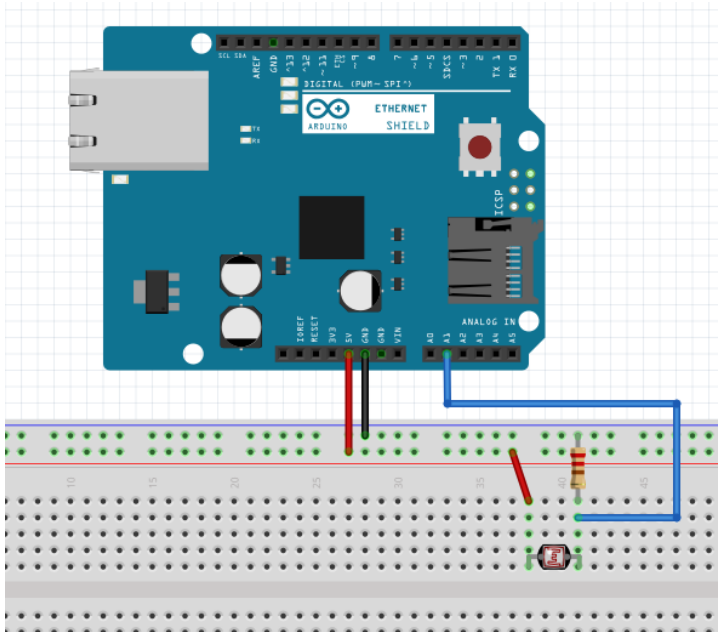


Figure 54/1. Connection of LDR sensor and Arduino Ethernet shield

### 5.5.4 Triggering time-based events

We can create time based events in order to control our hardware, be it power outlets, sensors, reminders or notifications etc.

In our example we are going to trigger time based events to turn on and off a power outlet at specific times using two inject nodes. The inject nodes will be connected to a switch node and it will activate the existing power outlet mqtt (Fig.23).

**Edit inject node**

---

Payload

Topic

Repeat

at

on  Monday  Tuesday  Wednesday  
 Thursday  Friday  Saturday  
 Sunday

Figure 55. Inject node to turn on light

**Edit inject node**

Cancel Done

Payload

Topic

Repeat

at

on  Monday  Tuesday  Wednesday  
 Thursday  Friday  Saturday  
 Sunday

Figure 56. Inject node to turn off the light

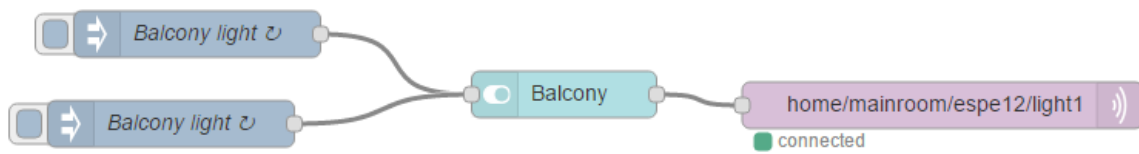


Figure 57. Time triggered balcony light

## 6 ACCESSING THE SYSTEM OUTSIDE THE HOME NETWORK

### 6.1 RPI Static IP

Usually the RPI IP address is a dynamic one which changes every time we disconnect it from the router or if the router restarts or disconnects from the internet, which makes our system not accessible.

We will configure our RPI with a static IP using SSH via PuTTY software.

We first type:

```
* sudo nano /etc/dhcpd.conf
```

And as we scrolled to the bottom we type the following commands:

```
* interface eth0
```

```
* static ip_address=192.168.1.197
```

```
* static routers=192.168.1.1
```

```
* static domain_name_servers=192.168.1.1
```

### 6.2 Port forwarding our RPI

In order to access our RPI NRED outside of our home network we should forward our RPI port.

Port forwarding is the most popular method to access home hardware outside of your home network. In order to have access to our RPI we need to use the public IP used by the router and the RPI port (1880) which we need to forward by logging into our router settings.

In the figure 60, we can see the settings which we need to set in order to forward our RPI.

This settings are almost the same depending on the router model, but those most common are:

RPI IP address: 192.168.1.197

Port: 1880

Protocol: All

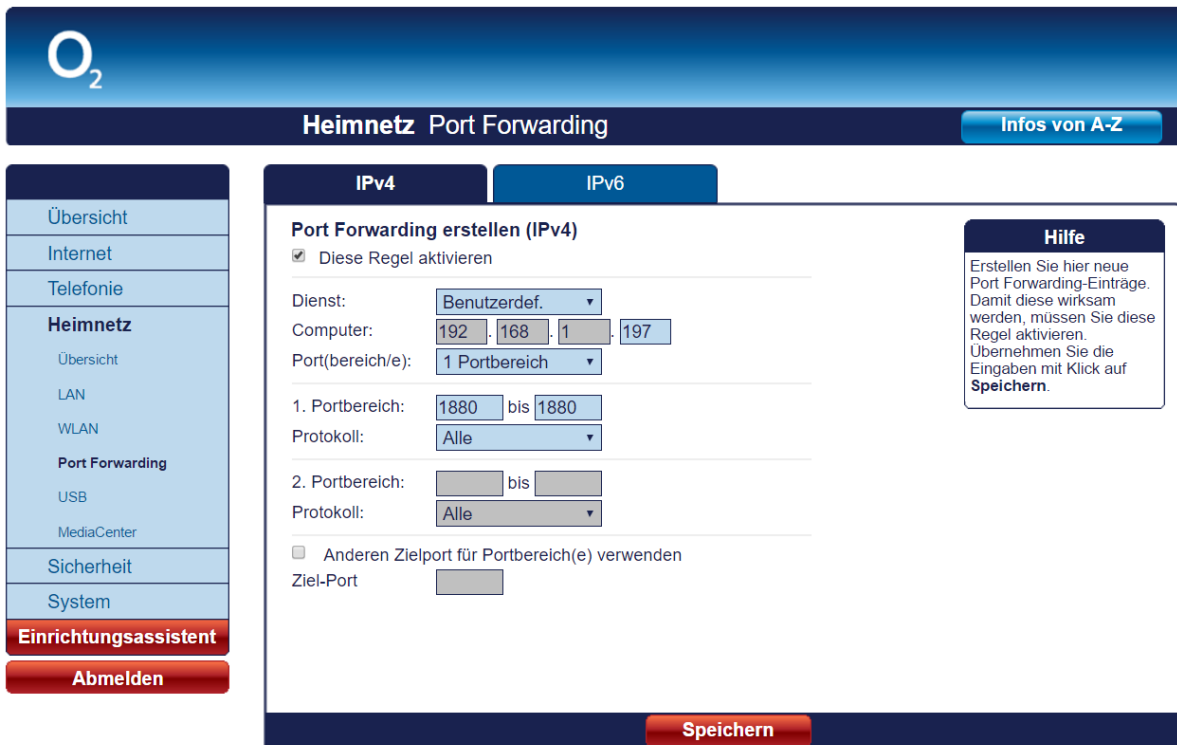


Figure 58. Router Port forwarding settings

To find our public IP address, we just type on google “what is my ip”.

Now we are able to access our RPI by typing our public IP followed by the port number in the web-browser.

We also need to have a static IP address for our router. Nowadays most of the Internet Service Providers (ISP) provide the clients with dynamic public IP’s which means that if a router is restarted or turned off, it will receive a new IP address.

Therefore we need also to bypass the dynamic IP address issue by using a service called Dynamic DNS.

There are many ways to set-up a free dynamic DNS for our RPI.

## 7 STORING AND ANALYSING THE SYSTEM DATA

After building the system we now need to store the data that the system is able to receive, in order to make it possible for the user, to check the activity of the system during a day, a week or even a month. We could be able to see the temperature values during a day in a graphic view, chart or list, and other sensors as well.

We will demonstrate three ways how to store temperature sensor data.

### 7.1 Storing data using Node-Red

NRED web-browser interface provides us with some tools that allows us to plot sensor data during a day, a week and even a month. The easiest way to see this data is by using a chart node.

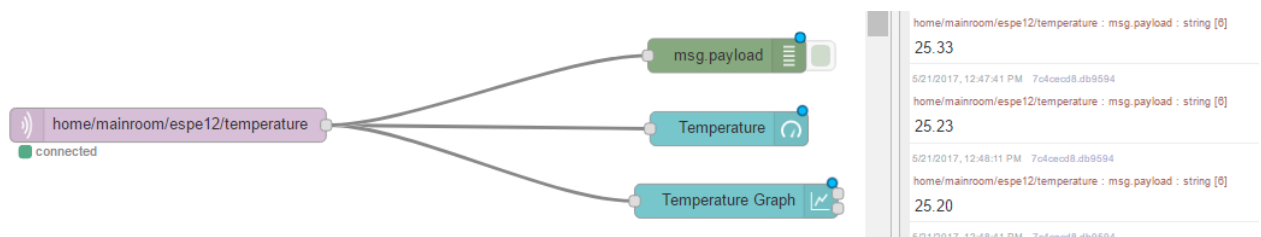


Figure 59. NRED configuration of the temperature chart node

As it can be seen in the figure 59, we connect a chart node with the temperature data source node and from the figure we have also connected a debug node in order to see the data feedback from the temperature mqtt node as it is shown in the right of the picture.

The chart node can be configured to plot data for a day and a week (Fig. 60), and we can easily check the temperature on every time of the day (Fig. 61).



**Edit chart node**

Cancel Done

Group: DHT11 [Day activity]

Size: auto

Label: Temperature Graphic

Type: Line chart

X-axis: last 1 days OR 1000 points

X-axis Label: HH:mm:ss

Y-axis: min 0 50

Legend: None Interpolate linear

Series Colours: [Color palette]

Blank label: display this text before valid data arrives

Name: Temperature Graph

Figure 60. Chart edit interface

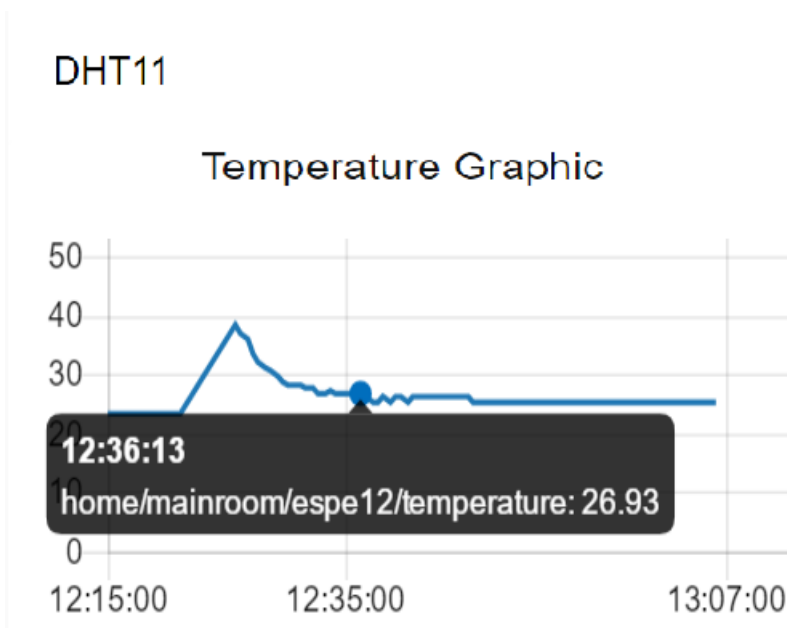


Figure 61. Temperature data shown during a specific time of the day

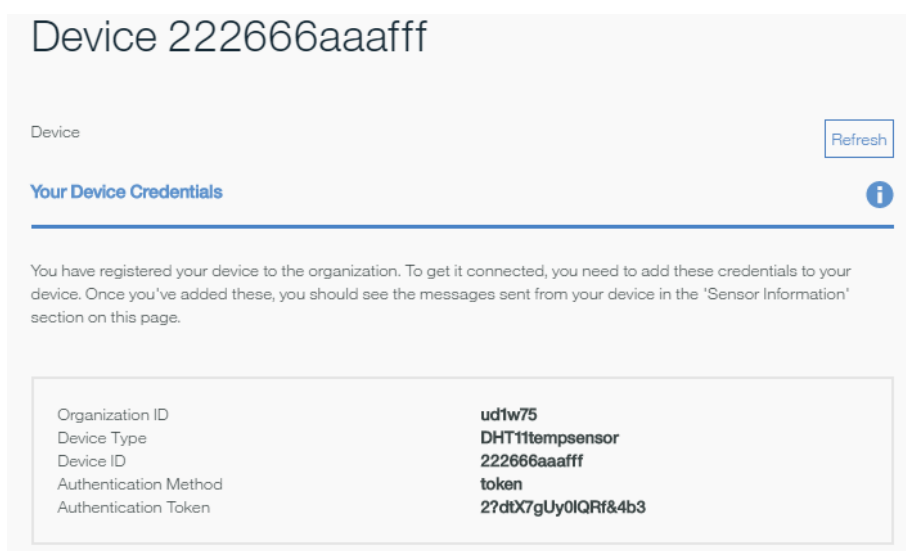
## 7.2 Storing data using cloud

Another useful method to store the data of the system is using open source clouds, and NRED can be very easily interfaced with the IBM Bluemix cloud. The Bluemix cloud platform offers a wide number of services but we are going to focus on the data preview and storage using Bluemix Watson internet of things platform.

Another very useful reason to use the IBM Bluemix cloud is that it enables us to access our data of the system outside the home network.

We start by creating a IBM Bluemix account, and after logging in we create a Watson IoT platform. After doing so we enter to the Watson IoT platform interface and create a new device on the device tab. We click add device and create a device type, there we enter just the name and click next, we enter an desired ID and click add device. Then it will show us the device credentials (Fig.62) which we should save for we need them later on.

Now we go to our NRED interface and add an output node called Watson IoT, then we connect it to the temperature mqtt node (Fig. 27) and later we double-click the node and enter the credentials (Fig. 63) of the device we created in Watson IoT platform.



The screenshot displays the 'Device 222666aaaff' page in the IBM Watson IoT platform. At the top, the device name is shown. Below it, there is a 'Device' label and a 'Refresh' button. A section titled 'Your Device Credentials' is highlighted with a blue header and an information icon. A paragraph explains that the device is registered and that credentials need to be added to connect it. Below this, a table lists the credentials:

Organization ID	ud1w75
Device Type	DHT11tempensor
Device ID	222666aaaff
Authentication Method	token
Authentication Token	2?dX7gUy0lQRf&4b3

Figure 62. Device credentials

The image contains two screenshots of the Watson IoT configuration interface. The top screenshot is titled "Edit Watson IoT node" and features a "Cancel" button and a red "Done" button. It includes a "Connect as" dropdown menu set to "Device", radio buttons for "Quickstart" and "Registered" (with "Registered" selected), a "Credentials" dropdown menu set to "Add new wiotp-credentials...", a text input for "Event type" containing "event", a "Format" dropdown menu set to "json", and a "Name" input field with a lock icon. The bottom screenshot is titled "Watson IoT > Add new wiotp-credentials config node" and features a "Cancel" button and a red "Add" button. It includes text input fields for "Organization" (ud1w75), "Device Type" (DHT11tempensor), "Device ID" (222666aaaff), "Auth Token" (masked with dots), and "Name" (Name).

Figure 63. Configuring the Watson IoT node

Then we deploy the application and in the Watson platform, our device status should show as connected (Fig. 64).

Now we go to boards tab and create a new board. We name it and press submit. Then we enter to the board and add a new card (Fig.65). Here we can see many kinds of ways how we can show our temperature data, from charts to gauges, bars etc. We can also add analytical cards where we can analyze the data we are receiving to the device.

We are going to create a gauge, a chart (Fig. 66) and a list card where we can see the temperature values. The chart can be configured to show the data for 5 minutes, till 24 hours (Fig. 67).

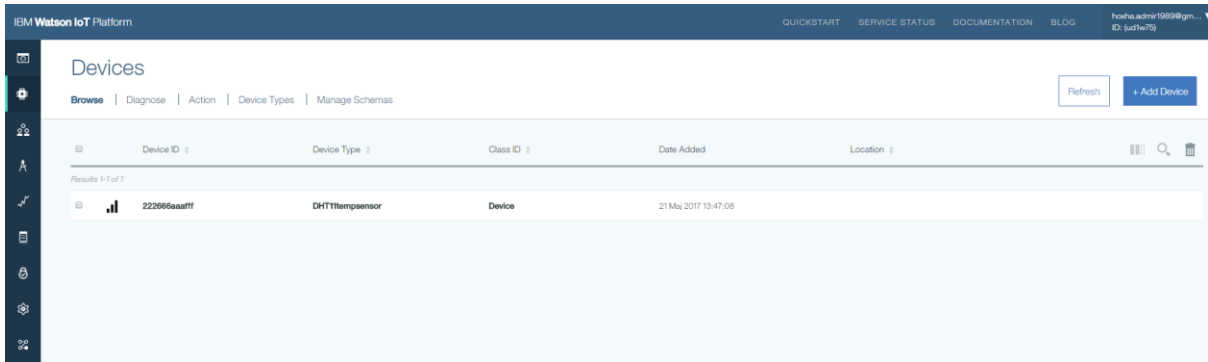


Figure 64. Preview of the device in the Watson platform

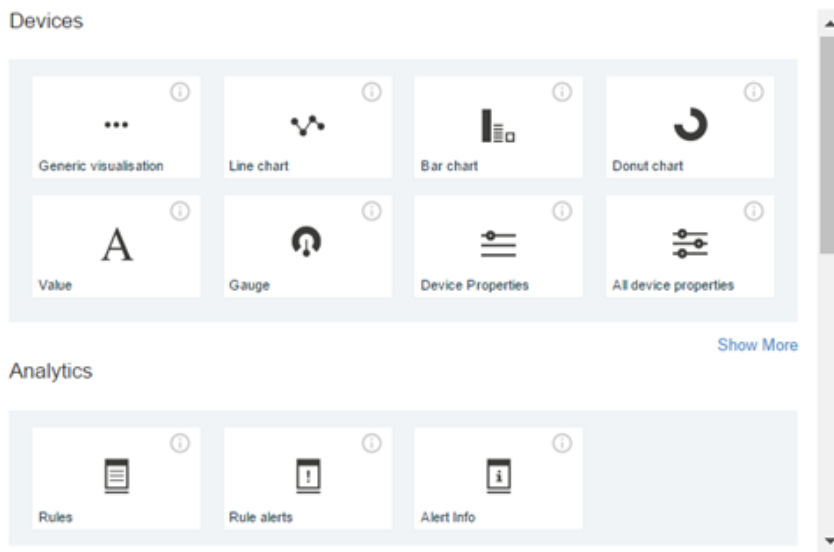


Figure 65. Adding a card for our device

We can also store unlimited data using the Historical data Storage feature in the extensions section of the Watson platform (Fig. 68). We can see unlimited data stored for our smart home system. The storage is done using CloudantNoSQL database.

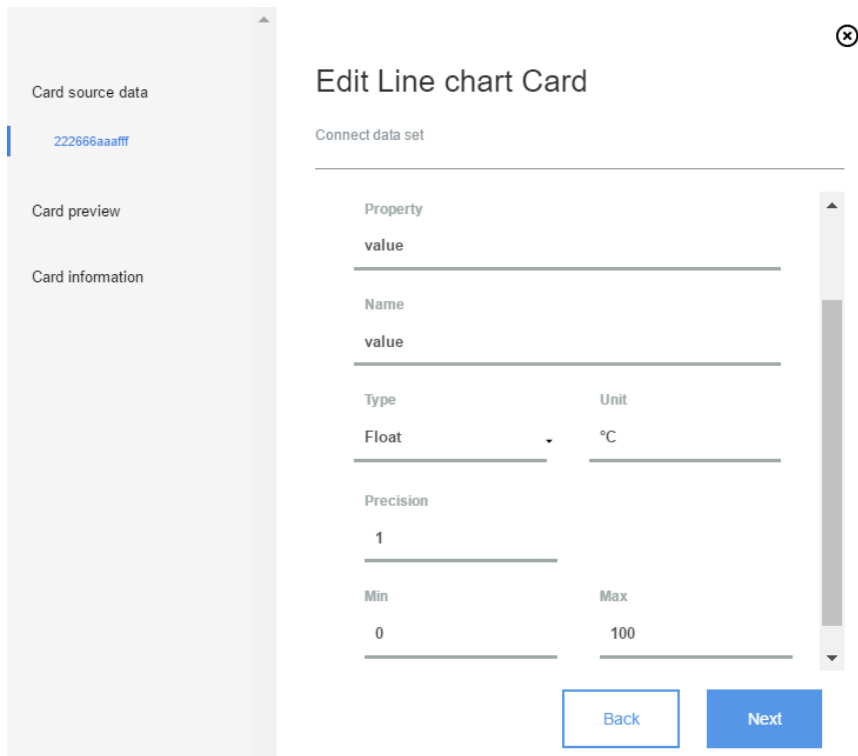


Figure 66. Creating a chart card for the device

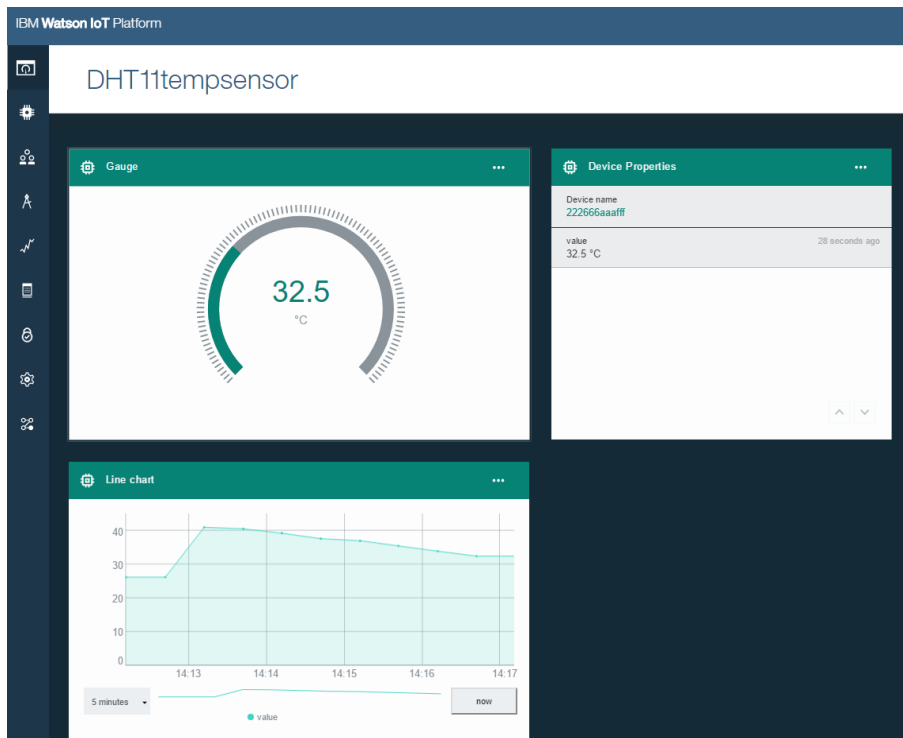


Figure 67. Watson platform sensor tables

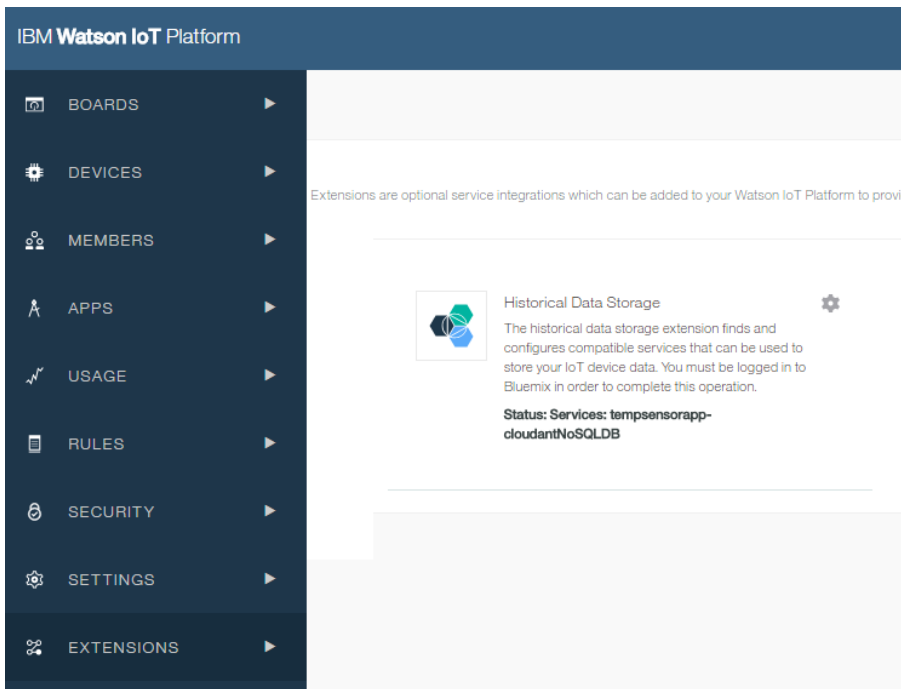


Figure 68. Configuring the historical data storage

In order to quickly login to our Watson platform dashboard we can simply login to IBM Bluemix account, and from there we can see our device and our data link (Fig.69).

If we want to see the device dashboard, we simply click to the IoT platform, and if we want to check the historical data stored we simply click to cloudandNoSQLDB database.

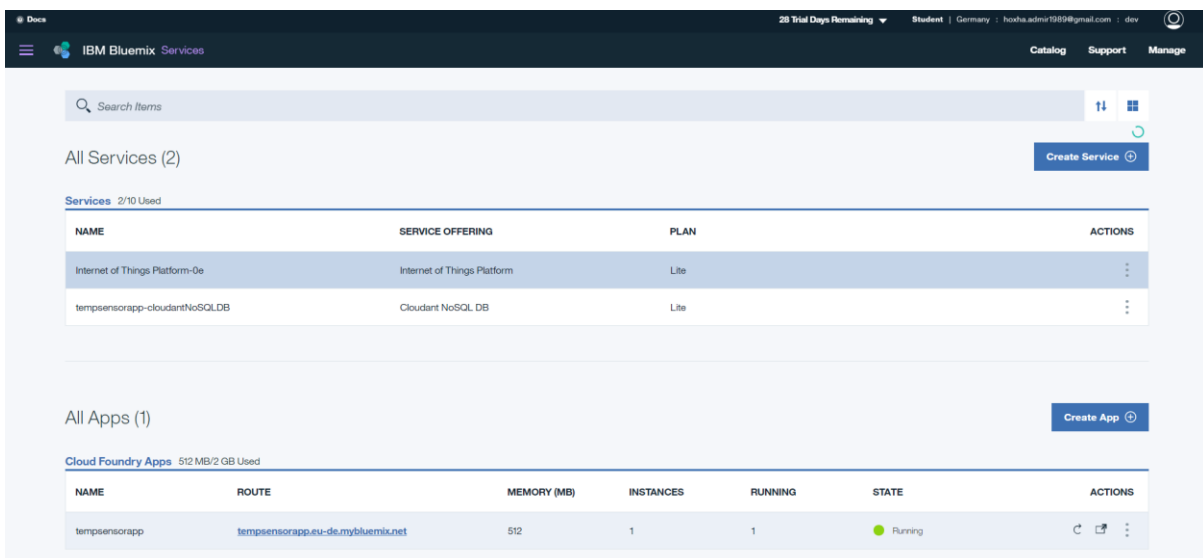


Figure 69. IBM Bluemix login page

In the cloudandNoSQLDB database we can see the data of the temperature sensor are organized according to day activity, and after we click on the first file (Fig.70) all the data is listed in the file (Fig. 71).

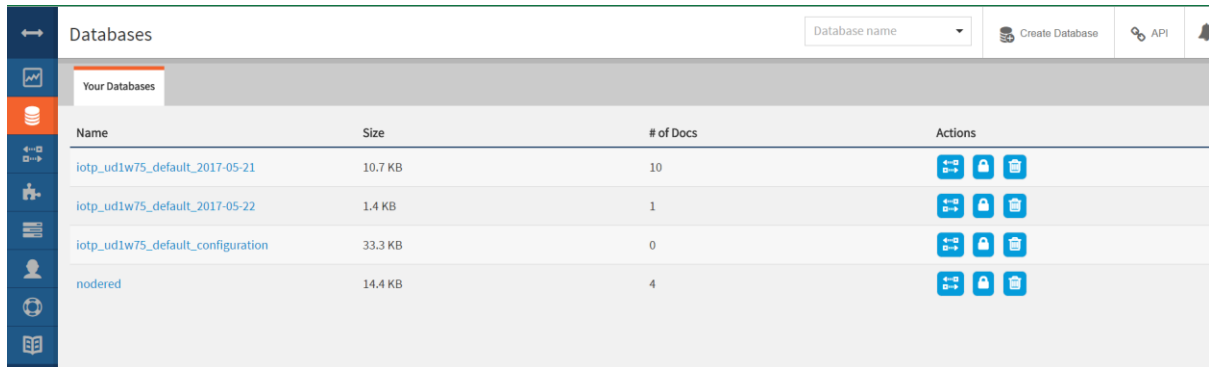


Figure 70. cloudandNoSQLDB database

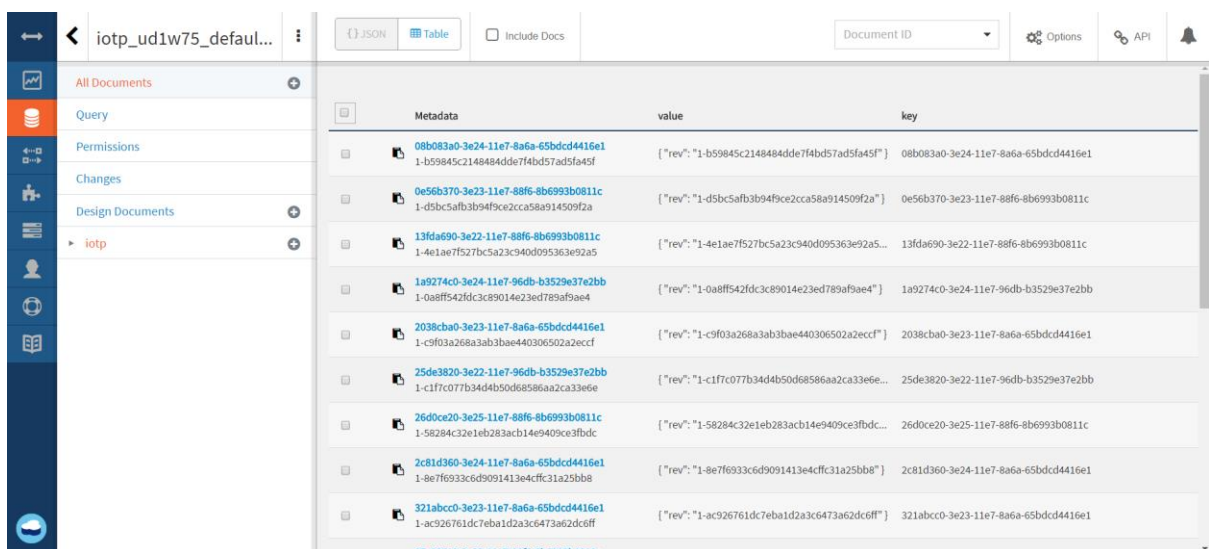


Figure 71. Temperature data stored in cloudandNoSQLDB database

### 7.3 Storing data using Matlab

We will display temperature and humidity data received from the temperature sensor to Matlab. Matlab has already available packages that can communicate remotely with a RPI and it can use it for its peripheral devices to be controlled. It basically allows to receive sensor data that are connected to the RPI.

For purposes of short time, we are going to demonstrate a data plotting of a DHT11 temperature sensor using the arduino uno board to send data to the matlab. The board will be connected to DHT11 sensor and connected to the PC. We will upload the code [35] used for the arduino (Appendix 1) and then we will run the matlab code (Appendix 2).

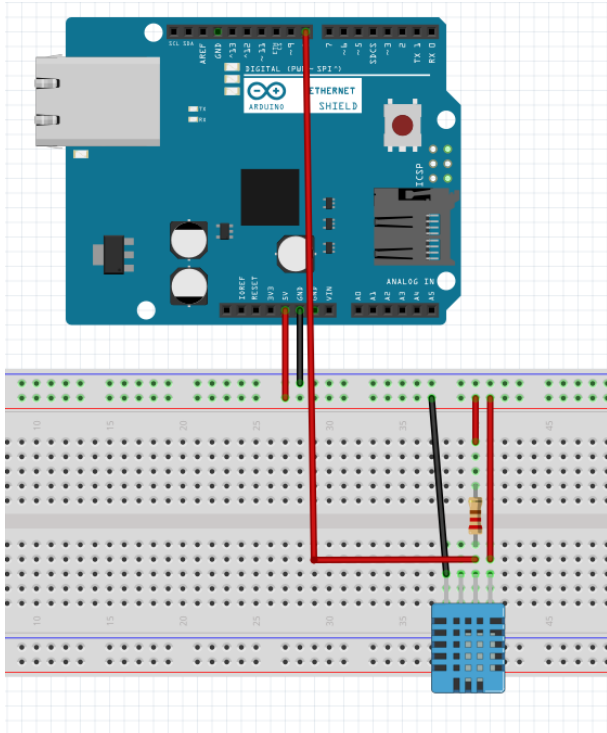
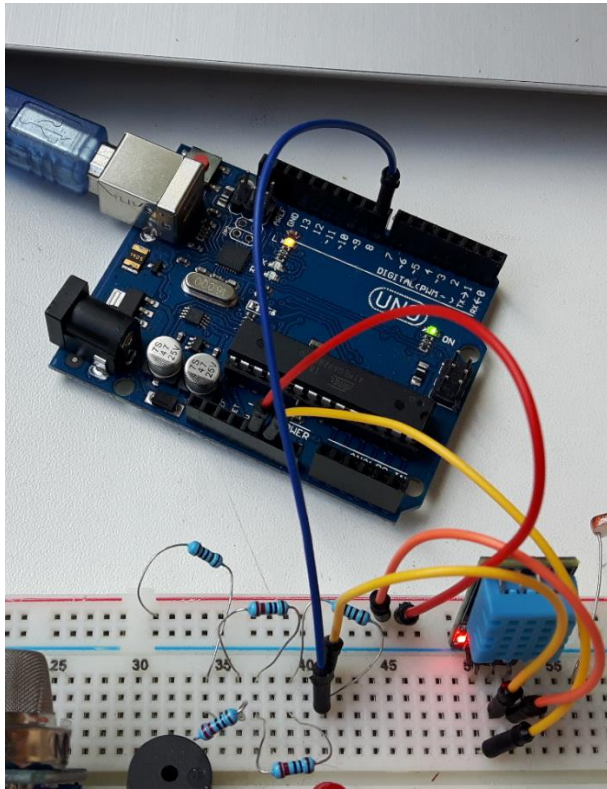


Figure 72. Connection configuration of DHT11 and arduino board



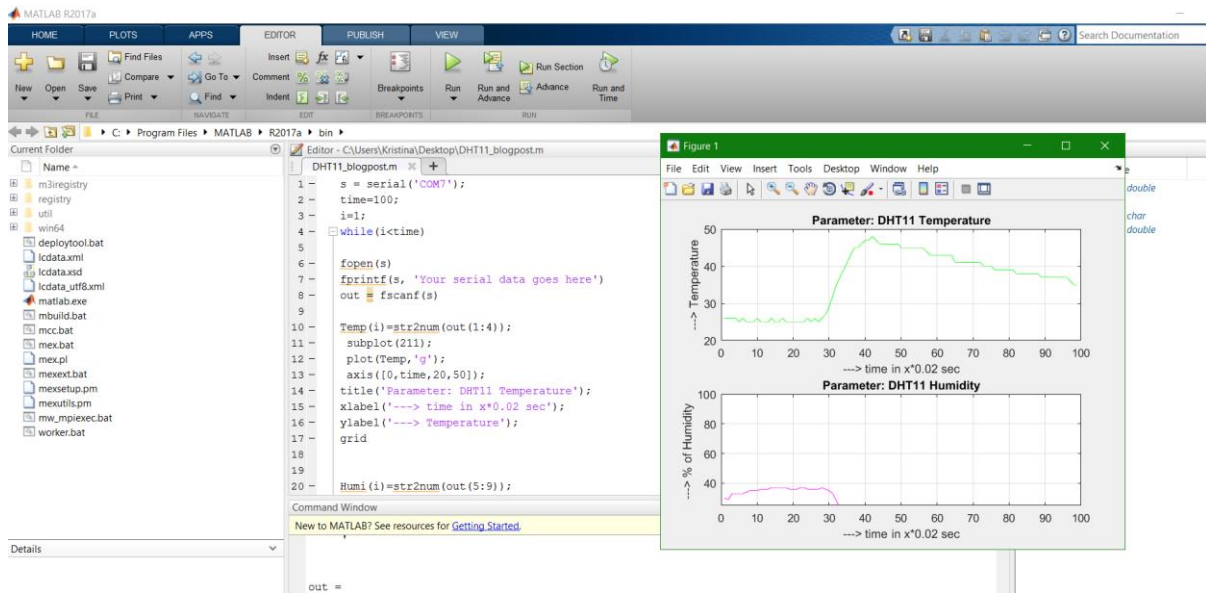


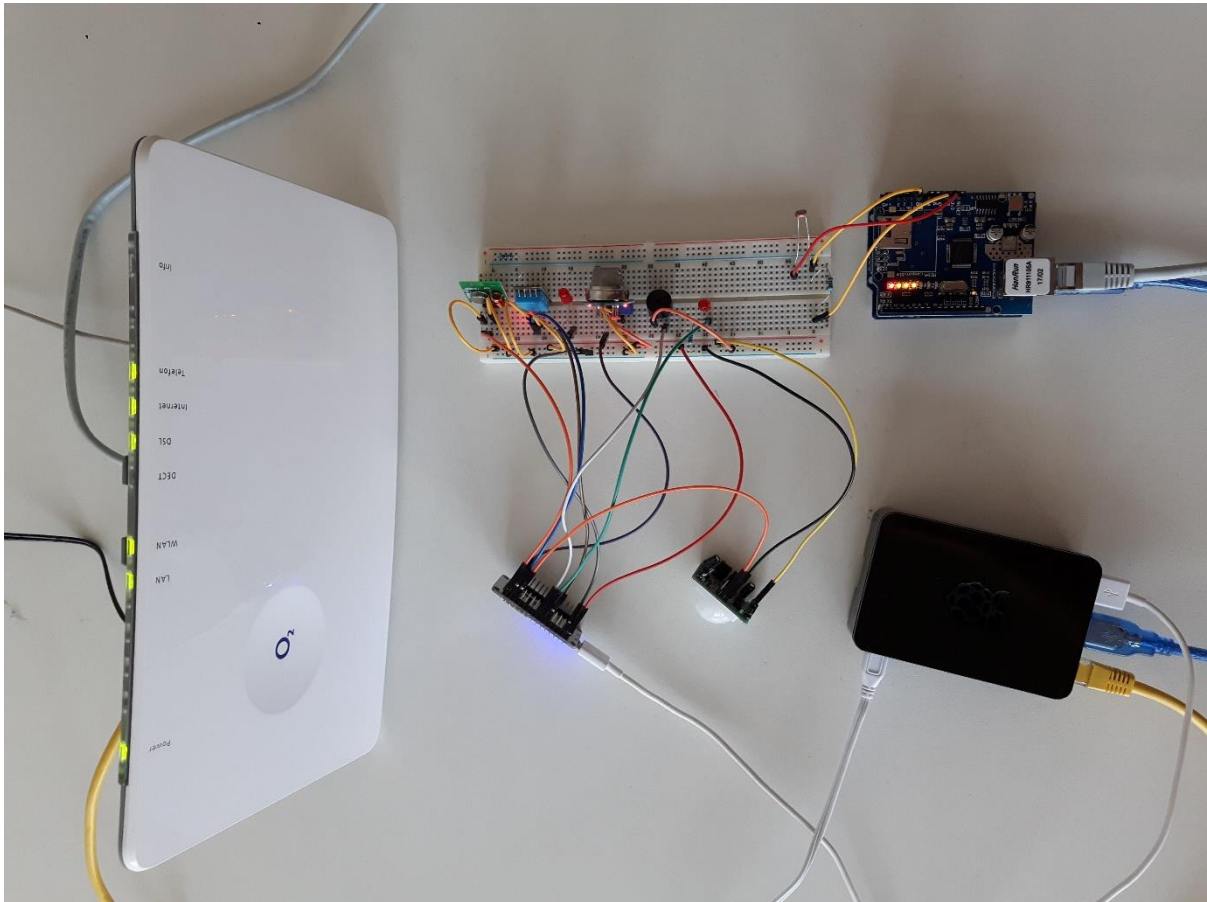
Figure 73. Matlab temperature and humidity graph

## 8 FINAL PROTOTYPE

We were able to build the system with all the mentioned hardware using only a cheap Wi-Fi module like the esp-e12 NodeMCU which costs less than six dollars.

We were also able to receive feedback from all the sensors, trigger email notification events, time based events and also sensor based events.

The NRED web-browser interface proved that it is very flexible for home automation tasks and it can be adopted with many hardware's.



*Figure 74. Home automation system configuration circuit*



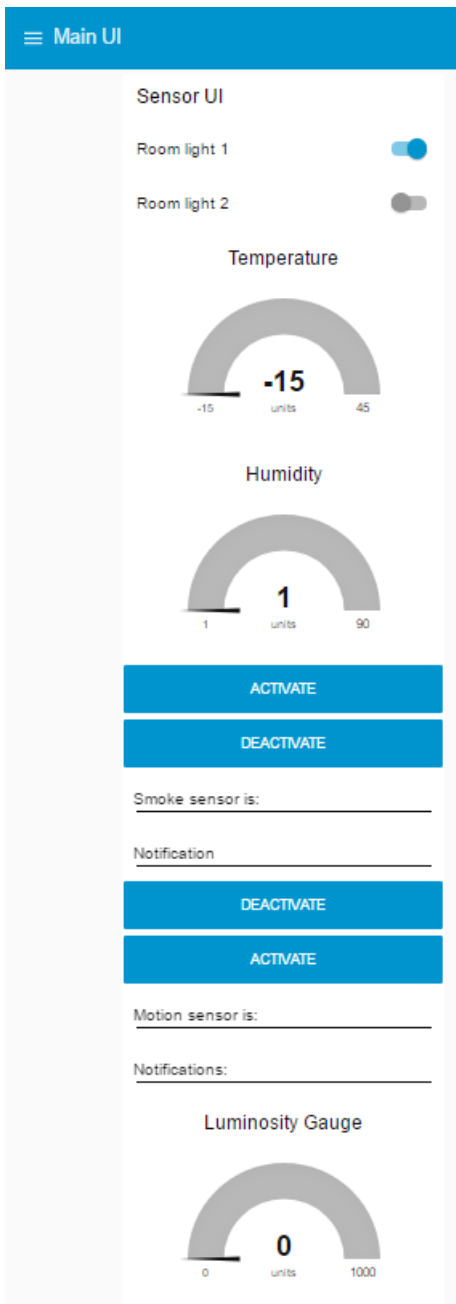


Figure 76. Main user interface of the NRED dashboard “Main UI”

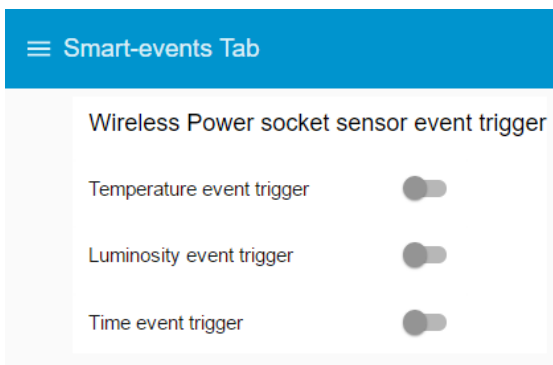


Figure 77. The interface tab of smart events that we created

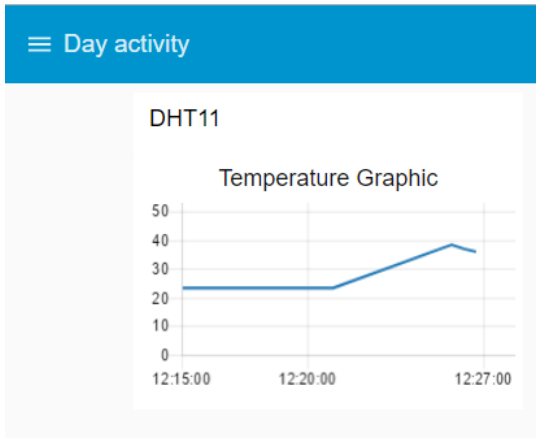


Figure 78. Temperature graph of day activity

## 9 KOKKUVÕTE

Juhtmevaba koduautomaatika süsteemid on tänapäeval muutumas üha odavamaks ja sellest tulenevalt ka populaarsemaks. Nutitelefonide, tahvelarvutite ja juhtmevabade tehnoloogiate laiem levik on muutnud juhtmevaba koduautomaatika (JKA) töökindlamaks ja kergemini realiseeritavaks. Selles lõputöös käsitletakse JKA süsteemi, mis kasutab avatud lähtekoodiga riistvara ja tarkvara, muutes saadud seadme taskukohasemaks. Antud seade kasutab erinevaid sensoreid, et koguda infot kodu keskkonnatingimuste kohta. Juhtmevaba andmeedastus toimub kasutades avatud lähtekoodiga ja turvalist MQTT protokollit. Andmed edastatakse RPI kohaliku serveri ja Node-Red abil üle interneti. NRED töötab RPI peal, millega on võimalik suhelda kasutades mistahes veebibrauseriga seadet. Niiviisi on võimalik jälgida või juhtida JKA süsteemi. Antud JKA süsteemi on võimalik kasutada turvasüsteemina, kodumasinat kaugjuhtimiseks või targa kodu süsteemina, mis on võimeline reageerima teatud keskkonnatingimustele ilma inimese sekkumiseta. Käesolev süsteem on paindlik, seda on võimalik laiendada lisasensoritega. Esp-e12 Wi-Fi moodul võimaldab süsteemi juhtmevaba ühenduse sensoritega, lihtsustades antud JKA süsteemi ja muutes seda praktilisemaks. Mainitud moodul ja sensorid võivad töötada ka akutoitel. Täiendavaks eeliseks on NRED veebibrauseriliides, mis kõrvaldab aega nõudva programmeerimise ja on ühilduv suure hulga riistvaraga, muutes süsteemi veelgi paindlikumaks. Kõigist käesolevas töös vaadeldud juhtmevabadest tehnoloogiatest osutus antud süsteemi jaoks sobivaimaks Wi-Fi, tagades turvalisuse, töökindluse ja ühilduvuse asjade interneti (Internet of Things) rakendustega. Antud JKA süsteem on võimeline teavitama kasutajat e-maili teel liikumisandurite häiretest ja võimaldab sündmuste käivitamist nii ajapõhiselt kui ka sensorite andmete järgi. Lisaks võimaldab antud seade vooluvõrgu pistikupesade kaugjuhtimist. Edasine töö antud teemal hõlmaks süsteemi täiendamist kaameraga, kodu jälgimiseks eemal olles. Seda oleks lihtne ellu viia teades, et RPI-1 on olemas kaameraliides. Edasiseks arenduseks võiks olla ka Android-rakendus, millega saaks juhtida JKA süsteemi, muuhulgas ka häälkäsklusi kasutades.

## 10 SUMMARY

Home automation systems are getting more and more popular these days along the masses as they are becoming cheaper and more affordable. With the advent of smartphones, tablets and the wireless technology and also with the improvement of the internet connectivity all over the world, it has made the wireless home automation (WHA) more reliable and easier to be implemented on our systems.

The thesis work treated a WHA which is using open source hardware and software, making it cheap and affordable. The system is using different sensors to gather data about the home conditions. The data then are sent wirelessly using safe and open source MQTT protocol to the RPI local server of the Node-Red via internet. The NRED is operating on RPI and we can interface it using any web-browser enabled device, to control the WHA or to monitor the data. Our WHA system can be used as a security system for our home, as a home automation system which allows remote control of home appliances and as a smart home system which is able to trigger events on certain conditions without the need of system monitoring, thus automating our home. The system is very flexible to upgrade with extra sensors. The esp-e12 Wi-Fi module eliminates the need of using wired connections with the sensors thus, simplifying our system and making it more practical. We can also power this sensors and the module using batteries.

Another advantage is the NRED web-browser interface which avoids unnecessary time consuming programing and is compatible with many hardware's, thus making the system easier to be upgraded with new sensors and hardware's.

Among all the wireless technologies that we reviewed on this thesis, we chose the Wi-Fi technology as the most appropriate for our application, with the advantages in security, reliability, compatible and low cost for Internet of Things application.

Our home automation system is able to notify us by email in cases of any motion detections in our home, it can trigger time based events, events depending on sensor values, it can trigger an alarm based on sensor data and it allows us to control power sockets remotely.

The future work on this topic could be the upgrade of the system with camera to remotely monitor the home while we are away, (this can easily be done also due to the fact that RPI supports camera interface), or to build an android application that can remotely control the system using voice command recognition technology, and many other features.

## 11 REFERENCES

[1] Raspberry Pi IoT Projects

- <https://link.springer.com/book/10.1007/978-1-4842-1377-3>

[2] Smart Home Area Networks Protocols within the Smart Grid Context (Ayesha Hafeez, Nourhan H. Kandil, Ban Al-Omar, T. Landolsi, and A. R. Al-Ali Computer Science and Engineering Department, American University of Sharjah, UAE Email: {g00045521, g00042737, g00029863, tlandolsi, aali}@aus.edu)

[3] Wireless connectivity for the Internet of Things – Texas Instruments

[4] Home Automation Systems - A Study - International Journal of Computer Applications (0975 – 8887) Volume 116 – No. 11, April 2015

[5] Evaluation of wireless home automation technologies – Conference Paper July 2011  
DOI: 10.1109/DEST.2011.5936601 (Source: IEEE Xplore)

[6] The Current State of Security in Smart Home Systems Threats in the Internet of Things - Version: V1.0 | Date: 2016-01-25 Author: | Responsible: Daniel Schwarz, SEC Consult Vulnerability Lab – Vienna

[7] Esp8266 power consumption - <http://bbs.espressif.com/viewtopic.php?t=133>

[8] Development of Wi-Fi Based Home Energy Monitoring System for Green Internet of Things - Mohamed Hadi Habaebi, Qazi Mamoon Ashraf, Amir Alif Bin Azman, and Md. Rafiqul Islam - JOURNAL OF ELECTRONIC SCIENCE AND TECHNOLOGY, VOL. 14, NO. 3, SEPTEMBER 2016

[9] Coping with Wi-Fi's biggest problem: interference  
(<http://www.networkworld.com/article/2215287/tech-primers/coping-with-wi-fi-s-biggest-problem--interference.html>)

[10] WEB BASED REAL-TIME HOME AUTOMATION AND SECURITY SYSTEM - Subhajit Dey<sup>1\*</sup>, Tamaghna Kundu<sup>1</sup>, Sourav Mukherjee<sup>1</sup> and Mili Sarkar<sup>1</sup>

[11] Wi-Fi Based Remotely Operated Smart Home Automated System using the Concept of Internet of Things - International Journal of Advanced Research in Electrical, Electronics and



- [12] ESP8266 module comparison: ESP-01, ESP-05, ESP-12, ESP-201, Test Board and NodeMCU - (<https://blog.squix.org/2015/03/esp8266-module-comparison-esp-01-esp-05.html>)
- [13] HOME AUTOMATION USING ARDUINO WIFI MODULE ESP8266 (<https://www.aiktcdspace.org:8080/jspui/handle/123456789/1558>)
- [14] Esp E-12 WI-FI Module – (<https://mintbox.in/media/esp-12e.pdf>)
- [15] Node-RED introduction – (<http://flows.nodered.org/>)
- [16] <https://www.hackster.io/ada-energy-17/home-automation-using-raspberry-pi-2-and-node-red-ed8ae>
- [17] <https://www.raspberrypi.org/downloads>
- [18] [https://drive.google.com/open?id=0B\\_BzxRImIcd-SWE2NVNPT1FycFE](https://drive.google.com/open?id=0B_BzxRImIcd-SWE2NVNPT1FycFE)
- [19] <https://github.com/knolleary/pubsubclient/archive/master.zip>
- [20] Renkforce Funk-Schalter-Set – (<http://www.ebay.de/itm/Renkforce-Funk-Schalter-Set-4teilig-Innenbereich-1208454-/222456853089?hash=item33cb763261:g:rooAAOSwSIBY3Ox7>)
- [21] RC switch library – (<https://github.com/sui77/rc-switch/archive/master.zip>)
- [22] Adafruit DHT11 library – (<https://github.com/adafruit/DHT-sensor-library>)
- [23] RPI Static IP - <https://www.modmypi.com/blog/how-to-give-your-raspberry-pi-a-static-ip-address-update>
- [24] G. Castagnoli, S. Brauer, and M. Herrmann, "Optimization of cyclic redundancy-check codes with 24 and 32 parity bits," IEEE Transactions on Communications, vol. 41, pp. 883 - 892, August 2002.
- [25] G. Castagnoli, J. Ganz, and P. Graber, "Optimum cycle redundancycheck codes with 16-bit redundancy," IEEE Transactions on Communications, vol. 38, pp. 111 - 114, August 2002.

- [26] M. Z. Huq and S. Islam, "Home area network technology assessment for demand response in smart grid environment," in Proc. 20th Australasian Universities Power Engineering Conference, 2010, pp. 1-6.
- [27] T. Han, B. J. Han, L. Zhang, X. Zhang, and D. C. Yang, "Coexistence study for wifi and zigbee under smart home scenarios," in Proc. 3rd IEEE International Conference on Network Infrastructure and Digital Content, 2012, pp. 669-674,
- [28] R. Melicio, J. C. O. Matias, J. P. S. Catalao, and N. C. Batista, "ZigBee wireless area network for home automation and energy management: Field trials and installation approaches," in Proc. Innovative Smart Grid Technologies Europe 3rd IEEE PES International Conference and Exhibition, 2012, pp. 1-5.
- [29] Wi-Fi-Advantages and Disadvantages. (2012) [Online]. Available: <http://digitalwalt.com/wi-fi-advantages-disadvantages/>
- [30] M. A. Sarijari, A. Lo, M. S. Abdullah, S. H. De Groot, I. G. M. M. Niemegeers, and R. A. Rashid, "Coexistence of heterogeneous and homogeneous wireless technologies in smart grid-home area network," in Proc. 19th International Conference on Parallel and Distributed, 2013, pp. 576-581.
- [31] F. A. Khan, A. Shahid, Z. A. Khan, and S. A. Khan, "ZigBee based reconfigurable clustered home area network," in Proc. Third International Conference Sensor Technologies and Applications, 2009, pp. 32-37.
- [32] Security: Smart Homes Using Internet of Things (IOT) - International Engineering Research Journal (IERJ) Volume 2 Issue 2 Page 558-561, 2016, ISSN 2395-1621
- [33] Design of an IoT Multi device for smart homes - International Research Journal of Engineering and Technology (IRJET)
- [34] IoT temperature and moisture sensor integrated to Node-RED and Bluemix IBM <https://developer.ibm.com/recipes/tutorials/iot-temperature-and-moisture-sensor-integrated-to-node-red-and-bluemix-ibm/>
- [35] DHT11 interfacing with Matlab and Arduino – (<http://embedded-electronics.blogspot.de/2014/11/dht11-interfacing-with-matlab-and.html>)

## 12 APPENDICES

### Appendix 1

```
#include <DHT11.h>
#include <LiquidCrystal.h>

int pin=8;
DHT11 dht11(pin);
void setup()
{
  Serial.begin(9600);
  while (!Serial) {
    ;
  }
}

void loop()
{
  int err;
  float temp, humi;

  if((err=dht11.read(humi, temp))==0)
  {
    //Serial.print("temperature:");
    Serial.print(temp);

    delay(DHT11_RETRY_DELAY); //delay for reread
  }
```

### Appendix 2

```
s = serial('COM3');
time=100;
i=1;
while(i<time)

fopen(s)
fprintf(s, 'Your serial data goes here')
out = fscanf(s)

Temp(i)=str2num(out(1:4));
subplot(211);
plot(Temp,'g');
axis([0,time,20,50]);
title('Parameter: DHT11 Temperature');
```

```
xlabel('---> time in x*0.02 sec');  
ylabel('---> Temperature');  
grid
```

```
Humi(i)=str2num(out(5:9));  
subplot(212);  
plot(Humi,'m');  
axis([0,time,25,100]);  
title('Parameter: DHT11 Humidity');  
xlabel('---> time in x*0.02 sec');  
ylabel('---> % of Humidity ');  
grid
```

```
fclose(s)  
i=i+1;  
drawnow;  
end  
delete(s)  
clear s
```