ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΥΣΤΗΜΑ ΕΞΥΠΝΟΥ ΣΠΙΤΙΟΥ & ΦΡΟΝΤΙΔΑ ΚΑΤΟΙΚΙΔΙΟΥ

Του φοιτητή
Παυλίδη Μιχαήλ
Αρ. Μητρώου: 164729

Επιβλέπων
Γιακουμής Άγγελος
Επίκουρος καθηγητής

9 Μαρτίου 2023

ΣΥΣΤΗΜΑ ΕΞΥΠΝΟΥ ΣΠΙΤΙΟΥ & ΦΡΟΝΤΙΔΑ ΚΑΤΟΙΚΙΔΙΟΥ
Κωδικός Δ.Ε. 22164
Παυλίδης Μιχαήλ
Επιβλέπων καθηγητής: Γιακουμής Άγγελος
Ημερομηνία ανάληψης Δ.Ε. 18-03-2022
Ημερομηνία περάτωσης Δ.Ε. Ιούνιος 2023

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Αφιέρωση»
Στους γονείς μου (Γιώργο,Καλλιόπη) και στα αδέρφια μου(Θοδωρή,Νίκο)!

Πρόλογος

Ο λόγος που επέλεξα την συγκεκριμένη εργασία είναι ότι θεωρώ πως ο συνδυασμός hardware και software μπορεί να φέρει άπειρους συνδυασμούς στους οποίος μπορεί να επωφελειθεί ο καθένας μας. Είναι εντυπωσιακό που η κάθε μικρή αυτοματοποιημένη δημιουργία μπορεί να βελτιώσει τον τρόπο ζωής κάποιου και να την διευκολύνει.

Περίληψη

Η συγκεκριμένη μελέτη πρόκειται για μία έρευνα και υλοποίηση ενός συστήματος έξυπνου σπιτιού που θα βοηθήσει τον αναγνώστη να κατανοήσει καλύτερα αυτά τα συστήματα ή θα τον ωθήσει να δημιουργήσει ένα δικό του. Ο συνδυασμός hardware και software μπορεί να μας δώσει άπειρους συνδυασμούς και εφευρετικούς τρόπους ώστε ο άνθρωπος να συνεχίζει να προοδεύει ή να βελτιώνει τον τρόπο ζωής του. Ένα σύστημα έξυπνου σπιτιού, διευκολύνει την καθημερινότητα του χρήστη, εκτελώντας διεργασίες απομακρυσμένα χωρίς να απαιτείται η φυσική του παρουσία στο σπίτι ή το ίδιο το σύστημα τις εκτελεί ανάλογα με το τρόπο σχεδιασμού του ή τις προϋποθέσεις που έχει θέσει ο ίδιος ο χρήστης.

«Smart Home System»

Pavlidis Michail

Abstract

This project is about researching and constructing a smart home system to help the reader understand those systems much better or exhort him to make one of his own. The combination of hardware and software can give us infinite combinations and creative ways so that humans keep progressing and improving their way of life. A smart home system can make the user's life easier, as it accomplishes processes remotely without the user being in the house. The processes can be executed automatically depending on how the system is made or the user's needs.

Ευχαριστίες

Όπως και στη ζωή έτσι και σε μιά διπλωματική εργασία δεν είναι εύκολο να επιτευχθεί κάτι χωρίς να υπάρχει οποιαδήποτε μορφή στήριξης από μια ομάδα ανθρώπων που περιβάλει το άτομο. Αν και αυτή η εργασία υλοποιήθηκε από τον συγγραφέα της θα είναι αγένεια να μην αναφερθούν τα άτομα που βοήθησαν, Επομένως σε αυτό το κομμάτι θέλω προσωπικά να ευχαριστήσω όσα άτομα έχουν σταθεί μαζί μου και με έχουν στηρίξει όχι μόνο κατά την διάρκεια της πραγματοποίησης αυτής της εργασίας αλλά και στην διάρκεια της ζωής μου.

Ξεκινώντας θέλω να ευχαριστήσω τους γονείς μου που με στήριξαν και με στηρίζουν με τόσους τρόπους που ακόμα και αυτές οι λέξεις δεν είναι αρκετό για να τους πώ ευχαριστώ για όλη την υποστήριξη που μου έδωσαν καθόλη την διάρκεια τις ακαδημαικής μου ζωής.

Τα αδέρφια μου που εκτός από αδέρφια υπήρξαν φίλοι,κολλητοί και συγκάτοικοι που ακόμα και αν μας χωρίζουν αποστάσεις χιλιομέτρων ξέρουμε ότι θα είμαστε δίπλα ο ένας για τον άλλο.

Τον Παππού μου Θεόδωρο και την Γιαγιά μου Δωροθέα που μου υπενθυμίζουν συνέχεια να βάζω τον καλύτερό μου εαυτό σε ότι κάνω για να προοδεύσω στην ζωή.

Μετά την οικογένεια θέλω να ευχαριστήσω τους κολλητούς και συνάδελφους Βασίλειο Κουκογιώργο και Νικόλαο Τσακίρη που γνωριστήκαμε και οι τρεις στην αρχή των ακαδημαϊκών μας χρόνων και μου μάθανε (και ακόμα μου μαθαίνουν) πως να προοδεύσω στον τομέα της Πληροφορικής.

Επίσης να ευχαριστήσω την Εταιρεία Ham Systems και όλη την ομάδα της, μιας και μου δώσανε την ευκαιρία να εργαστώ μαζί τους και παρείχαν γνώσεις & υλικά για την επίτευξη αυτής της εργασίας.

Τέλος να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Άγγελο Γιακουμή που συμφώνησε να υλοποιηθεί αυτή η εργασία και υπήρξε σωστή πηγή προσθήκης περιεχομένου σε αυτή την εργασία.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

IoT	Internet of Things
HRD	High Dynamic Range
IDE	Integrated Development Environment
AES	Advanced Encryption Standard
I2C	Inter-Integrated Circuit
GPIO	General Purpose Input-Output
SoC	System On a Chip
LPG	Liquefied Petroleum Gas
MQTT	Message Queue Telemetry Transport
RFID	Radio Frequency Identification
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
PWM	Pulse Width Modulation
I2S	Inter-IC Sound
VCC	Voltage Common Collector
GND	Ground
IR	Infrared
MISO	Master In Slave out
MOSI	Master Out Slave In
PIR	Passive Infrared
PWM	Pulse Width Modulation
DC	Direct Current
A/C	Air Condition
mm	millimeter
nm	nanometer
NO	Normally Open
NC	Normally Closed
COM	Common
UI	User Interface
API	Application Programming Interface
PCB	Printed Circuit Board

Chapter 1: Introduction

1.1 Smart home

A smart home system is known as a home setup where devices can be controlled automatically or remotely by the user. The devices used in a smart home system are connected through the internet and the user only needs an internet connection or a network device (a device that connects the rest) to gain access and control his home devices [1]. A smart home can be set up through wired or wireless systems, which can be convenient and money-saving for the user [2]. The user has the opportunity to gain information or control functions that can be:

- Temperature and humidity[1]
- Lighting
- Heating
- Audio & Visual
- Security Surveillance
- Monitoring Energy consumption
- Taking care of a pet by providing it with food and water

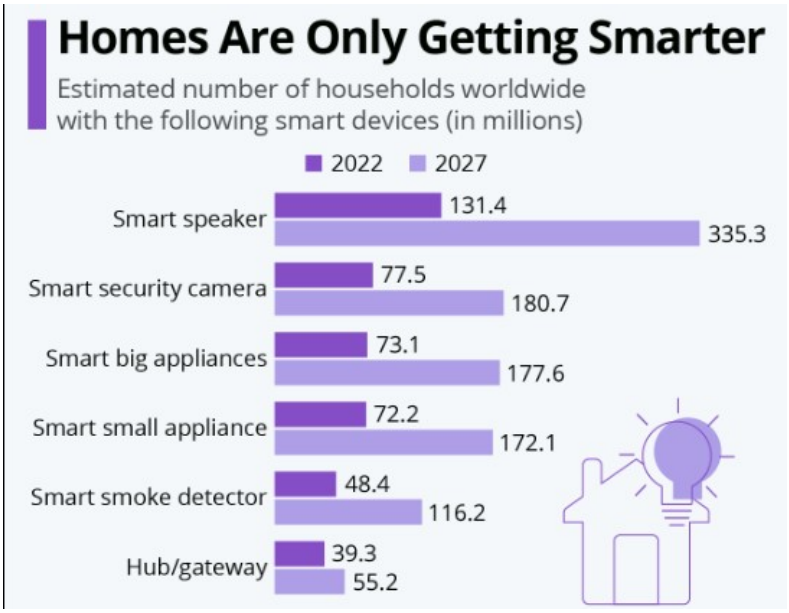


Figure 1.1: Smart home devices and applications per household are predicted to increase
Source

Thanks to smart home systems, the user can control his home devices and appliances remotely.

[1] Humidity is the concentration of water vapor in the air

1.1.1 What is a smart device

A smart home system needs smart devices connected to each other or connected to a network that uses wireless protocols (WiFi or Bluetooth). But what makes a device smart?

1. **Context awareness:** The device has to gather information about its environment, and according to that data, the device has to act accordingly.

2. **Autonomous computing**: The device can perform tasks without the direct command of its user

3. **Connectivity**: The device can connect to a network (wired or wireless). This is important since a device without connectivity would be useless for context awareness and autonomous computing. While most smart devices do not require human interaction, we must consider that most do. A smart device can have a direct (smartphone) or indirect (temperature) probe interaction with its user, but what matters most is that, in the end, the user will use the device's data. A smart home system can make the user's life easier since many functions can be done automatically. Moreover, it can reduce unnecessary electricity usage and, as so, reduce cost, which is energy related. To create a smart home system, we need to create a way to get information or control them remotely. [3]

1.2 History of smart homes

1.2.1 Before smart homes

Figure 1.2: One of the first X10 products: A remote lamp controller
Source

While the terminology of the smart home did not exist yet, in the 1900s, with the introduction of electric power distribution, the first electric devices were made to save time in-home labor. Thanks to electrical devices such as washing machines, water heaters, and refrigerators became the first devices of home automation. These home devices were not considered smart since their users had to adjust them to work, but thanks to electricity, they were the first autonomous devices that did not demand a massive human effort to function. [4]

1.2.2 The first smart home system

[5] Years later, in 1975, in Scotland, Pico Electronics developed and released X10, a general-purpose home automation platform still in use today. Back then, this platform sent digital data through radio frequency bursts to the house's electrical wiring to remotely control home devices.

The first home automation system arrived in households, however, these systems were expensive, and they had many issues, such as:

- Slow response times
- Unencrypted data
- Signal loss and interference if a neighbor had the same system.
- System relied on heavy power lines

1.2.3 Millennium era

Figure 1.3: The Zigbee module was one of the first wireless- low power technologies for personal area networks

Source

In the early 2000s, most smart home problems were solved thanks to Z-wave and Zigbee wireless technologies. While both technologies can offer many features for a mesh network, they have the edge over the other in different categories. Zigbee devices can be faster and handle more devices in a mesh network, but Z-Wave does not interfere with WiFi and demands less power to run [6]. Both technologies use different radio frequencies (Zigbee 2.4GHz - Z-Wave 908.42MHz), which can be affected depending on the type of devices we connect. When both of them were released in public, it was the era in which most security protocols were made since these devices have AES-128 encryption currently used in financial institutions. Thanks to Z-wave and Zigbee, everyone could have access to that technology without the need to use cables. Lastly, these wireless technologies provided enough bandwidth to connect multiple devices and control them remotely, which previous technologies struggled with.

1.2.4 2010

After 2010 many major companies started getting interested in smart home systems. One of them was a new company called Nest Labs, created by an ex-engineer of Apple, Tony Fadell. Fadell managed to build Wi-Fi devices such as smoke alarms (Nest Protect) and thermostats (Nest Learning Thermostat).

Figure 1.4: A Google Nest Thermostat monitors and controls room temperature
Source

Later on, Nest was bought by Google, making new versions of the previous devices and introducing new smart home products such as:

- **Nest Cam Indoor**: This device was made for indoor security surveillance, featuring excellent video resolution, night vision, and motion alerts. Moreover, this device had a**two-way talk** feature, which allowed the user to talk and listen through the camera.
- **Nest Cam Outdoor**: Like the indoor version, this one was made for outdoor surveillance, as the design was made to endure outdoor conditions.
- **Nest Cam IQ**: This product is an upgraded model of the Nest Cam Indoor, as it features a 4K camera sensor with HDR. Also, this camera was equipped with advanced artificial intelligence, which could differentiate people from moving objects and recognize human faces.
- **Nest Secure**: This system is an alarm system that includes three different devices:
  - **Nest Guard**: Allows to turn security on and off, detects motion, and sounds the alarm.
  - **Nest Detects**: It is installed on a door or window and informs the user when it is opened or closed.
  - **Nest Tag**: Turns the alarm on and off by using it on Nest Guard.

  This system could also be controlled by using the Nest app, allowing the user to arm or disarm the alarm and get a notification if the user forgot to set the alarm or if some activity was recorded when the user wasn't at home.

- **Nest Doorbell**[2]: This device functions as a doorbell with facial recognition. Like the Nest Cam products, Nest Doorbell detects motion and allows the user to talk and hear others in the sight of the camera.

1.2.5 2012

Two years later, smart devices could be controlled and monitored with an application called SmartThings which 2014 was bought by Samsung. The Smart things app was the first certified platform that allowed users to control and monitor the devices' energy usage in real-time. Thanks to this mobile application, devices registered on a server could be controlled remotely, from home appliances to TVs and speakers.

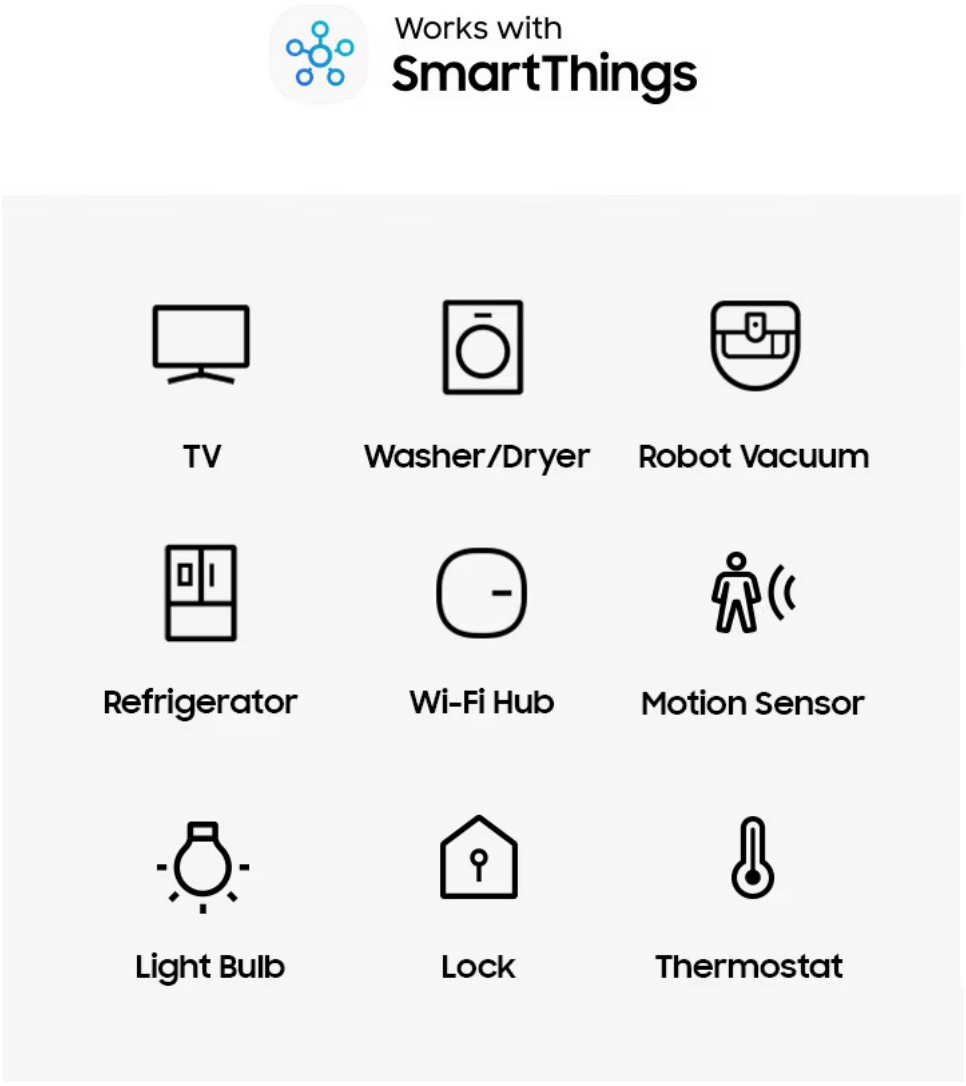


Figure 1.5: With the Smart Thing application we can control devices and automate home applications
Source

[2]Known as Nest Hello when it was first released

1.2.6 Nowadays

Today smart home systems are available and open-source to anyone despite their knowledge of smart home technology. Open-source projects such as HomeAssistant, OpenHab, and Calaos [7] allow technology enthusiasts to create their systems by their criteria. These open-source systems run on a local server (most use Raspberry Pis) and use libraries for the connected sensors and devices. Companies have started producing and selling products, and even individuals can make their systems, depending on their needs to experience a more comfortable way of life and have more sense of control in their household. As numerous companies are created or start to get interested in home automation, some companies have won customers, with their products being used daily, such as:

- Google Home
- Amazon Echo
- HomePod (Apple)

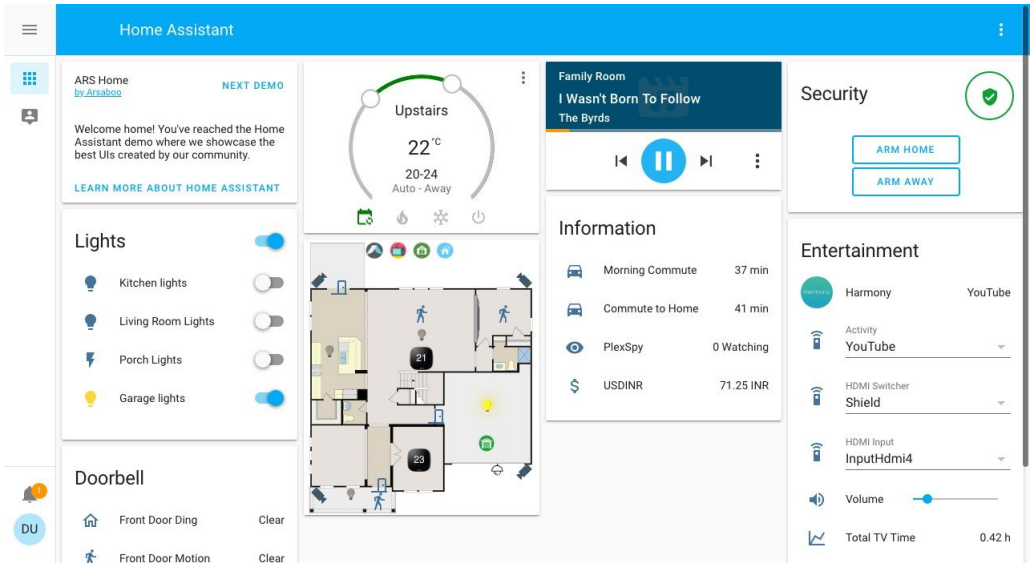


Figure 1.6: An example using the open source Home Assistant System
Source

1.3 Research on Smart Homes

According to research, people believe using smart home systems has improved their lifestyle by having control of their houses. As most smart home systems are installed by US companies (with China ranked second), by 2025, the number of smart homes used worldwide is forecasted to increase to 480 million [8]. Moreover, it is confirmed that many people would think of installing a smart home system, as younger people state that they would pay 20% more to have remote control in their houses [9]. Thanks to smart home technology, many products and devices are being used by elders with health issues who can control these devices even by using voice activation. As smart home systems are applied, a significant percentage of people affirm that they can save time and money by reducing unnecessary energy consumption from heating or cooling systems. Moreover, most users would invest in smart home security, from cameras

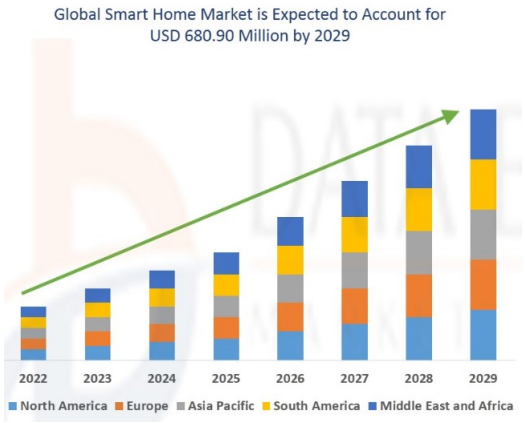


Figure 1.7: Since there is an increase in the smart home market, the demand from customers is going to increase

Source

that can be monitored remotely to smart locks that can be opened and locked by the user. Additionally, the user feels fulfilled when he monitors his house remotely and feels safer. Companies that sell smart home systems produce devices that are isolated from the rest in case of a cyber-attack so that the threat will not spread quickly to the rest of the system. [10]

Chapter 2: Hardware & Software

2.1 Arduino IDE

The Arduino Integrated Development Environment is a coding tool to program Arduino or other microcontrollers. The Arduino code (sketch) will be compiled and uploaded to the microcontroller. The microcontroller will translate the language of the sketch, and then it will remain in the board's memory.

The Arduino code is written in C++ using functions that do not exist in other programming languages and are necessary for Arduino sketches which are:

1. **Setup()**: The setup function is responsible for defining the board's initial state, pins, and other variables. After the sketch is uploaded to the board, all the coding inside this function will run only once, but they can change values with the loop function.
2. **Loop ()**: In this function, the code runs in a loop infinite times after the setup function is complete. Using the Loop, we can determine which values we want to manipulate or which actions can be done.

In the Arduino IDE, we can use libraries to control the hardware we will use. A ***library*** is a folder containing files with C++ (.cpp) code files that hold the function implementation and header files(.h) which describe the structure of the library and is used to declare all variables and functions. Arduino IDE has installed a few libraries, but not the ones we will use. There are two ways to gain access and use them:

1. Search them using ***Sketch > Include Library > Manage Libraries***
2. Download them from a third party and include them using a ***.zip*** file.

2.1.1 Libraries used in code

- **WiFi**
- **PubSubClient**
- **SHTSensor**
- **SPI**
- **MFRC522**
- **ESP32Servo**
- **IRremoteESP8266**
- **IRsend**
- **ir_Gree**

2.2 ESP32

An ESP32 is a low-cost - low-power Microcontroller with integrated Wi-Fi and dual-mode Bluetooth (Bluetooth 4.2 and low energy). Espressif Systems create this Microcontroller with a series of SoC. ESP32 is the descendant of ESP8266 and a much better microcontroller, as ESP32 includes an extra core, faster Wi-Fi, and more GPIO pins. ESP32 has an operating frequency of up to 160MHz and has an Xtensa Dual-Core 32-bit Microprocessor. It has 48 pins, of which 10 of them are not exposed, leaving us with

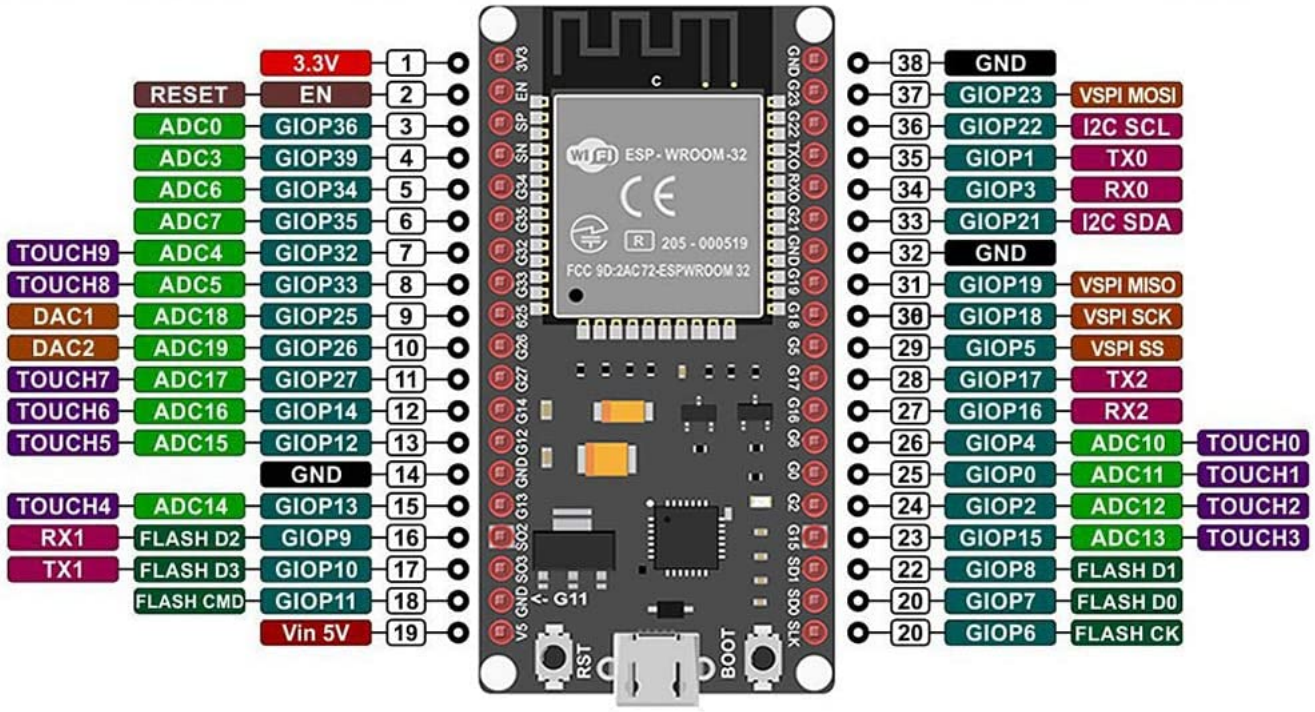


Figure 2.1: ESP 32 Pinout
Source

38 visible pins (depending on the version we choose). Since we have chosen an ESP-32 development board, 2 of its pins offer voltage of 3.3 and 5V separately. Moreover, the pins can be assigned different functions, which will be decided in the code. In summary, the ESP32 pins are:

- 18 Analog-to-Digital Converter channels
- 3 SPI Interfaces
- 3 UART interfaces
- 2 I2C interfaces
- 16 PWM output channels
- 2 Digital-to-Analog Converters
- 2 I2S interfaces
- 10 Capacitive sensing GPIOs

2.2.1 USE ESP-32 on Arduino IDE

While Arduino IDE does not have a debugger, the user can exploit the serial monitor to monitor the values of the variables and detect any anomalies in his code.

As the Arduino IDE officially supports its official microcontroller boards, we can install other boards, like ESP32, to program them. In File>Preferences, we enter the following line: Afterwards, we search the:

Tools>Board>Board manager

for ESP32 by Espressif Systems and install it. To test if we can upload to our microcontroller, we have to connect it to the PC that runs Arduino IDE using a USB cable. Next, we choose from the programming environment the USB port the microcontroller is connected to, and then we choose the board ESP-32.

2.3 Raspberry Pi

Raspberry pi is known as a single-board computer created by Raspberry Pi Foundation. The Foundation's primary purpose was to encourage individuals (despite their knowledge) to get familiar with computing or get used to programming. However, Raspberry pi is now used for different applications to such a point that even industries use them since it is a low-cost computer that operates in an open-source ecosystem.

Figure 2.2: Raspberry pi 4
Source

As we mentioned before, most people use them to get familiar with Computer Science, mainly in programming. However, those experienced in that department can use it for electronic coding devices for

physical projects, and one of those projects can be a Smart Home system. For this project, we use a Raspberry pi 4 with 2GB RAM, which is faster than prior Raspberry pi models since it has a 1.5-GHz Broadcom CPU and a faster MicroSD card slot with an operating system running. In this project, the Raspberry pi will be used as a server, and it will run Node-RED and a communication protocol known as MQTT. [11]

2.4 Node-RED

Node-RED is a programming tool to connect the hardware devices and sensors with our server to get the data or use the devices remotely. It is a useful open-source tool for building IoT applications like ours. Node-RED runs on the web browser, and with the assistance of visual programming, we can connect nodes, known as code blocks, to achieve a task by receiving Payload messages. With Node-RED and an MQTT protocol, we can establish a connection with our ESP32. As we use Node-RED, we will construct a dashboard that is visualized using JSON programming language to view our home conditions, such as temperature or humidity, and even control some of our hardware, like lighting or our pet food container.

2.4.1 Having Node-Red on Raspberry Pi

To install Node-Red on Raspberry Pi we need to enter the following line in a terminal:

bash <(curl -sLhttps://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

After the installation, it would be preferable to configure Node-Red by using the command:

node-red admin init

Which, we can set up a security system by adding a username and password, after that we have to make Node-Red run automatically when the Raspberry pi boots up, using the command:

sudo systemctl enable nodered.service

To access the Node-Red, we type the Raspberry's IP address followed by:1880 and log in with the username/password credentials we set before. After gaining access, we can use Node-Red to create nodes and connect them with flows to perform tasks that will be required in our project.

In order to view the data from the devices or interact with them, we need to build a UI that Node-Red can provide by using a dashboard. By going to Menu>Manage Palette, we search for node-red dashboard and install it. The dashboard can be accessed by adding /ui in the URL address of the raspberry pi's IP, which will look like this:

http://Rasbperry-Pi-Adress:1880/ui

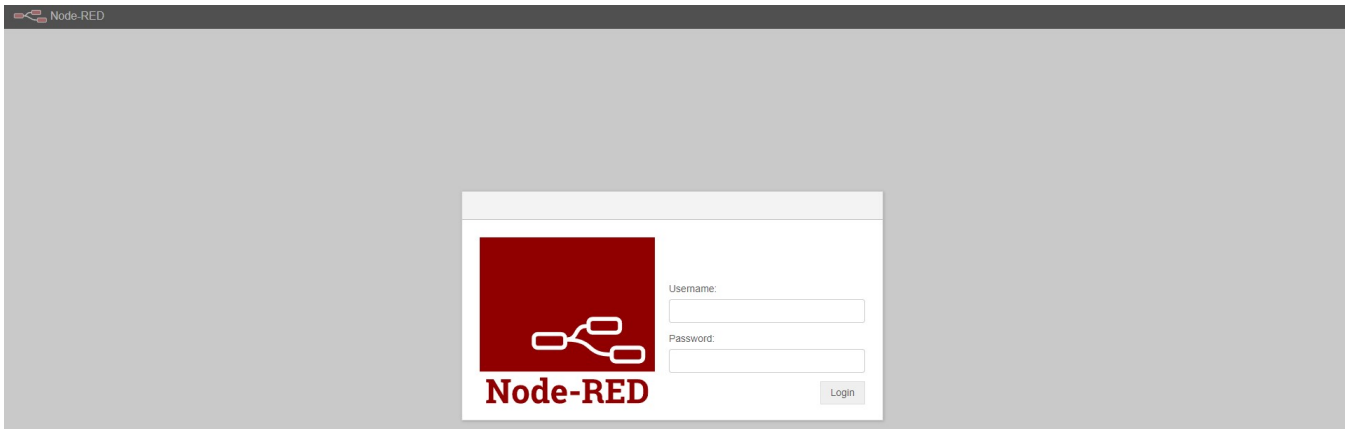


Figure 2.3: Node RED requires login credentials in order to create nodes and flows

Afterwards, we can edit the layout of the dashboard and arrange it however we want.

While we can use just our personal computer, our dashboard is considered a local site, and we cannot gain access remotely. However, Node-RED offers a proper tool to access our website locally and remotely.

2.4.2 Remote Red

Remote RED is a tool that allows us to access the dashboard using our smartphone. Since we live in times where a smartphone is required wherever we go, we could use it to view and control our dashboard remotely without the need to be in the same Wi-FI network that our MQTT broker is connected.

By installing the Remote-RED package with this command:

node-red-contrib-remote

We have to add a node known as a config node for the page we want to access and configure it. With this Node we can define settings like:

1. **Name**: The name of the page which will be displayed in the app.
2. **Serving host**: The local IP of the host whose UI is to be shared.
3. **Protocol**: The one which is used by the web server (HTTP)
4. **Serving Port**: The port that will be forwarded (1880)
5. **Base URL**: The initial page to be called
6. **Server location**: The server which will tunnel to the localhost. We have 3 options depending on our region:
    - **Europe**: Germany
    - **Asia**: China

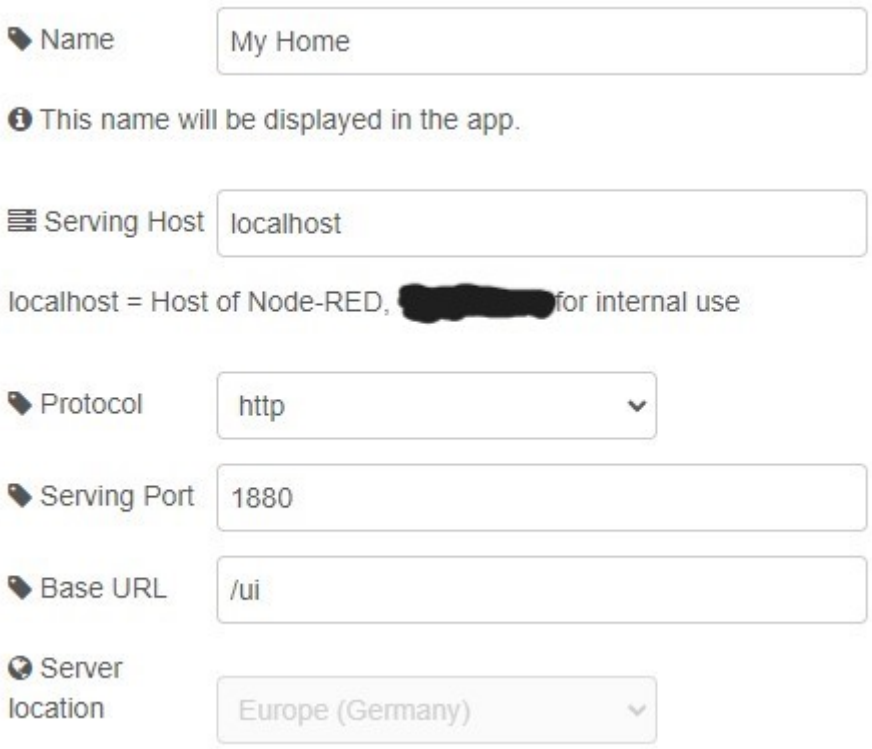


Figure 2.4: Using these configures we can use the Remote RED app with our smartphone

- **America**: USA

These servers create a tunnel to our local server/site, and we can open the Dashboard UI with our smartphone by installing the remote red application. After installing the application, we must connect our smartphone to the dashboard UI. In order to do that, we need to add a Node-Red Instance from our smartphone. An instance is a connection to the config node we are using in our dashboard. The application will demand a QR code which can be found in the config node settings. After scanning it, we can have access to the dashboard.

Additionally, with Remote red, we can program notifications (push notifications) to appear on the phone whenever necessary.

Using the notification nodes, we can optimize the title and the text of the notification, and at the time a message arrives at the notification node, the notification we configured is going to be seen on the device

2.5 MQTT

It is a standard messaging protocol that is very popular for IoT applications [12]. It is a simple and lightweight messaging protocol designed for devices that use minimal network bandwidth and connect them remotely, making it perfect for our project. We use the MQTT protocol so that our ESP-32 and Raspberry Pi exchange data with each other and control our devices. The reasons why we choose the MQTT protocol are:

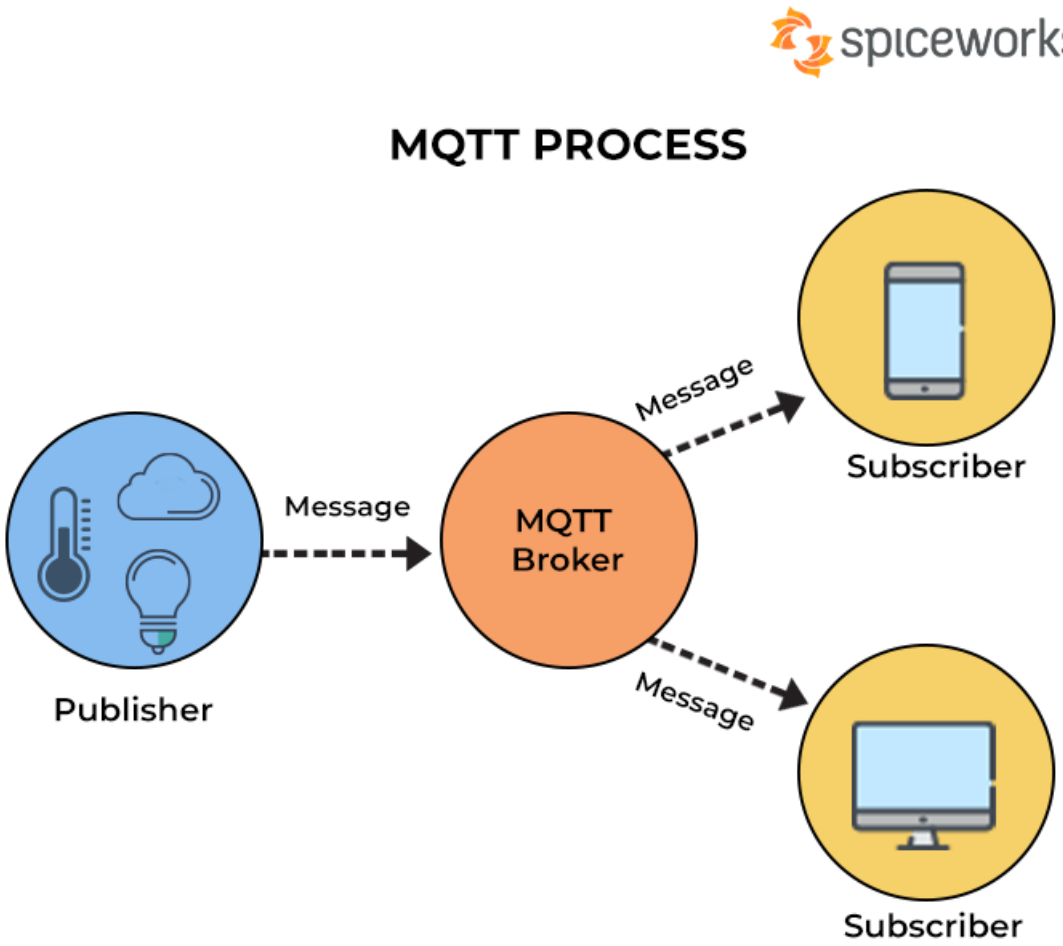


Figure 2.5: A simple explanation on MQTT
Source

1. **Efficient with minimal resources**
2. **Easy to use**
3. **Simple communication between devices**
4. **Two-way communication between the devices we use**
5. **Scalability with IoT devices**
6. **Reliability**

The MQTT protocol is not based on traditional client/server communication, but MQTT is built around an architecture that needs several devices to exchange data with each other in the form of **topic**s. These devices are known as **Clients** and **Brokers**.

- **Clients**: A client can be any device (ESP 32) or tool (Node-RED) that connects to the broker. Clients cannot connect and do not communicate directly with each other. Precisely the data ex-

change between devices can be accomplished by a proxy which in the MQTT protocol is known as a broker.

- **Broker**: The broker receives the client's request to receive data from one or multiple topics, and it acts as a proxy, making sure to receive or transfer data to the client.

Topics can be multiple, and we can specify how we want to receive that message. The topics are represented with strings separated by a forward slash, with each slash representing a topic level. The devices, however, need to be established in what topics they need to receive/send data, and the broker has to be configured so it knows where it needs to send the topic's data. For that reason, MQTT uses the **publish/-subscribe** method.

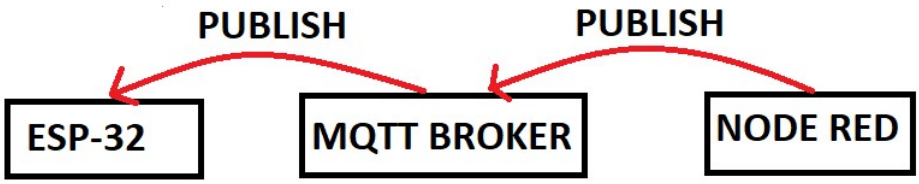


Figure 2.6: Node RED publishes data to the topics ESP-32 has subscribed

- **Subscribe**: As we said above, the clients do not connect directly with each other. In order to receive messages, the client needs to subscribe to a topic; to do that, the client has sent a Subscribe request to the broker. After the broker responds with a confirmed message that the subscription is acknowledged (known as **SUBACK**) it handles the client's subscription topic(s).
- **Publish**: In order to send or receive data, the clients need to publish the data to the broker in the form of a topic. As the broker receives the data, it checks which clients have subscribed to this topic and transfers the data to those who have subscribed..

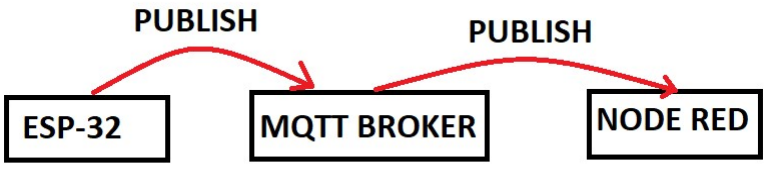


Figure 2.7: ESP-32 publishes data that can be viewed to Node RED

The MQTT broker is responsible for receiving all messages and sending them where they are meant to be sent by publishing the message to the clients that have subscribed to the topic. In our project, we will use an open-source broker made by Eclipse, known as Mosquitto.

2.5.1 Mosquitto Eclipse

[13]

The Mosquitto broker is an open-source message broker made by Eclipse that implements the MQTT protocol and is also known as a lightweight and practical broker used on devices such as single-board computers or even full-functioning servers. Because it is lightweight, it can deliver messages competently using the publish/subscribe model we mentioned above, making it perfect for IoT projects that use low-power devices and sensors.

Figure 2.8: Mosquitto Eclipse is an open-source MQTT broker
Source

The Mosquitto broker is going to be installed in our Raspberry pi, using its terminal window. By running the following command:

sudo apt update && sudo apt upgrade

Next, we need to configure the Mosquitto broker so that it will start automatically when the Raspberry Pi boots by using this command:

sudo systemctl enable mosquitto.service

2.6 Relay module

A relay module is an electrical switch that can be activated by the signal of a microcontroller which can activate or turn off a device connected to the relay. There are two led indicators to confirm if the relay is working and if it is activated or not. The red one, which indicates the device's power, turns on if the relay is powered, while the other one (green) turns on whenever the relay is active, indicating the device's status and if current flows through the relay coil. [14] To ensure the connections are reliable, the devices we want to control are screwed through terminal blocks, making wires difficult to solder connect directly. A relay module has three pins that require connection with jumper wires:

- **VCC**: Which will be 5V.
- **GND**: Ground of the relay.

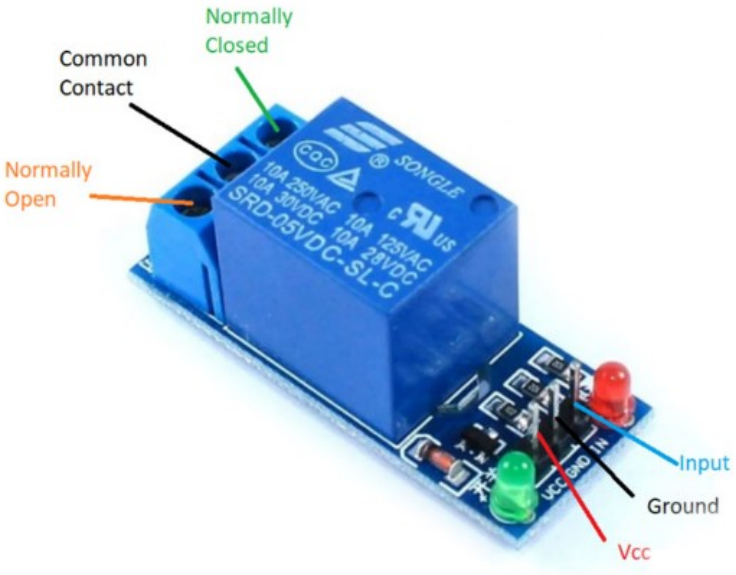


Figure 2.9: A 1-channel 5V Relay module
Source

- **IN**: The digital input pin will be connected to the microcontroller.

Moreover, it has 3 terminal blocks to connect a device

- **NO**: Normally open means the relay's switch will be open (The device is closed).
- **NC**: Normally closed means the relay's switch will be closed (The device is open).
- **COM**: In the common terminal we connect the device we want to control.

We will use two relay modules of 5V for two devices which we will explain in the next chapters.

2.6.1 Office Lamp

While an office lamp is a simple device for lighting, it is an excellent example of how to remotely control lighting devices (light bulbs or even led strips). In order to control the office lamp will require connecting the lamp's cable with the relay module. While an electrical socket will power up the lamp, one of the wires of the lamp's cable will be connected with the COM terminal, and it will be continued through the NO terminal since, when our system boots up, we want the device closed.

2.6.2 Water pump

A water pump is used in order to flow water. Water pumps can be used in IoT applications, either small (gardening) or large (agricultural purposes). For our project, we will use a water pump to provide water for our pet. The water pump we will use is a low-cost one that requires 2.5-6V DC voltage. This water pump will be placed in a "reservoir" full of water, and because the pump is submersible, it can be placed inside the water (the water levels have to be higher than the pump). Using a water level module (which

we will explain in chapter 2.11), we will measure if the water levels are high enough so that our pet will have enough water for the day. The water pump positive wire is connected with a relay module to the NO terminal, so the default behavior of the pump will be off when not used. The COM terminal and the negative wire of the pump are connected to a 9V battery snap connector, where a 9V battery will power the pump.

Figure 2.10: The water pump is deployed when the water level is low
Source

2.7 SHT 40

Measuring the temperature and humidity of the room is expected in smart home systems. The user needs to know the conditions of his home so that the user can heat or cool his home and even stop devices that can change the temperature (A/Cs or thermostats) so that energy consumption can decrease and save money. In order to measure the conditions of the room, we are going to use an SHT40 sensor probe.

Figure 2.11: An SHT 40 Probe for temperature and humidity measurements
Source

This is a sensor made from Adafruit in the form of a probe that can detect the temperature and humidity of the area in which our sensor is deployed. This device uses a chip-to-chip protocol known as I2C, which helps communicate with low-speed peripherals. Moreover, the power supply of this sensor ranges from 3.3V-5V, and it can measure humidity and temperatures from -40 to 125 degrees Celsius.

- **VCC**: Operating voltage from 3.3V to 5V
- **GND**: Ground pin

- **SDA**: Serial Data pin
- **SCL**: Serial Clock pin

2.7.1 DHT

Another great alternative for temperature and humidity sensors would be one of the DHT sensor series (Another product of Adafruit). Either DHT11 or DHT 22 are excellent choices since both return almost accurate temperature and humidity sensors, but DHT22 has better measurement ratings in a broader range than DHT11. While the DHT22 is more accurate in humidity readings, it would be more efficient if

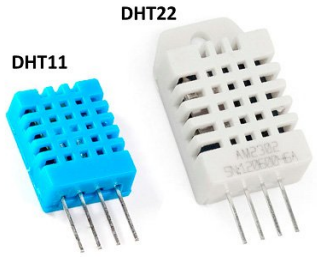


Figure 2.12: Both DHT sensors are used for temperature and Humidity readings
Source

we used SHT sensors since they have more precise measurements in both Temperature and Humidity readings. [15]

2.8 MAGNETIC SENSORS

Figure 2.13: A magnetic door sensor can be used to alert us when a door or window is open
Source

Alternatively, magnetic locks are a simple yet effective way to increase security in a smart home system. They can be used in doors and windows so that the user gets updated if he forgot them open or if there is a breach so that the user will be informed about a possible break-in and an alarm will be triggered.

Additionally, they can be used in refrigerators in case the user forgets the fridge door open so that the food does not spoil. In our project, we will place the contact sensors at the apartment's main door. One is placed at the door, and the other at the door frame while connected to a digital pin of the ESP 32 and ground. They create a magnetic field indicating that the door is closed when they are near each other. When the door opens, the distance increases between them, and the magnetic contacts will separate, breaking the magnetic field and detecting that the door is open. While it would be rational to use wireless contacts, they are more easily worn. Additionally, wired magnetic contacts can draw power from the microcontroller, and they do not need battery changing, unlike wireless Bluetooth sensors.

2.9 RFID

RFID, also known as Radio Frequency Identification, is a system that uses electromagnetic fields to recognize and identify tags embedded in particular objects (cards or keychains). [16] RFIDs are helpful for home security since they can provide access to the house and even disarm an alarm timer when the user opens the door. An RFID needs two things to work:

1. **An RFID tag**: The tag is a device that contains a microchip that contains information. Since it has no batteries it is considered a passive device and is powered by the RFID reader.

2. **An RFID reader**: The reader is used in order to read the tag

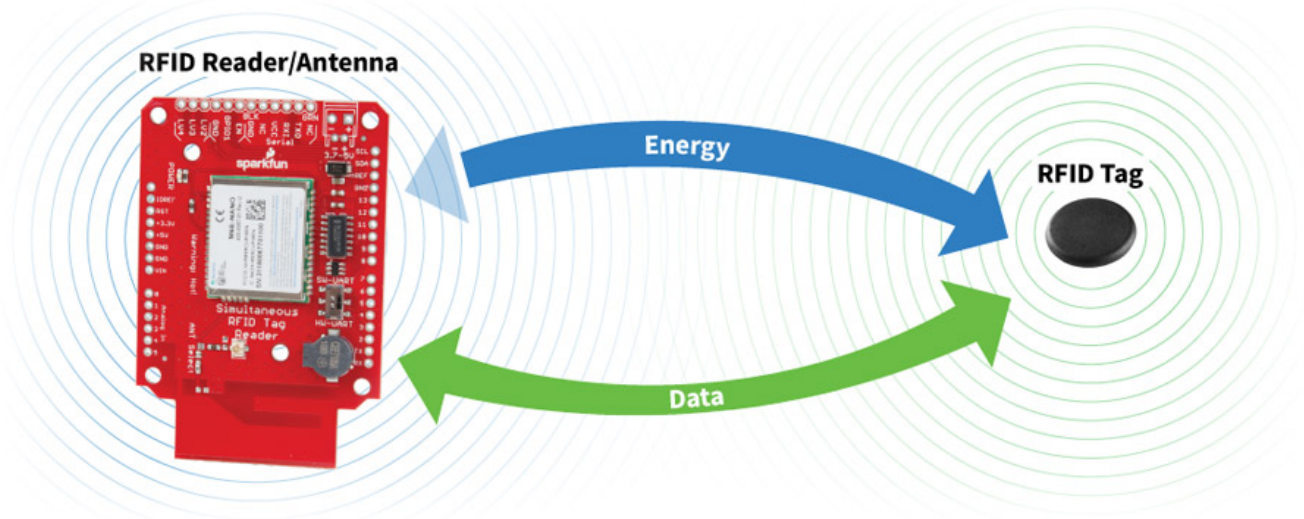


Figure 2.14: The RFID tag is powered up when it is nead the reader
Source

It is essential to mention that tags and readers contain antennas to communicate when they get close to each other. When the RFID tag approaches the reader, the latter generates an electromagnetic field that goes through the antenna of the RFID tag and powers up its chip. The chip sends its stored information, including a 96-bit binary number; for simplicity, we use its hexadecimal form. This hexadecimal number is transferred to the reader as another radio signal (backscatter). When the reader detects this phenomenon, it sends the information to a computer or, in our case, to the ESP32.

2.9.1 MFRC522

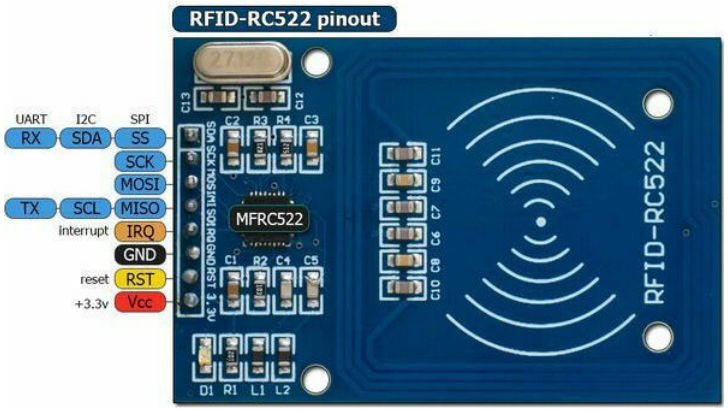


Figure 2.15: An RC522 RFID reader
Source

This module, built by the NXP Company, is an integrated read-and-write card chip for contactless communication and one of the cheapest and most reliable solutions. It also creates a 13.56 MHz electromagnetic field to communicate with RFID tags. This module can be powered easily with the ESP-32 since its voltage can be until 3.3V. An RC522 RFID Module has eight pins which we can see in table 2.1. As

Table 2.1: RFID PINS

VCC	The pin which is responsible for the supply of power.
RST	A reset and power down pin.
GND	Ground pin
IRQ	An interrupt pin that can alert ESP-32 when an RFID tag is nearby.
MISO	This pin acts depending on the usage we will initiate.
MOSI	An SPI input to the module
SCK	Accepts the clock pulses provided by the SPI of ESP-32
SDA	Serial Data pin, that its behavior depends on how we will initialize it.

the table mentions,each pin has a specific role. Firstly the VCC pin gets a power supply from 2.5-3.3V and with the GND pin we ground our module. The RST pin determines the module's mode, as when it goes low the module goes to power down mode and resets when the module's signal is on a rising edge the module resets. While we won't use the IRQ pin we can notice that the MISO pin will act depending on which mode we use (MISO, I2C, or UART). The same goes for the SDA pin which will be used as signal input (SPI) rather than Serial data (I2C) or Serial Data Input (UART).

2.10 Servo Motor

A servo motor is a motor device that can be controlled by a microcontroller and it can rotate or push parts that can be used in many applications (robotics or cameras). [17] The one we will use, the SG90 micro servo motor, is a small motor that can rotate from 0 to 90 and 180 degrees. To give precise rotation, we use PWM, a method to control digital devices, by producing analog signals, which means that the signal wave can be high or low depending on the timing. Inside the servo, some gears make the outer annulus gear turn. To make the gears turn, a DC motor is used, which is powered and controlled by the microcontroller. The servo motor has 3 pins:

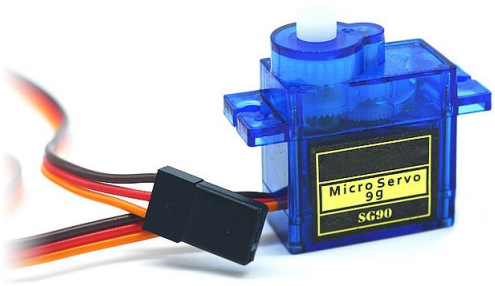


Figure 2.16: A Micro Servo SG90 motor
Source

- **VCC**: 5V pin
- **GND**: Ground Pin
- **PWM**: This pin is controlled by the microcontroller, which controls the pulse.

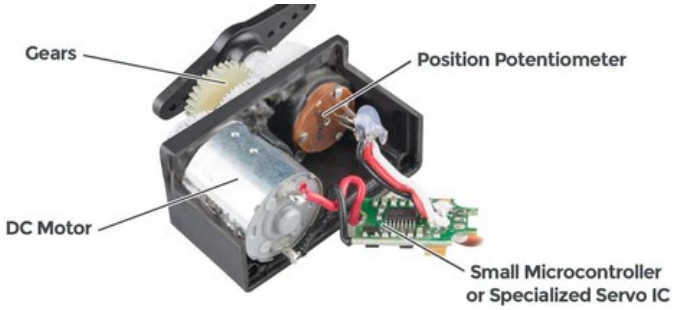


Figure 2.17: Inside an SG90 Servo Motor
Source

We will use the servo motor to provide food as the motor will open and close to drop food onto the pet's plate. [18]

2.11 PIR

The PIR sensor (also known as Pyroelectric or IR motion) is an electronic sensor used to detect movement through IR radiation [19]. The model we are using, HC-SR501 [20], is an inexpensive sensor and an effective way to detect motion from 7 meters and; valuable they can detect human and pet movement as both of them emit an amount of IR radiation (invisible to the human eye course) from their body heat, which the sensor can detect by using a pyroelectric sensor to detect sudden heat energy change in the environment. For that sensor to work, it must adjust to the temperature it is deployed. When a human or pet moves inside its range, the pyroelectric sensor generates an electrical current, becoming a digital output signal.

To increase the activity and effectiveness of the sensor, we use a type of lens known as the Fresnel Lenses.

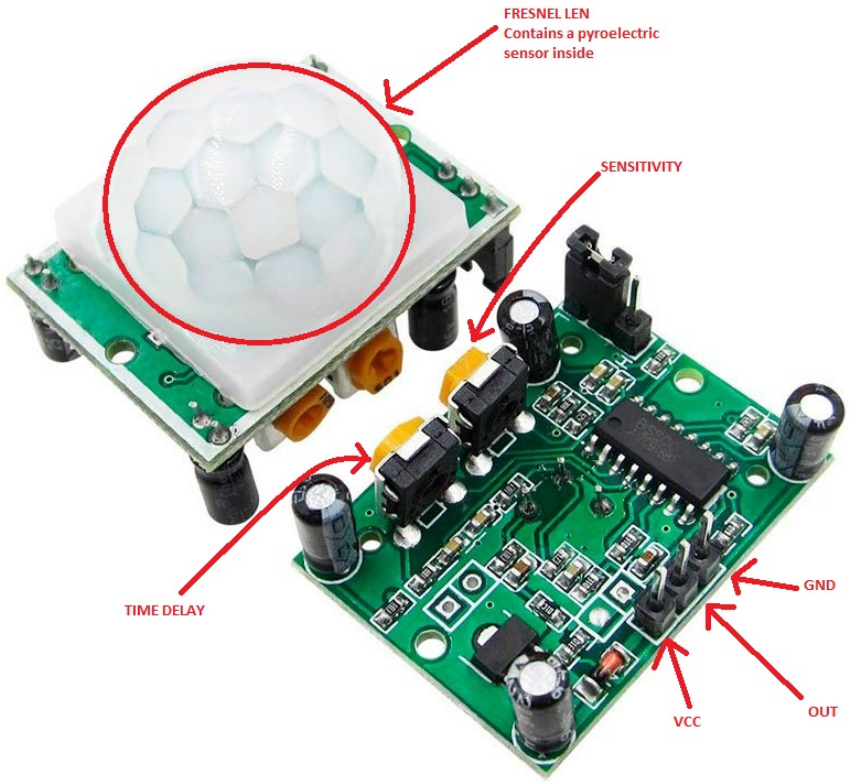


Figure 2.18: An HC-SR501 PIR sensor
Source

2.11.1 Fresnel Lens

Fresnel lens, which was developed for lighthouses, is a compact lens that effectively reflects the lighthouse's light horizontally and vertically. [21]
This invention, made by the French physicist Augustin-Jean Fresnel was one of the giant leaps in lighthouse lighting technology, as many ships managed to float safely from danger. While a lighthouse could be built with only an ordinary glass lens, that would be impossible since it would require lots of resources, which would not make it compelling enough.

Fresnel lens is a unique type of lens that consist of a series of concentric grooves etched into the plastic. These grooves act as individual refracting surfaces that bend parallel rays and merge in a parallel beam that can travel a long distance. [17] Fresnel lenses are used in the following ways [22]:

- **Lighthouses**
- **Searchlights**
- **Floodlights**
- **Traffic signals**
- **Decorative lights in buildings**

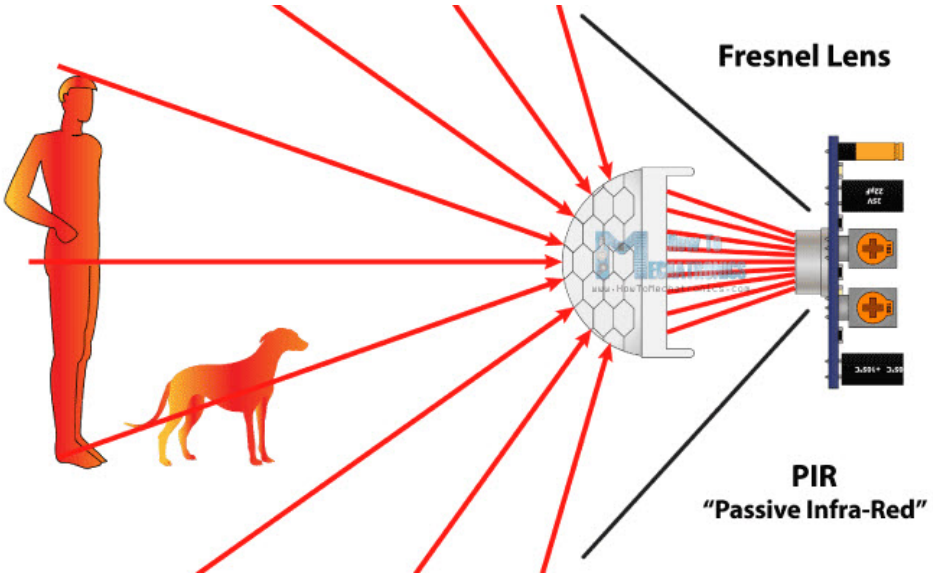


Figure 2.19: A PIR sensor detects IR radiation from body heat
Source

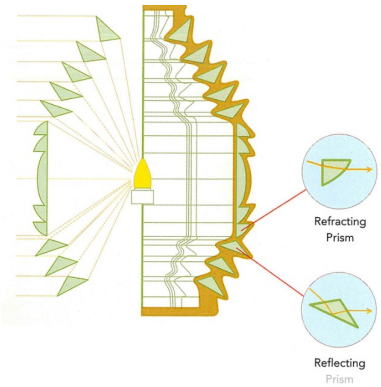


Figure 2.20: Thanks to prisms the light can be reflected in a desired location
Source

- **Cameras & Small projectors**

In PIR sensors, Fresnel Lens is made of plastic that covers the pyroelectric sensor, increasing the effectiveness and sensitivity of the sensor, which can see through that plastic and used as an IR filter.

2.11.2 Inside a PIR sensor

The PIR sensor has 3-pin which are:

- **VCC**: Voltage input from 4.5 to 20 Volt
- **OUT**: This is a 3.3V logic output. In a logic output, we have two results:
  - **LOW**: No movement
  - **HIGH**: Movement detected

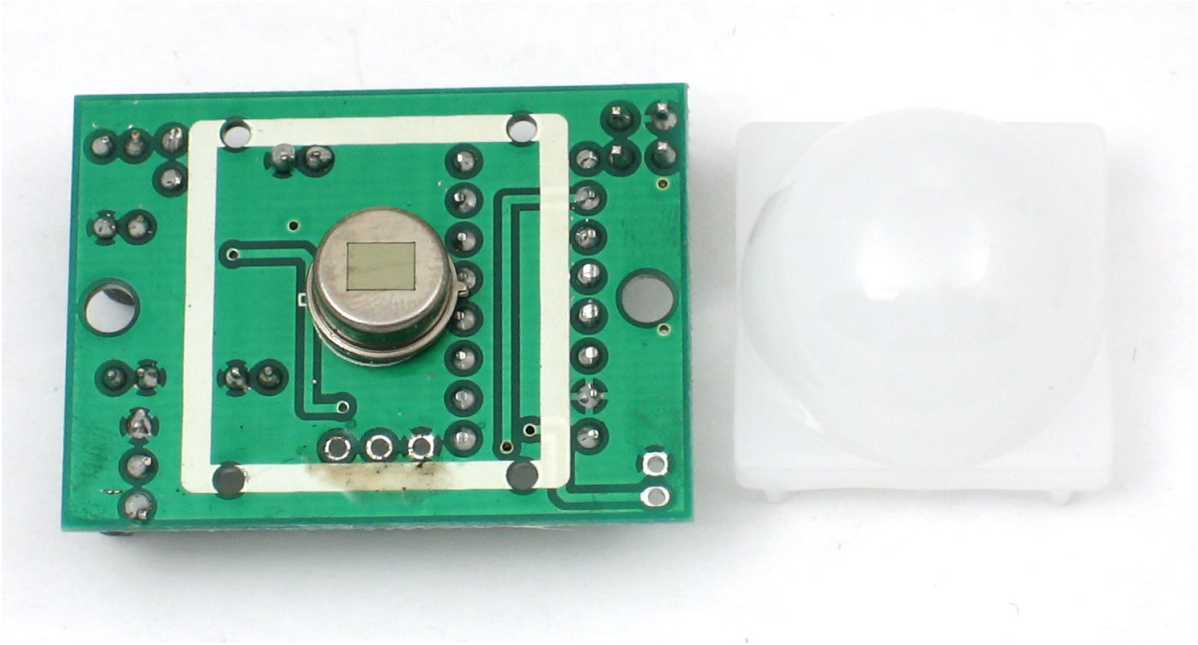


Figure 2.21: The Pyroelectric sensor inside the PIR that detects IR radiation

- **GND**: Ground connection.

Moreover, the sensor uses two potentiometers to adjust two parameters:

1. **Sensitivity**: The maximum distance at which the sensor will detect movement (ranging from 3-7 meters)
2. **Time**: It sets how long the sensor will remain in a HIGH state. The minimum is 3 seconds, and the maximum is 300 seconds (5 minutes).

For our project, we will use two PIR sensors ... both will be deployed in a wall next to the door, with one set on a high position of the wall and the other on a lower [23]. The reason is that when the higher PIR detects motion, it will mean that a human is passing despite whether the lower is triggered. However, when only the lower is triggered, our pet is in range of that sensor.

2.11.3 Ultrasonic sensor

Another alternative for detecting movement is using an Ultrasonic sensor. While this sensor does not detect movement, it detects the closest objective in front of it and measures the distance between itself and the object. An HC-SR04 is an excellent suggestion as it is low-powered and easy to interface. This sensor uses two ultrasonic transducers; one acts as a transmitter and the other as a receiver.

The transmitter converts the electrical signal to ultrasonic pulses, which the receiver gets and produces an output pulse which translates to the distance of the object or person in front of it [24]. Like the PIR, the Ultrasonic sensor is powered up to 5V and has four pins:

- **VCC**: 5V output of our microcontroller

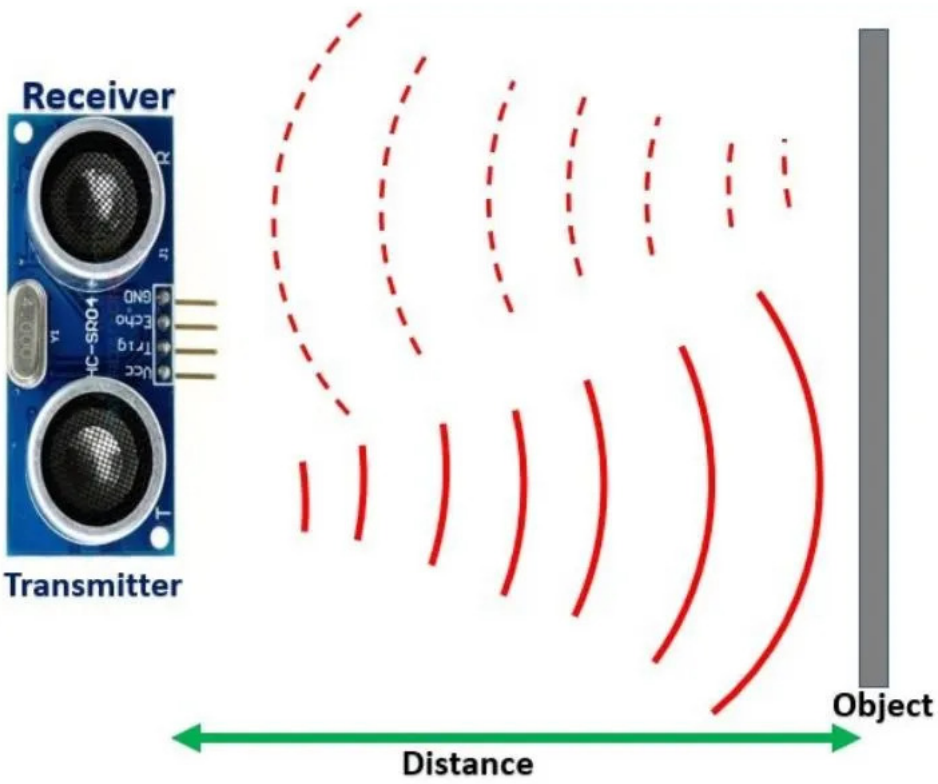


Figure 2.22: An ultrasonic sensor sends waves with the transmitter that reflects back to the receiver, measuring the distance

Source

- **Trig**: This pin triggers ultrasonic sound pulses, and when set HIGH, it initiates an ultrasonic burst.
- **Echo**: This pin remains high when the sensor transmits the ultrasonic burst, and when it receives an echo, it goes LOW. The moment the sensor is HIGH, it calculates the time until it goes LOW, and as a result, it calculates the distance using this mathematical function:

$$Distance = Speed \times Time \tag{2.1}$$

- **GND**: Ground Pin

2.11.4 PIR vs Ultrasonic

Both sensors are valuable and practical when it comes to motion detection despite their different functions to do. However, PIR sensors are preferable for general detection as ultrasonic can detect the object's exact location near it. Moreover, regarding security, security professionals consider ultrasonic sensors low-security devices. An ultrasonic sensor can easily be tricked using materials to absorb the ultrasonic pulses, such as bed sheets or animal fur [25]. Since our project uses motion detection to notice if our pet is near the door, we prefer to use PIR sensors as they can detect sudden temperature changes in front of the sensor. Lastly, the PIR sensor has a larger field of view and can be deployed in house areas where the Supersonic sensor is inferior.

2.12 Water Level Module

This module will be used inside the water container to measure how much water there is inside the container. This module has ten parallel exposed copper traces, five of which are power traces, and the rest are sense traces. This module uses both types of traces as a resistor to mention how much water there is. The more the module and traces get deeper in the water, the more the resistance decreases, and the less the sensor is immersed in water, the more the resistance increases [26]. This sensor generates a digital

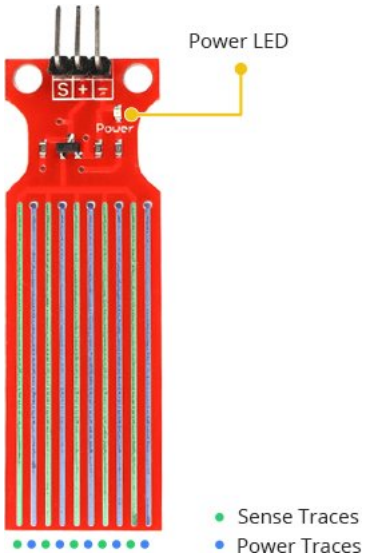


Figure 2.23: A water level module uses copper traces to recognize the level of water in a container
Source

output depending on the resistance, and it is simple to use while it requires three pins:

- **S(Signal)**: It is an analog output pin that will be connected through an analog pin of our ESP32.
- **+VCC**: This pin provides power to the sensor ranging from 3.3-5V; however, the analog output depends on the sensor's voltage.
- **-GND**: Ground Pin

2.13 MQ2

An MQ2 sensor is considered one of the best and most widely used sensors to recognize and detect chemicals in the air. Its reliability makes it useful for indoor air quality monitoring and can also be used for fire detection. The sensor uses a Metal Oxide Semiconductor known as a Chemiresistor. It operates on a 5V voltage supply and can detect the following:

1. **LPG**
2. **Smoke**
3. **Alcohol**

4. **Propane**
5. **Hydrogen**
6. **Methane**
7. **Carbon Monoxide concentrations**

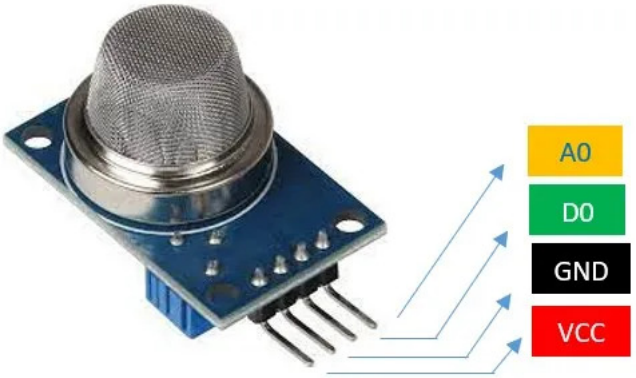


Figure 2.24: An MQ2 gas sensor
Source

While it is impressive that this sensor can detect such a high number, it can not recognize any of the above, so it is preferable to measure the gas density of a specific room. The sensor is covered with two layers of stainless steel mesh, ensuring that the inside element does not cause an explosion since it detects flammable gasses. This mesh, known as an anti-explosion network, protects the sensor, filters out unnecessary particles, and allows only gas to pass through. Inside the mesh, we can see six connecting legs and a sensing element forming a star-shaped structure on the sensor's base. Two of the six legs are connected to a Nickel-Chromium coil inside an Aluminum Oxide Based Ceramic to keep the sensor heated all the time.

Meanwhile, the remaining four legs connect with a Tin Dioxide coating, which is sensitive to combustible gasses, and as a result, these elements are responsible for sensing gas elements. The sensor's analog

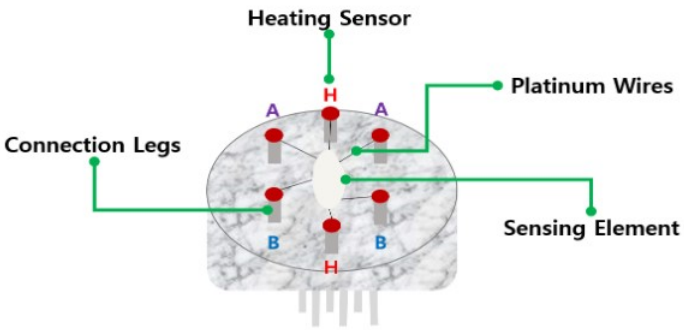


Figure 2.25: Inside an MQ2 sensor
Source

output changes depending on whether there is gas concentration. When the air is clean, the electrons

from the tin dioxide are attracted to oxygen molecules outside of the sensor, forming a barrier of Oxygen and electrons. As the electrons are far away from the coating of Tin Dioxide, the latter becomes highly resistive, preventing electric current flow. However, the moment the sensor senses gas near itself, the Oxygen reacts with the gas shrinking its amount, and the electrons return to the coating lowering its resistance levels and allowing the current to flow through the sensor [27].

2.14 Flame Detection Module

Fire alarm systems are widespread to use in smart home systems. If there is a fire in the room, the user can be notified and call the fire department, or the system can notify emergency services if the user cannot. Moreover, as smart home systems can be customized, they can include a sounding alarm or activate a fire suppression system. While we have a sensor to check for gas leaks or flame fumes, the MQ2 sensor cannot detect flames. An IR Flame sensor module is a simple way to achieve that. This module is equipped with an IR photodiode sensitive to IR lights. Flames emit levels of IR light that are not visible to the human eye but can be detected by the IR led. When the IR led recognizes the flame, an Op-Amp checks for a change in the sensor's voltage. The default voltage output is 5V, meaning there is no fire, and when it recognizes the flames, the output voltage will be 0V.

Figure 2.26: Fire Module
Source

2.15 IR Modules

IR technology is a wireless means of communication to control devices. In a household, many devices use IR technologies such as TVs or A/Cs, and to control them, we need a remote control with an IR led that sends a signal to the device, which corresponds depending on the signal received. Nevertheless, we need the remote controller in the same area as the device we want to control. Using IR modules, we will control air conditioning without needing to be in the same room or using a remote controller. To understand how we will use IR and A/C, we must explain how it works.

2.15.1 A/C with Remote control

As mentioned above, the remote control has an IR led known as a transmitter, which sends pulses of IR light (which flashes on and off thousands of times per second) translated as binary codes. These

binary codes have a specific function (Power ON/OFF, for example) and are received by the IR (receiver) inside the device we want to control. As the receiver gets the binary code, the device's microprocessor understands the command, which the remote control requests and the device executes [28].

2.15.2 IR Receiver module

Figure 2.27: An IR Receiver module
Source

To use an IR transmitter to send the signals to the device we want to, we need to know the exact signals per function; also, it is crucial to learn which IR protocol is used to make it easier to decode the signals. We connect the IR Receiver module with the ESP 32, and we use the library ***IRremoteESP8266*** and specifically the example code ***IRrecvDumpV2***. To connect the IR receiver, we need to connect three pins:

1. **GND**: Ground pin
2. **VCC**: 5V
3. **OUT**: The output we will authorize to a pin of ESP 32.

```
21:26:38.261 -> Timestamp : 000020.705
21:26:38.261 -> Library   : v2.8.4
21:26:38.261 ->
21:26:38.261 -> Protocol  : GREE
21:26:38.261 -> Code      : 0x4C0B205001200038 (64 Bits)
21:26:38.261 -> Mesg Desc.: Model: 2 (YBOFB), Power: On, Mode: 4 (Heat), Temp: 27C,
21:26:38.267 -> uint16_t rawData[139] = {9004, 4470,  670, 538,  648, 556,  644, 166
21:26:38.341 -> uint8_t state[8] = {0x4C, 0x0B, 0x20, 0x50, 0x01, 0x20, 0x00, 0x38};
```

Figure 2.28: Using the remote control of the A/C we figure out that it uses GREE protocol

After we upload the example code to our ESP 32, we use the remote control of the A/C and press the power on button while aiming at the receiver. The receiver will recognize the protocol being used and extract the binary signal.

2.15.3 IR Transmitter module

After we get the IR protocol, we can use an IR transmitter to send commands to our A/C. This module, made by DFROBOT, contains a Light emitting diode, which generates an IR light with a 1mm to 750 nm wavelength. [29]

For this module to work, it needs to be in the line of sight and at a proper distance from the A/C. As the A/C receives the signal with its own IR receiver, it converts the beam of light signal to an electrical one, and thanks to the integrated circuit of the A/C, it can execute the command it was given. The module sends a digital wave signal to 38 kHz frequency and requires a 5V power supply. This module has three pins:

- **VCC**: 5V power supply
- **GND**: Ground
- **D**: Digital input.

Figure 2.29: IR transmitter module made by DFROBOT
Source

Chapter 3: Explaining The System

Now that we have explained everything related to the hardware and the software we are using, we will see how all of those components work together and how they connect.

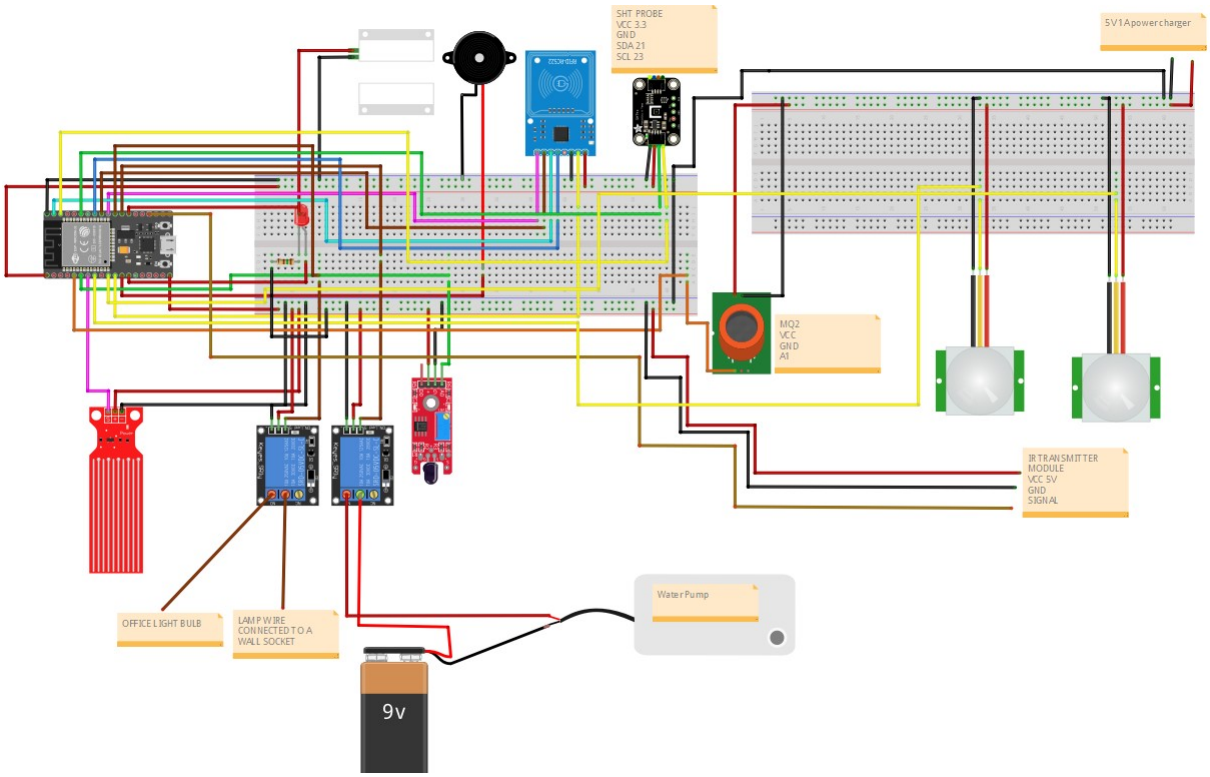


Figure 3.1: The complete electric circuit

3.1 Code explanation

Here we will explain how the ESP32 is connecting. After including all the libraries and initializing all the variables and pins we will use in our ESP32, we set up the Wi-Fi and the IP address of the MQTT broker to which the ESP32 will connect. After ESP32 is connected to the Wi-Fi, it will subscribe to the topics and need to receive control orders from Node-RED. Afterward, the loop function will be run with the sensors, or the state of the devices will be sent to Node-RED, which the user can view via the dashboard.

3.2 ESP 32 Connections

It is crucial to mention that ESP 32 and all its sensors will be deployed on a breadboard. While the particular ESP 32 model we have chosen(**DEVKIT 1**[3]), has the potential to power 3.3 and 5V devices, more is needed, as most devices powered up by 5V can interfere with the other devices, and give us false information. In one example, when we use the relay module to open our office lamp, the PIR sensor to detect the pet will give HIGH output. For this issue, we can use another breadboard that a 5V 1A charger can power up, and while the devices' digital and analog pins are connected to the ESP-32, they are powered up by the 5V charger. The exact pin mapping can be seen in table 3.1.

[3]Development Kit

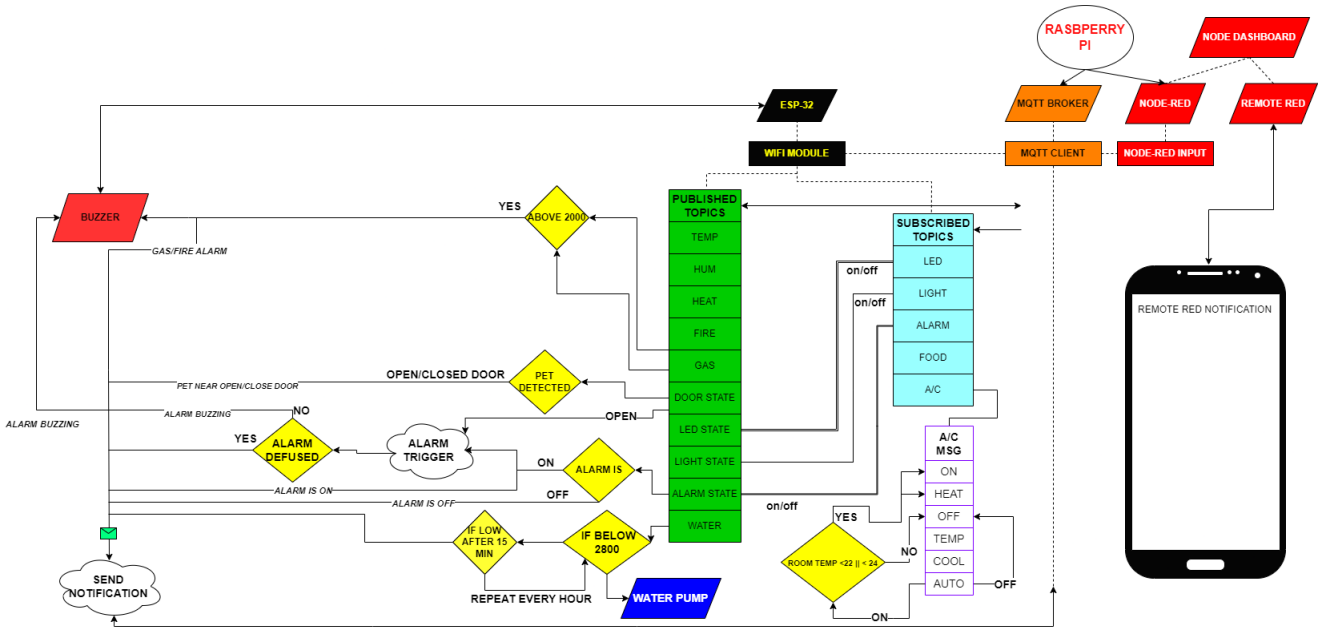


Figure 3.2: Caption

3.3 Node Red & ESP 32 Communication

As mentioned above, Node-RED and MQTT are installed in the Raspberry Pi. While both software could be installed on a personal computer, it is preferable to be used by a device (the Raspberry pi) that can run efficiently with minimal use of resources, as both Node-Red and MQTT will run automatically the moment Raspberry Pi boots up. After that disclaimer, let us see how Node-RED MQTT and ESP 32 will communicate with each other. In our project, both clients (ESP 32 and Node-RED) receive data from each other by using the Mosquitto MQTT broker to establish communication with each other.

3.3.1 ESP 32 Topics

ESP-32 uses sensors or devices to gather information about the environment. It will publish this information via topics through the broker, which will be transferred and displayed on our Node-RED dashboard. The topics that ESP 32 publishes are related to:

1. **Temperature** - SHT40
2. **Humidity** - SHT40
3. **Heat index** - SHT40
4. **Gas** - MQ2
5. **Fire** - Fire Module
6. **Water Level** - Water Level Module
7. **Door State (OPEN/CLOSED)** - Magnetic Switch Lock
8. **Led State (ON/OFF)**

3.3V	VCC1
5V	VCC2
GND	GROUND
GPIO35	FIRE MODULE ANALOG PIN
GPIO34	MQ2 ANALOG PIN
GPIO33	PIR 1 DIGITAL PIN
GPIO32	WATER MODULE SIGNAL PIN
GPIO27	RST RFID PIN
GPIO23	MOSI RFID PIN
GPIO19	MISO RFID PIN
GPIO18	SCK RFID PIN
GPIO25	PIR 2 DIGITAL PIN
GPIO22	SCL SHT40 PIN
GPIO21	SDA SHT40 PIN
GPIO17	OFFICE LAMP RELAY PIN
GPIO16	WATER PUMP RELAY PIN
GPIO15	IR TRANSMITTER SIGNAL PIN
GPIO14	BUZZER (+) PIN
GPIO12	LED (+) PIN
GPIO5	SS RFID PIN
GPIO4	MAGNETIC LOCK SWITCH

Table 3.1: ESP 32 Pin Mapping

9. **OFFICE LAMP STATE (ON/OFF) -** RELAY MODULE
10. **Alarm state (Armed/Unarmed)**

As we notice, we publish this information to the dashboard, and we can also use publish to receive notifications through our smartphone, which we will see later.
Since, in our system, some devices can be controlled remotely with the dashboard or automatically, the ESP-32 needs to subscribe to the related topics for those devices to be controlled by the user or be used automatically. These subscripted topics that allow us to control devices remotely are related to the:

1. **LED**
2. **Office lamp**
3. **Alarm**
4. **Food For Pet -** Servo motor
5. **A/C (ON/OFF) -** IR Transmitter
6. **A/C TEMPERATURE (20-27 Celsius) - IR Transmitter**
7. **A/C HEAT MODE -** IR Transmitter
8. **A/C COOL MODE -** IR Transmitter
9. **A/C AUTO MODE -** IR Transmitter

3.3.2 Node RED Nodes

As mentioned in chapter 2, Node-RED uses nodes and payload messages to receive information. However, we will use different types of nodes connected via flows to build our dashboard. The nodes we are using are :

- **MQTT IN**: They are used when they receive data from ESP-32 to view the sensors' data or confirm the state of a device.

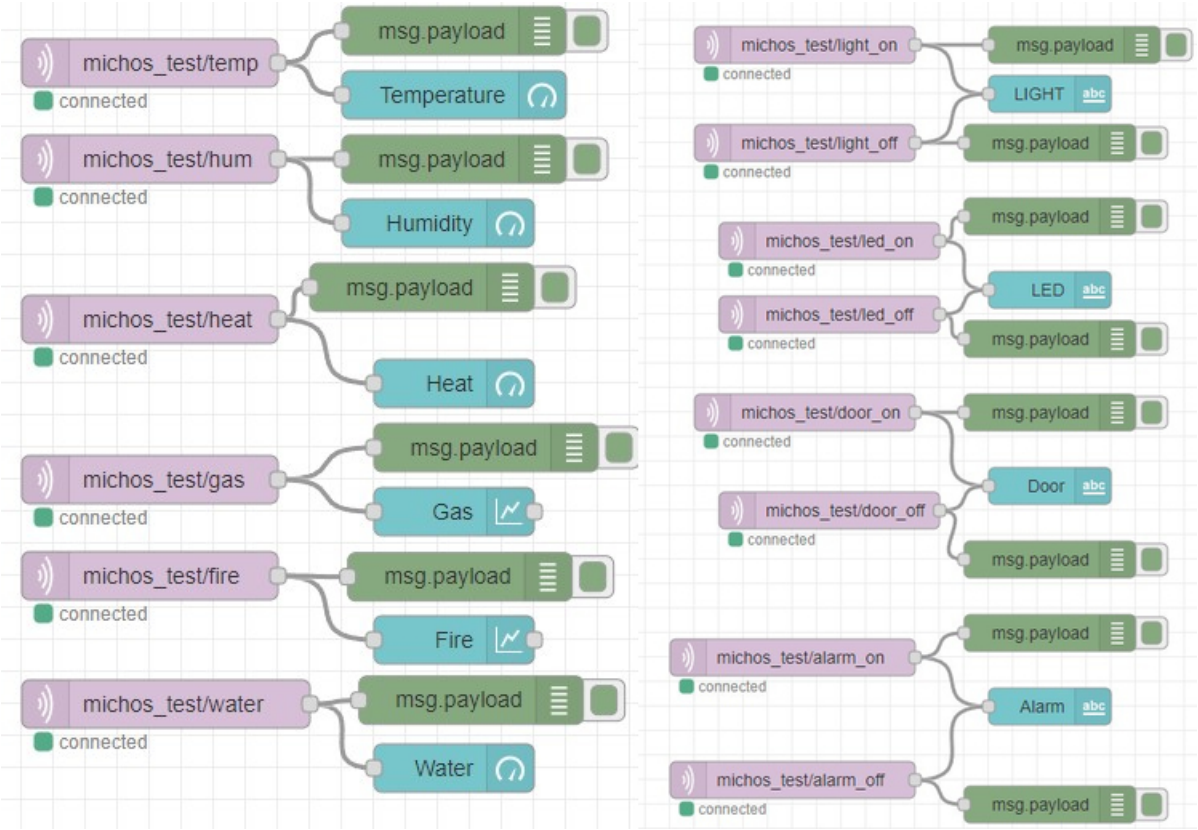


Figure 3.3: We use MQTT IN nodes to view the data and confirm the device's state.

- **MQTT OUT**: Used to sent commands to our ESP-32 devices.
- **Message**: Simple JavaScript objects with any property set. Since we will use their default behavior, they will act as payload messages, meaning they will contain the message they receive. Message nodes are used only in MQTT IN nodes.
- **Dashboard**: The dashboard nodes are divided into two categories:
  1. **Display**: These nodes showcase the data we receive. These nodes are:
     - Text Nodes
     - Gauge Nodes
     - Chart Nodes
  2. **Control**: We can control our devices using these nodes.

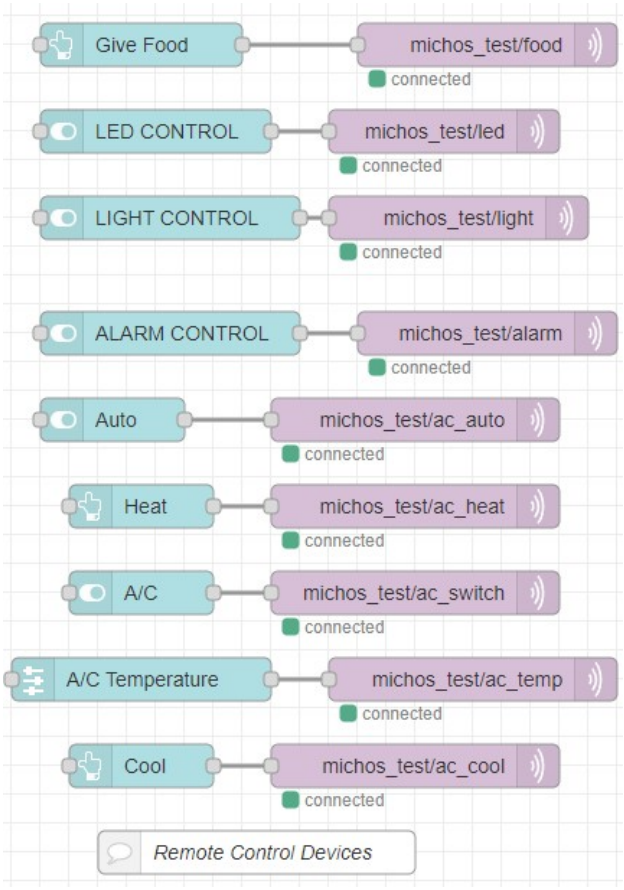


Figure 3.4: We control our devices with MQTT OUT nodes

  - Button nodes
  - Switch nodes
  - Slider nodes

- **Template**: We can apply a few appearance changes to our dashboard using template nodes. For our example, we use this node to show the current date & time with the code in figure 3.3.
- **Notification Nodes**: These nodes are used to receive smartphone notifications.
- **Inject Node**: This node injects a message into a flow. The inject node can transfer a message payload in various types (JSON objects), and the transfer can be done automatically at regular intervals.
- **OpenWeatherMap Node**: This node will get the current weather conditions. If we want a specific message, we need to use a debug node to find the exact information we desire(temperature or weather details).

3.4 OpenWeatherMap API

OpenWeatherMap provides weather data for the user's location in a reliable and fast manner. To access weather data, the user must create an account and get an API key on the dashboard. Moreover, we can choose the location we want for the weather forecast. We can read the whole object using a debug message

```
<script id="titleScript" type="text/javascript">

$(function() {
    if($('.md-toolbar-tools').length != 0){
        loadClock();
    }else setTimeout(loadClock, 500)
});

function loadClock(){
    $('#clock').remove();
    var toolbar = $('.md-toolbar-tools');

    var div = $('<div/>');
    var p = $('<p/ id="clock">');

    div.append(p);
    div[0].style.margin = '5px 5px 5px auto';
    toolbar.append(div);

    function displayTitle(lh) {
        p.text(lh);
    }

    function upTime() {
        var d = new Date();
        p.text(d.toLocaleString());
    }

    if(document.clockInterval){
            clearInterval(document.clockInterval);
            document.clockInterval = null;
    }

    document.clockInterval = setInterval(upTime,1000);
}

</script>
```

Figure 3.5: With this Javascript code, we can have time and date on the top right of the dashboard

to acquire the exact information we need (weather details or temperature). Using the debug messages, we can acquire the ”addresses” in which our information is placed and use these ”addresses” in the value format of each node. [30]

3.4.1 API KEY

We need authorization from the user to use an API on our dashboard. An API key is used for authenticating requests for a specific service. While the API key differs per user, it cannot identify its user. [31]

3.4.2 How to use it in Node-RED

Usually, to use the OpenWeather API, we would need an HTTP request node with a GET Method. Furthermore, to gain information about our city, the URL form would like something like this:

https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}&{units}

The units parameter is used to change the temperature unit of measurement. The unit measurement will

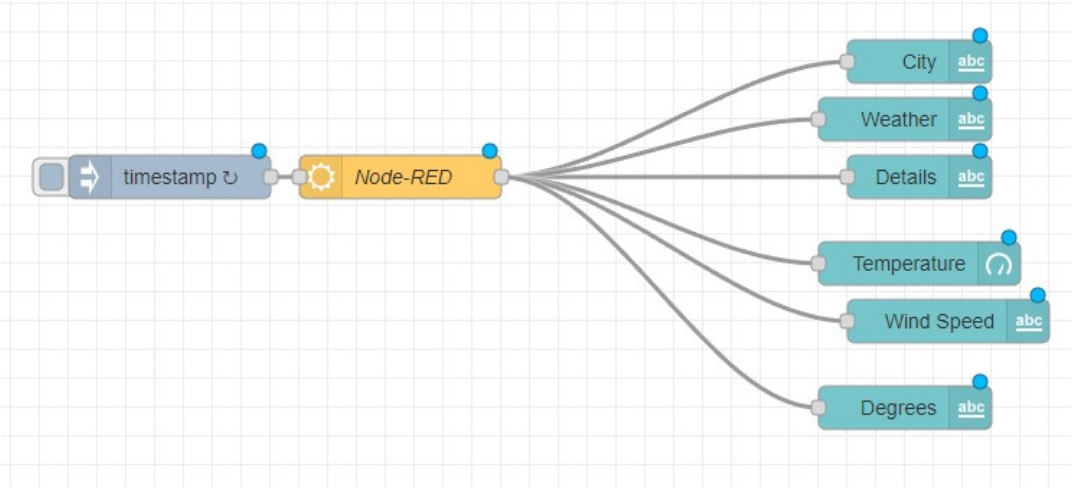


Figure 3.6: These nodes are used to extract information from the OpenWeatherMap API

be applied by default (Fahrenheit) if not used. However, since we want the temperature unit in Celsius, the units parameter will be equal to metric. So, for example, in case we want to learn about the current weather of Thessaloniki in Celsius, and we have an API key which (for example purposes) is: ***a983nf*** the URL form will be similar to:

https://api.openweathermap.org/data/2.5/weather?q=Thessaloniki&appid=a983nf&metric

Thanks to Node-RED, we can find nodes that automatically do this for us; by using the User Settings Palette, we can find the OpenWeatherMap node library and install it in our workplace. Afterwards, in the OpenWeatherMap node, we can configure which City and Country we want while adding the API key.

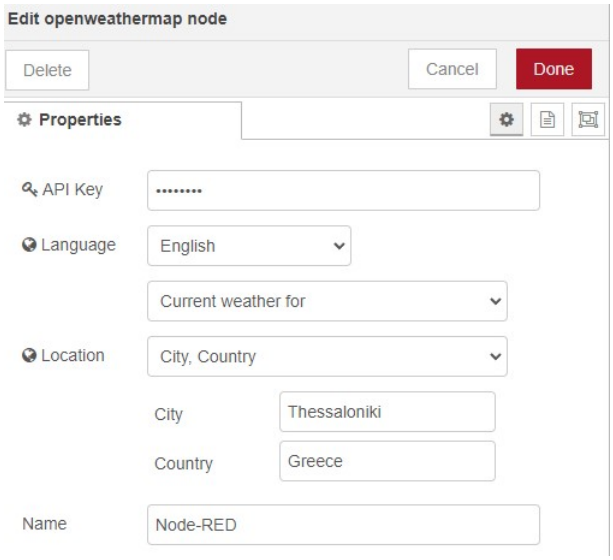


Figure 3.7: The configured settings for the OpenWeatherMap node

3.5 Remote RED

Thanks to the Remote RED app we install on our smartphone and adding our dashboard as an instance, we can view our dashboard and control our devices remotely without the need to be in the same room or use the same network. Moreover, Remote RED allows us to receive notifications on our smartphones. In order to do so in our programming code, we need to publish the topic to Node-RED. Also, we connect **MQTT IN** nodes with the notifications nodes, from which we can configure the name and details of the notifications and, if we want, the sound to distinguish the notifications. The notifications will be sent depending on various conditions set in our code.

Figure 3.8: As we open the Remote RED application, we can see all the instances we have added in order to control our dashboard remotely

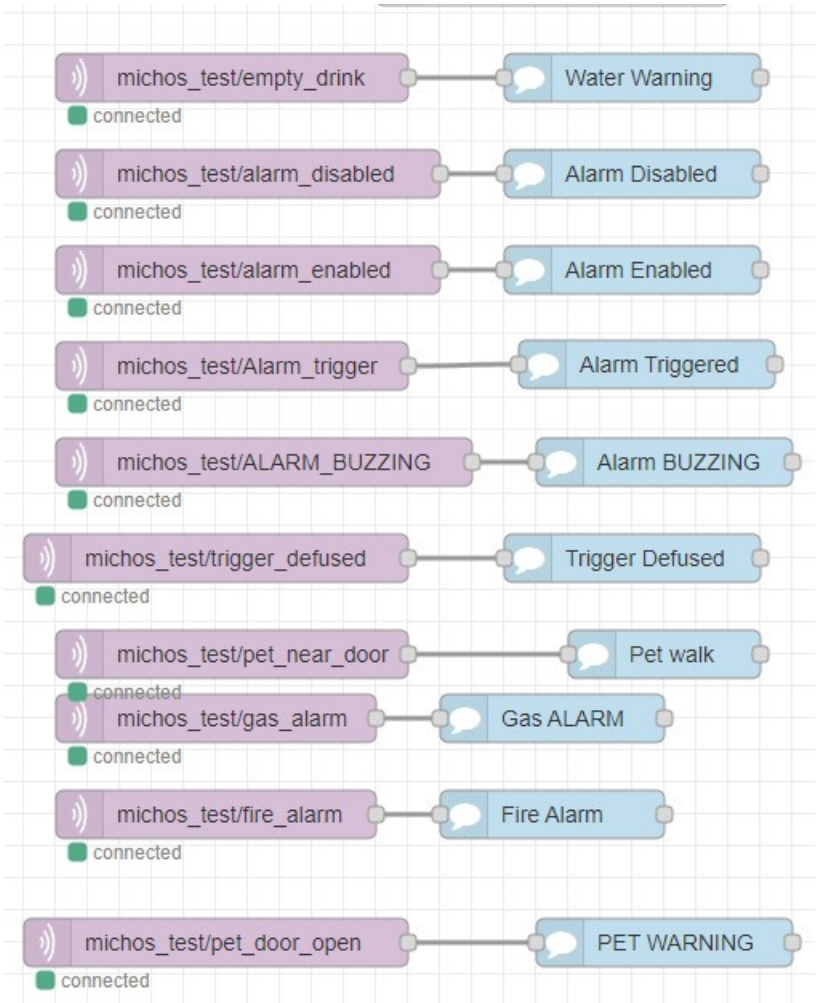


Figure 3.9: The MQTT IN nodes are connected with Notification nodes

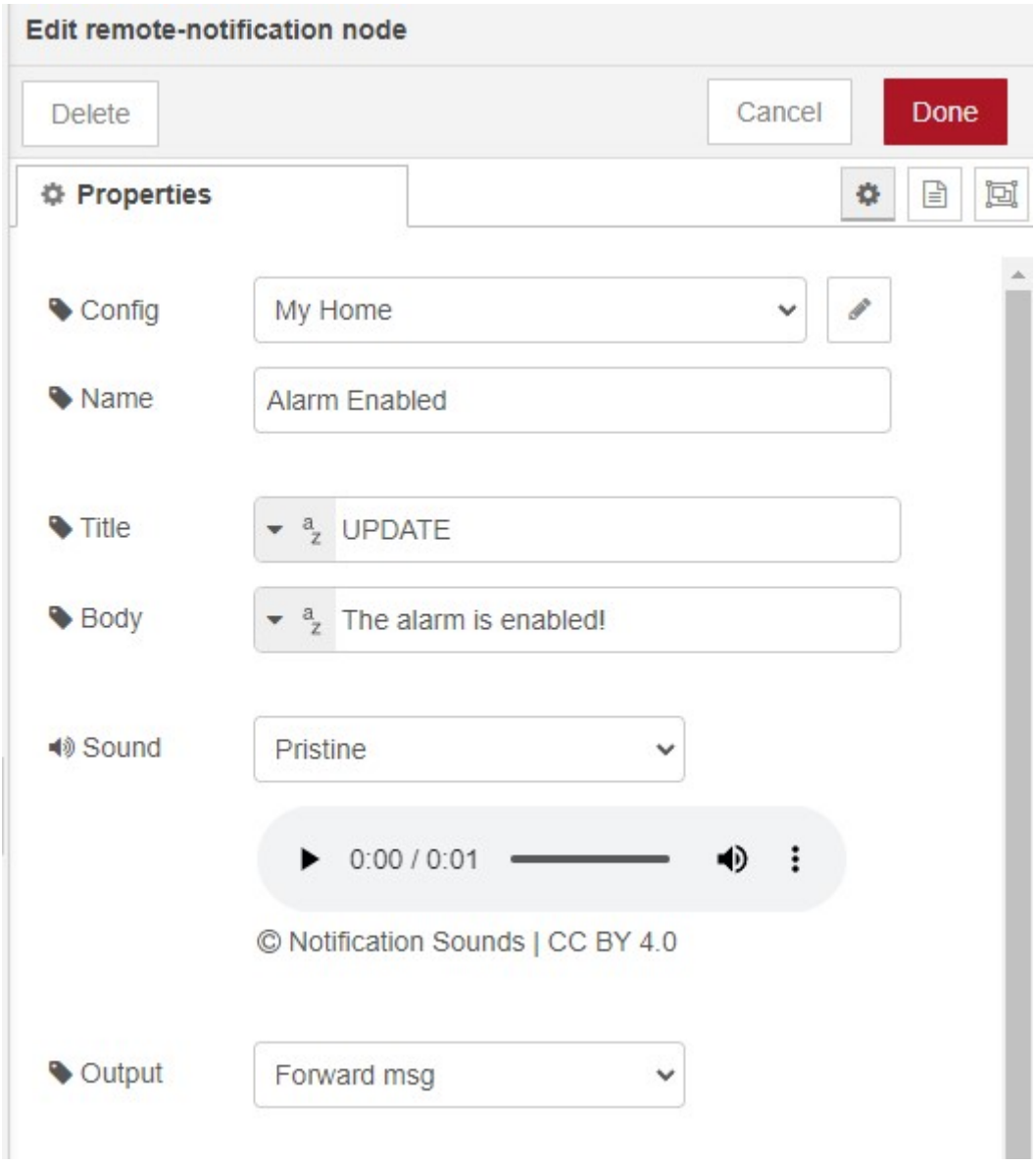


Figure 3.10: The configure options on notification nodes

Chapter 4: Node RED Dashboard

The Node-RED Dashboard is a UI that is used to show the nodes that are used. The nodes and the data can be seen in real-time thanks to Javascript. As we will see we use, the Dashboard will have four different layouts so the user can navigate between them thanks to a sidebar menu. While explaining the purpose of each layout, we will also explain each measurement and control method of our sensors and devices.

4.1 Main Page

The main page is the first one the user sees when accessing the dashboard. This page focuses on the home conditions of the house, and we introduce three different groups to change or get updated on the conditions.

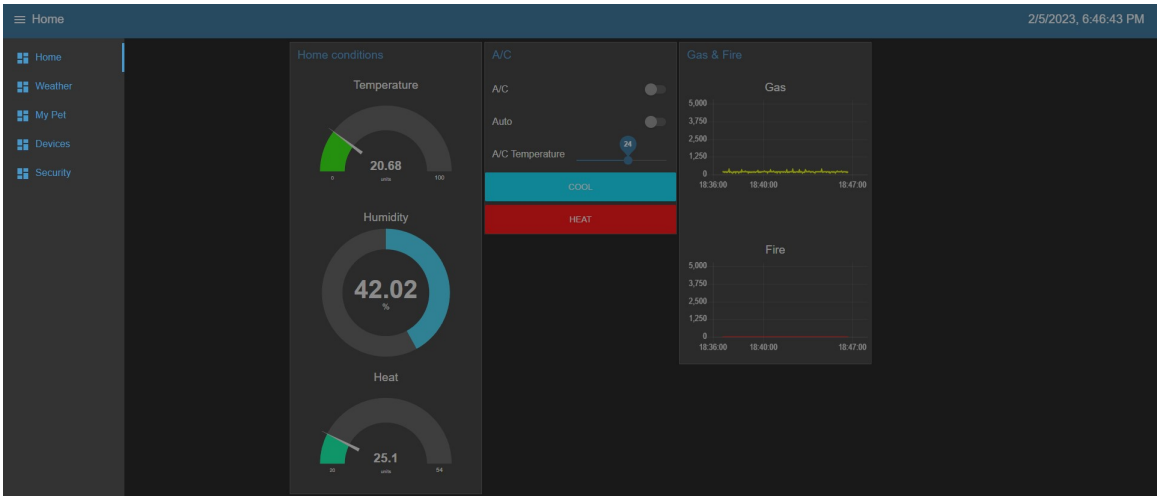


Figure 4.1: The main page and the menu

This group is divided into 3 parts:

4.1.1 Home Conditions

In that group, we can see three measurements in the form of gauges :

1. **Temperature**: Using the SHT-40 probe, we measure the room temperature where the probe is deployed. We can adjust the Auto mode of the A/C depending on the temperature to have the proper condition for the user and the pet. The ideal temperatures of the user's house depend on various variables [32] such as:
    - Temperature outside
    - Year's season (Winter 20-22 Celsius, Summer 18-20)
    - Time of the day. At night times, the human body is more comfortable in cooler conditions.
    - The age of the user as elders are more vulnerable to cold conditions

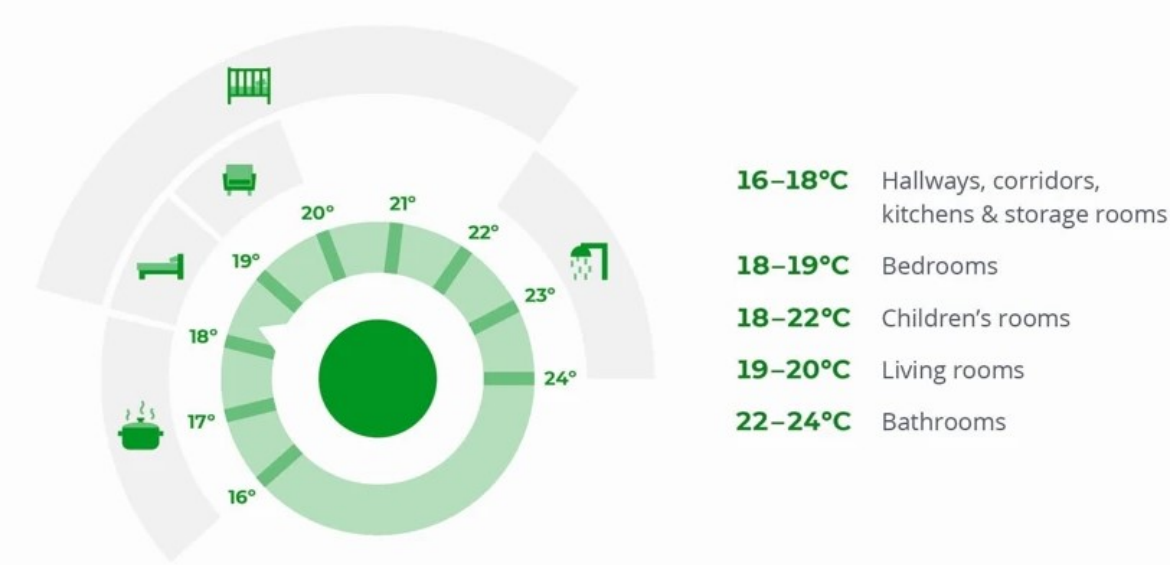


Figure 4.2: Depending on the room the ideal temperature differs from others

- Which room is heated
- If the user has a pet [33] and how old it is (Puppies and kittens need constant warmth). Moreover, the room temperature range differs depending on the pet type.
  - **Dogs**: 20-22 Celsius in Winter and lower temperatures in Summer. Dogs with thick fur need to stay cooler in Summer, so the temperature should not pass 27 Celsius [34].
  - **Cats**: Unlike dogs, cats can endure higher temperatures ranging from 30-32 Celsius. On the other hand, most cats cannot endure temperatures lower than 21 Celsius.

To display temperature, we use a gauge that is divided into three levels:

- **0-20 Celsius**: Cold conditions
- **20-27 Celsius**: Room conditions
- **27-40 Celsius**: Hot conditions

2. **Humidity**: Like temperature, humidity is measured by SHT-40. A house needs to have balanced humidity levels from 40-60% [35]. Low levels increase the chances of the user being compromised of cold conditions and infections, which leads to itchy or dry skin. On the other hand, if the levels are too high, the house can have moisture problems leading to worse symptoms for those with asthma or allergies and even mold spreading through the house. Depending on the levels of humidity, we have three sectors which are:

   - **0-40%**: Too dry
   - **40-60%**: Perfect room conditions
   - **60-100%** Too much moisture

3. **Heat Index**: The heat index [36] is a variable about what the temperature feels on the human body when relative humidity[4] is combined with air temperature. This is an essential aspect since this

[4]Relative Humidity measures the current moisture levels in the air out of how moist the air could be

takes into consideration the comfort of the human body. When the humidity is high enough, the water in the air cannot evaporate fast enough, and the human body feels that the room temperature is much hotter than it is. We temporarily convert the temperature from Celsius to Fahrenheit to calculate the heat index in our code. Given that Celsius is C and Fahrenheit is F, the unit conversion is calculated by this equation:

$$F = (C \times 1.8) + 32 \tag{4.1}$$

Classification	Heat Index	Effect on the body
Caution	80°F - 90°F	Fatigue possible with prolonged exposure and/or physical activity
Extreme Caution	90°F - 103°F	Heat stroke, heat cramps, or heat exhaustion possible with prolonged exposure and/or physical activity
Danger	103°F - 124°F	Heat cramps or heat exhaustion likely, and heat stroke possible with prolonged exposure and/or physical activity
Extreme Danger	125°F or higher	Heat stroke highly likely

Figure 4.3: The heat index can have different body effects, depending on its range
Source

Now that temperature is converted to Fahrenheit, we need to calculate Heat index(H) by using Humidity which will be presented as R (Relative Humidity), the given equation gives a close measurement of the heat index, but it was given through multiple regression analysis[5] with an error of ±[1.3%]:

$$H = -42.379 + 2.04901523F + 10.14333127R - 0.22475541FR - 0.00683783T^2 - 0.05481717R^2$$

$$+0.00122874F^2R + 0.00085282FR^2 - 0.00000199F^2R^2 \tag{4.2}$$

This equation gives us the heat index inside the house, but the equation will be converted back to Celsius to be presented in the dashboard. To do that, the conversion from Fahrenheit to Celsius is:

$$C = (F - 32) \times 0.5556 \tag{4.3}$$

With that in mind, the heat index in our dashboard is divided into three groups:

- **20-26 Celsius**:Not heat threat
- **26-41 Celsius**: Alarming heat threat
- **41-54 Celsius**: Extreme heat threat

4.1.2 A/C Controls

While A/C could be added to the devices, it would be more comfortable if the user could see the temperature conditions in his home and control the A/C without switching layouts. The first toggle switch opens and closes the A/C, and using the slider, we can control the temperature of the A/C. Moreover, the Cool and Heat buttons will change the A/C's mode. Lastly, when the auto switch is on, it will check the

[5] A set of statistical processes for estimating which variable has an impact on a "dependent" variable

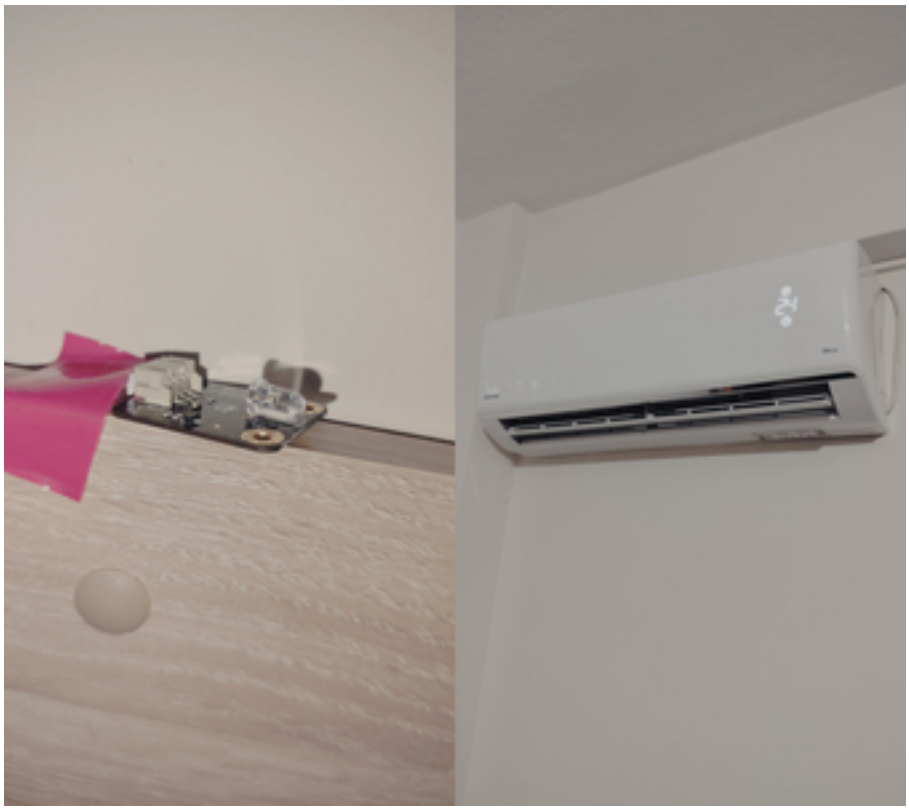


Figure 4.4: Using the IR transmitter module (left) we can control the A/C (right)

room's temperature, and if it is too low (below 22 Celsius), the A/C will turn on and close automatically when the temperature reaches 24 Celsius.

4.1.3 Gas & Fire

In this group, we gain information from the MQ2 and the Fire module sensors to check for gas leaks or fire hazards. Both of them can be viewed in the form of separate graphs. While we can configure the graphs to show us the last data in a past period, we have chosen to show us the measurement from the last 10 minutes. If one of those measurements gets too high, the buzzer will sound, and a notification will be sent to the user's smartphone, alerting him about the high measurements in one of those graphs.

4.2 Weather

In this layout, we can see the weather conditions outside the house. Weather information uploads when our system boots and the user can get hourly updates. Using weather conditions, the user can know how cold or hot it is outside and if it is raining so the user can take out his pet for a walk. While the user can see the temperature (in celsius) and how is the weather, the user can also see how windy it is outside.

4.3 Pet

In this layout, the user can provide food by pressing the **"GIVE FOOD"**, which will make the servo rotate to make the pet's food drop into the food bowl, and then it will close. Next to the button is a gauge

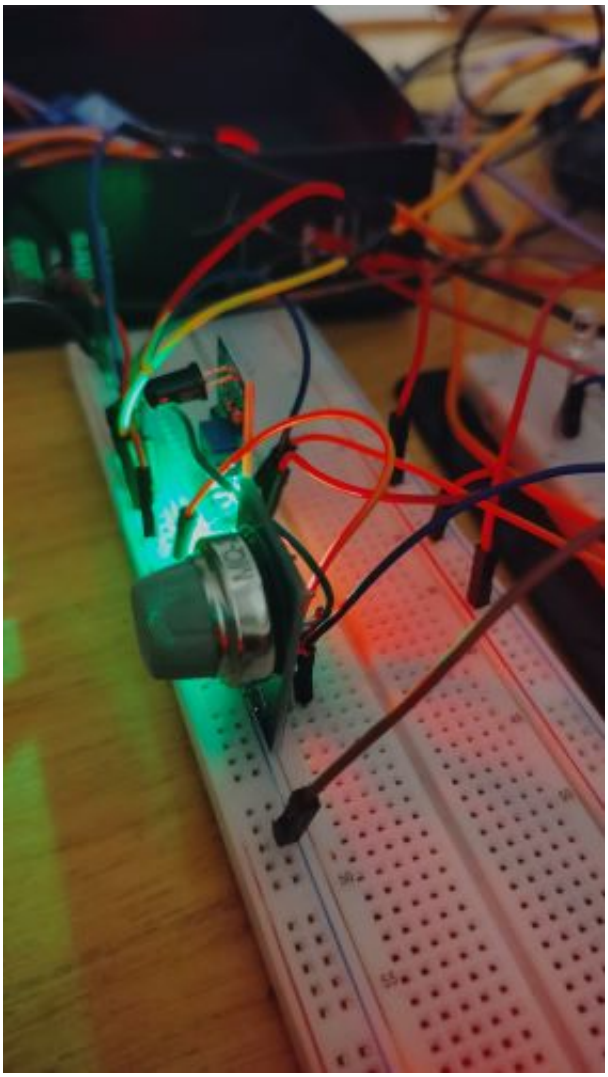


Figure 4.5: With the MQ2 and Fire module sensor we surveillance for fire and gas leaks

showing if there is enough water for the pet. If the water levels get low, the water pump will start filling the pet's water bowl for 5 seconds, and it will close. However, there is a case where the water reservoir will be empty, so the pump could be activated continuously without providing water. Since we do not want the pump to run until its battery runs out, the pump will be activated for five seconds, and then it will stop, starting a timer.

As the timer reaches 15 minutes, a notification will arrive to the user, reminding him to fill the reservoir and the water bowl. From that point, the user will receive hourly reminders to fill both containers, which will be rare since the reservoir can hold 1.5 lt and the water bowl up to 2 lt.

The consumption of water depends on various variables such as [37]:

- **Dog or Cat**: Dogs should drink 28.41ml (1 ounce) per 0.45 kg (1 pound) of their body weight. However, the number could decrease if the dog eats wet food. Conversely, a cat drinks less water than a dog, as adult cats should consume between 142.05-284.1 ml (5-10 ounces) of water. Like

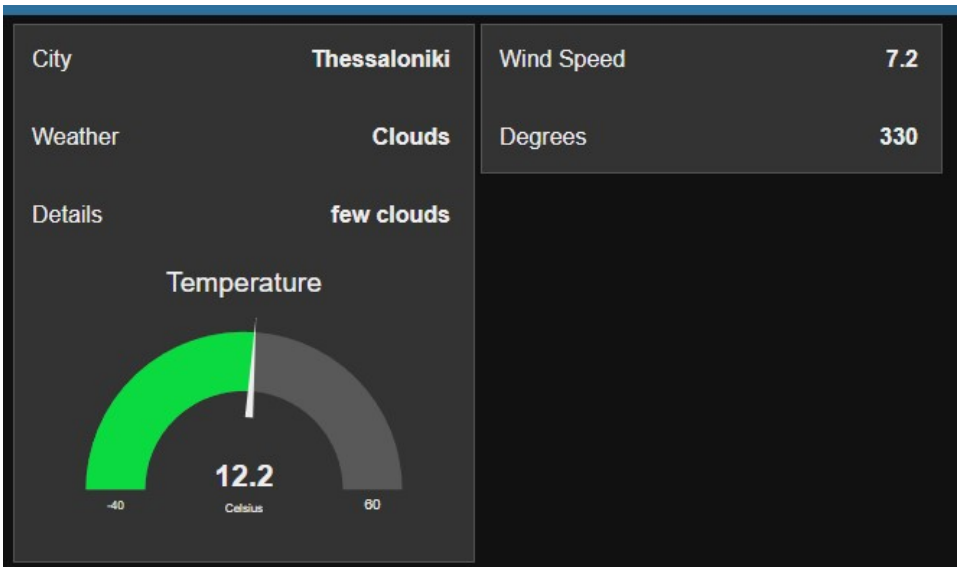


Figure 4.6: A current weather forecast can be viewed in our dashboard

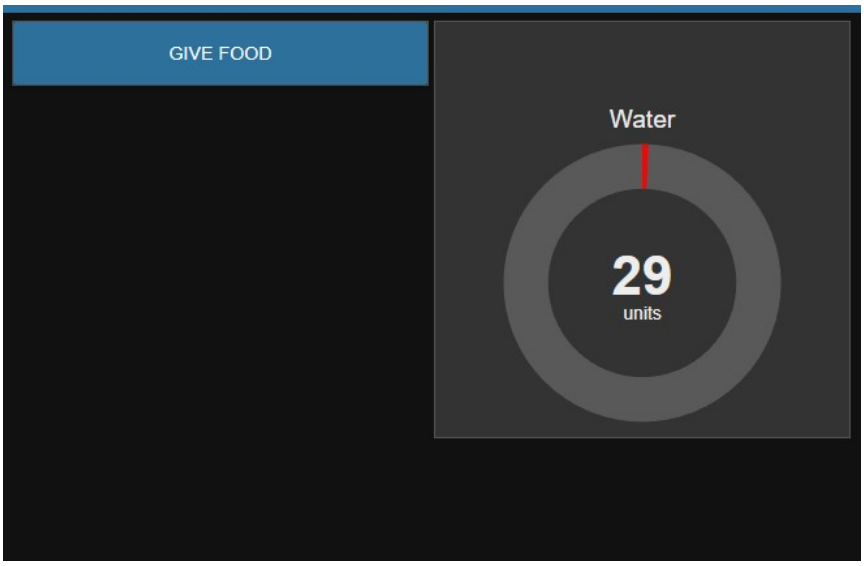


Figure 4.7: The button is providing food and the gauge shows how much water there is in the pet's bowl

dogs, cats' water consumption can change if the cat eats dry food or canned food[6].

- **How old is the pet**: Young dogs or puppies require more water than their adult counterparts. Nevertheless, since puppies will fulfill their thirst from their mother's milk when eating dry food, they will need to rely more on water. According to experts, a puppy needs one-half cup of water every two hours. This theory also applies to cats and kittens with smaller amounts of water. After a cat comes to an age where it will stop depending on its mother's milk, a kitten will need 70 ml of water daily, and at the time it becomes an adult, according to its weight, the amount of water can reach shyly 300 ml.

- **How active it is**: Depending on the dog's activities (mostly walks or playing), a dog will require to drink more water than less active dogs and stay in the house. However, that does not apply mostly

[6]Canned cat food contains high moisture

Figure 4.8: The servo motor is placed in the opening of the container. When we press the button, the servo will rotate the disk, and the food will drop into the small hole

Figure 4.9: Inside the yellow reservoir, the water pump will flow water through the tubes if the water is low thanks to the water level module

to cats since they stay mainly in the house.

- **How hot is the climate**: A dog or cat that lives in a hot climate will need to hydrate more.

While the amount of water consumption is small in both Canines and Felines, it would be advisable to keep enough water, so the owner doesn't fill the water all the time.

4.4 Devices

In this layout, we can control devices used in the household either remotely or automatically. For this project, we have chosen to control devices only related to the house's lights. Two switches control a led and an office lamp; depending on the device's state, we can see each device's state in case the system resets.

Figure 4.10: We can control the devices with switch controls and confirm their states with their texts below

4.5 Security

In our smart home system, many sensors are used for security surveillance, the main one is the magnetic lock switch, which is placed on the door, and we can see the state of the door (Open or Closed). The user

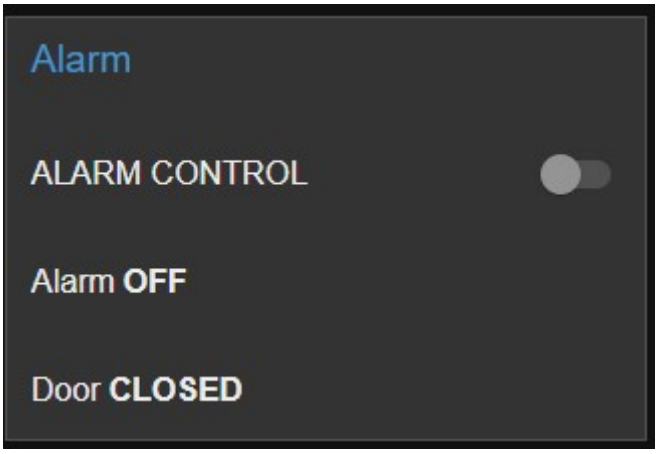


Figure 4.11: We can arm the alarm or disarm it in this layout

can toggle the alarm on and off and confirm the state of the alarm by checking the alarm status below; moreover, a notification is sent to the user when the alarm system is armed. In case the alarm is armed and the door opens, it will trigger a 30-second timer; during that time, the user can disarm that timer in two ways:

- By toggling off the alarm switch
- By using an authorized RFID tag on the RFID reader

Meanwhile, a notification will be sent to the user that the timer has started and the user has that exact time to disarm it. In case none of the above ways does not happen and the timer passes 30 seconds, the

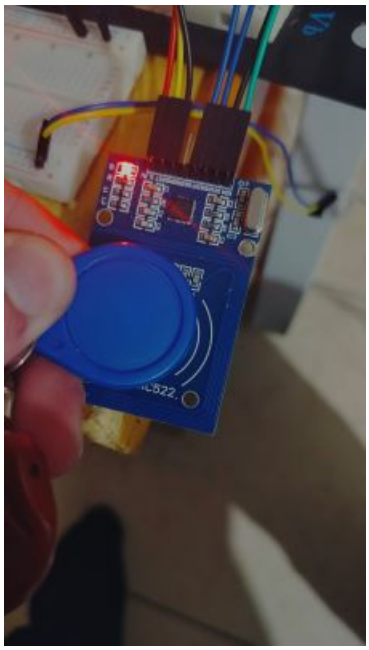


Figure 4.12: Using an authorized RFID tag we gain access to the house

buzzer will sound, and the user will receive an alarm message that the alarm is sound, and in that case, the user can call the police authorities.

We have set the PIR sensors near the door, which will be used to check if a person or a pet is near the door. If a person is seen by both sensors or the above one, it will recognize that a human is in sight. But if the lower PIR sensor solely recognizes sudden temperature change, we have two cases depending on the door's state. If the door is closed and the pet is near it, a notification will be sent to the user to let them know that the pet is near the door and might need to go for a walk. Otherwise, if the pet is detected near an open door, the user will receive a warning that the pet is near an open door and could get out of the apartment.

Figure 4.13: The magnetic lock sensors are placed on the door and door frame

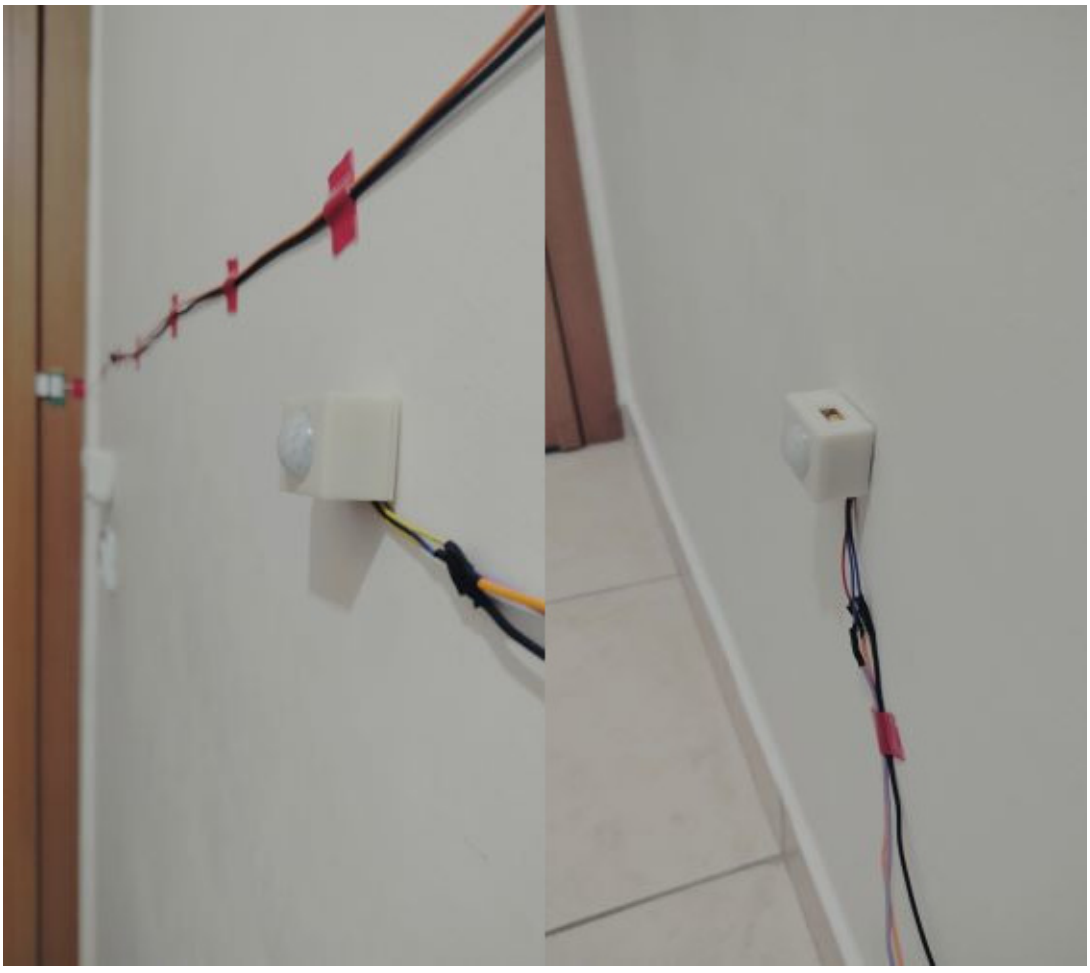


Figure 4.14: Using the PIR sensors we can detect who is passing though (left for humans, right for pets)

Chapter 5: Conclusions

After building and testing this system, it is time to analyze how the user benefited from a smart home device and what are the weaknesses of this system.

5.1 Strong points

1. **Automated devices & Remote Control**: The user can control devices remotely without the need to be in the house.
2. **Reduced energy**: Thanks to remote control, automating devices, and low-resource sensors, the user does not have to worry about overusing a device, as most of them close when a criterion is met.
3. **Time-saving**: If the user is not in the house or in the same room with the device he controls, he can open or close the device remotely using his smartphone.
4. **The user takes better care of his pet**: The user gets updated if his pet wants to go for a walk and gets warnings if he has left the door open so that he will act immediately before his pet escapes. Moreover, providing food and water becomes easier as both functions work automatically.
5. **Magnetic locks cannot be tricked easily**: While this sensor can be tricked by using another magnet in theory, it is hard to do in practice, as the intruder has to create a gap between the door and the sensor which is connected to the microcontroller, in order to place the magnet. However, when he opens the door and places the magnet, the system will recognize that the door is open, setting off the alarm.
6. **PIRs are effective and cannot be tricked easily**: Both PIR sensors work effectively, recognizing whether a human or a pet is near the door. The sensors cannot be tracked even by placing an object in front of them.
7. **Can be used by more than one user**: More users can use this system by scanning the QR code in the Remote RED node.
8. **The user can use the information about his house**: The user can gain information about his house conditions and act accordingly. If the user sees that his house has a low temperature, he can open his A/C remotely or make the A/C do it automatically.
9. **Low cost - Low resource sensors and devices**: Most devices and sensors can easily be acquired in the market and are inexpensive. Moreover, all devices can constantly work without consuming too much energy.

5.2 Weaknesses

1. **All in one system**: All sensors and devices are connected with only one ESP-32; if the microcontroller fails, all sensors and devices cannot be used anymore.

2. **No visual surveillance**: While we have data about our house, if one of the sensors shows incorrect data (if, for example, the Fire Module Sensor senses fire), the user cannot confirm if he is away from his house.

3. **The API key can be stolen**: While the API key is being used on a private system, it can easily get stolen by an attacker. [31]

4. **No login credentials for our dashboard**: Our system works in a private network, but if someone gains unauthorized access to the system, he can easily control the devices or gain information about our house.

5. **RFID tags**: As mentioned above, RFID tags and cards have specific data that can easily be scanned even with a smartphone. Moreover, if someone gains access to the tag's data, they can easily duplicate it in a tag or card using an RFID Reader writer.

Figure 5.1: Using an RFID writer we can duplicate RFID tags

6. **No databases used**: As all sensors and devices show data about the current conditions of the house, the user cannot examine previous measurements in case of an anomaly on the system or if he missed a notification.

7. **No backup server**: If Raspberry Pi or ESP-32 fails due to a power cut, the system cannot work without power.

8. **No notification when the system is down**: In case ESP-32 or Raspberry Pi will not work due to malfunction or power cut, the user will not know if the system is not working. Moreover, some

automated devices will not work correctly. If the user leaves a device open (A/C), the device will still run, and the user has to either restore the server or close the device with other means (A/C remote controller).

Chapter 6: Improvement suggestions

While we have built an effective and reliable system, it can still be improved. In this section, we will suggest ways for the project to improve or inspire those building a smart home system to improve their systems.

6.1 Each device on its own

An intelligent way to improve our system is to differentiate our sensors and devices by using more microcontrollers that can be connected to our system. All microcontrollers can communicate and exchange data with each other. Moreover, the other devices can work without issues if a microcontroller malfunctions. Lastly, in case of a malicious attack on one of the microcontrollers, the attacker will not get easy access to the rest of the devices.

6.1.1 PCB BOARDS

One way to individualize each device would be to design PCBs instead of breadboards. One tool that would help us create PCBs is EAGLE AUTODESK which provides the user to design schematics and component placement, and thanks to FUSION 360, the user can simulate a 3D model of the PCB.

6.2 Adding more devices

Since our system can be more flexible on devices, we can add more devices, such as:

- **Smart Thermostat**: Another way to heat or cool our home is by using thermostats that can be controlled with Wi-Fi or Bluetooth. The user's dilemma is whether he wants a market product or experiments on its own to make one using two cooling or heating relays.

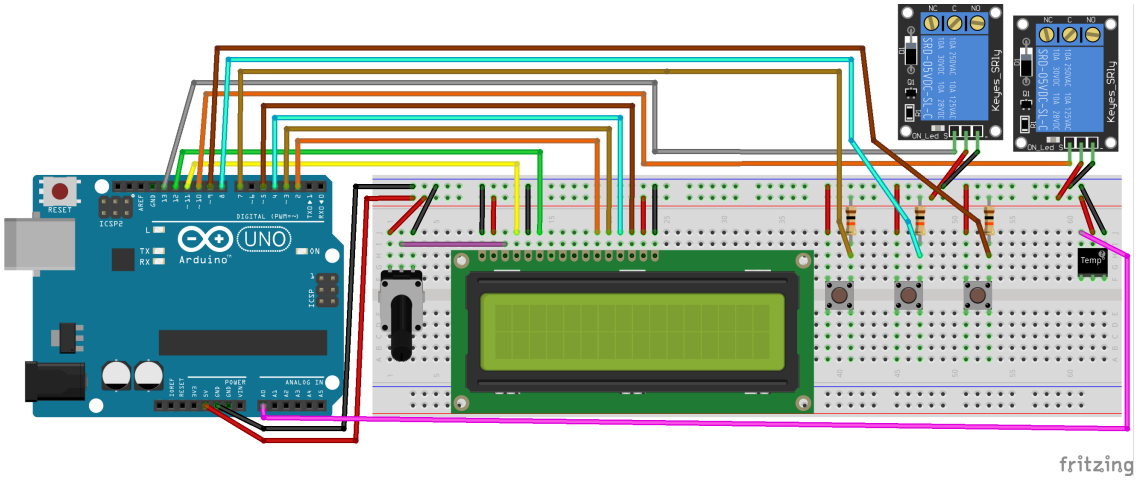


Figure 6.1: One relay is responsible for heating and the other for cooling

- **More Magnetic Sensors**: The user can use more magnetic sensors in different places in his house, like doors or windows, since our pet could escape in one of them. Moreover, we can name the entries (for example, ”Living room Window” or ”Apartment door”), so the user will know which is left open in case they forget it open or an intruder breaks into it.
- **Adding another PIR sensor**: While using 2 PIR sensors is adequate, we could use a third one and place it between the first sensors. The reason to use a third PIR sensor is in case the house has a child.

6.3 Camera Surveillance

To have visual surveillance of our house, we could use a camera module. One device we could use is the OV7670 Camera module sensor.

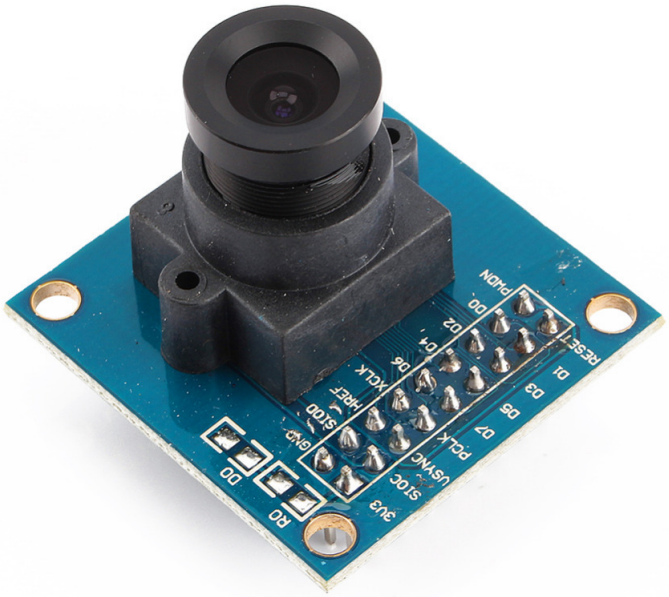


Figure 6.2: A low cost 0V7670 camera sensor
Source

This sensor is powered up by 3.3V, and while its’ resolution is pretty low(656x488 pixels), the image array can operate up to 30 frames per second in VGA (Video Graphics Array). This camera sensor is an inexpensive yet effective way to surveillance our home to confirm several conditions.

6.4 Login Credentials System

A login system would be a great way to make our smart home system more secure. A login system usually demands a username and password, which could be stored in a database. The user will gain access if the database confirms the username’s existence and the password inserted.

6.5 Pet Water Consumption Monitoring

Using databases, we could monitor how much water our pets consume during the day. Monitoring how much water our pet has consumed is vital as dehydration can lead to kidney failure in dogs and cats or a sign of a disease. On the other hand, the pet's owner should be careful about overhydration as it can lead to water intoxication for Canines or could be a sign of diabetes for felines.

Chapter 7: Final Thoughts

A smart home system can be efficient and flexible depending on the user's needs. While there are already companies that produce Smart Home systems and devices the demand for both of them will increase massively as people want to make their lives easier, and more comfortable in their households giving the user the feeling that he is in control of it even when he is not inside. Moreover, smart home systems for pets will be ideal since they will save much time for the owner, and he can use surveillance devices to keep an eye on his pet.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] C. Wilson, T. Hargreaves, and R. Hauxwell-Baldwin, "Smart homes and their users: a systematic analysis and key challenges," *Personal and Ubiquitous Computing*, vol. 19, pp. 463–476, 09 2014.

[2] M. Li, W. Gu, W. Chen, Y. He, Y. Wu, and Y. Zhang, "Smart home: Architecture, technologies and systems," *Procedia Computer Science*, vol. 131, pp. 393–400, 2018.

[3] M. Silverio, "What is a smart device? | built in," 06 2022.

[4] "How our homes became smart: The history of home automation," 01 2019.

[5] N. Hart, "The evolution of the smart home: How it started [part 1]," 03 2022.

[6] A. Bradford, "Z-wave vs. zigbee: Which should you choose?," 10 2021.

[7] H. Mousa, "17 open-source free home automation systems," 08 2019.

[8] M. Armstrong, "The market for smart home devices is expected to boom over the next 5 years."

[9] V. Kumar and R. K. Chawda, "A research paper on smart home," *International Journal of Engineering Applied Sciences and Technology*, vol. 5, pp. 530–532, 07 2020.

[10] L. CVETKOVSKA, "30 smart home statistics for your own high-tech castle," 2020.

[11] R. Santos, "Esp32 mqtt publish subscribe with arduino ide | random nerd tutorials," 06 2018.

[12] C. BasuMallick, "What is mqtt (mq telemetry transport)? working, types, importance, and applications |," 07 2022.

[13] "Eclipse mosquitto," 01 2018.

[14] "5v single-channel relay module," 12 2020.

[15] "Comparing temperature/humidity sensors."

[16] "Esp32 - rfid/nfc."

[17] "All about glass | corning museum of glass," 10 2011.

[18] P. Khatri, "Automatic pet feeder using arduino," 04 2018.

[19] "Arduino - motion sensor | arduino tutorial."

[20] D. Workshop, "Hc-sr501 with arduino raspberry pi," 01 2018.

[21] C. Woodford, "How do fresnel lenses work?," 12 2022.

[22] "Advantages of fresnel lenses | edmund optics."

[23] Ilyena, "Doguino: Sensing a dogs movement with arduino and pirs," 08 2016.

[24] "How hc-sr04 ultrasonic sensor works how to interface it with arduino," 03 2019.

[25] J. Strauchs, “Mythbusters are busted | security magazine.”

[26] “In-depth: How water level sensor works and interface it with arduino,” 11 2019.

[27] L. M. Engineers, “How mq2 gas/smoke sensor works? interface it with arduino,” 08 2018.

[28] https://www.howstuffworks.com/about author.htm, “How remote controls work,” 11 2005.

[29] “Digital ir transmitter module sku dfr0095-dfrobot.”

[30] “Current weather data - openweathermap.”

[31] “Why and when to use api keys | cloud endpoints with openapi.”

[32] S. Marcus, “What’s the ideal room temperature for your home? | ovo energy,” 11 2021.

[33] K. Lazaridis, “Ιδανικές θερμοκρασίες για κατοικίδια,” 05 2022.

[34] “The perfect house temperature for dogs.”

[35] K. Harris, “What is the ideal home humidity level for your family?,” 05 2019.

[36] N. US Department of Commerce, “What is the heat index?.”

[37] D. S. Wooten, “How much water should dogs cats drink each day? | hill’s pet,” 02 2020.

[38] “All about servo motor sg90.”

Appendix A: ESP32 CODE

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "SHTSensor.h"
#include <SPI.h>
#include <MFRC522.h>
#include <ESP32Servo.h>
#include <IRremoteESP8266.h>
#include <IRsend.h>
#include <ir_Gree.h>


#define SS_PIN  5  // ESP32 pin GIOP5
#define RST_PIN 27 // ESP32 pin GIOP27
MFRC522 rfid(SS_PIN,RST_PIN);
// Replace the next variables with your SSID/Password combination
const char* ssid = [redacted];
const char* password = [redacted];

// Add your MQTT Broker IP address
const char* mqtt_server = "[redacted]";

WiFiClient espClient;
PubSubClient client(espClient);

//LED & LIGHT
int led = 12;
int light = 17;

//WATER PUMP
int pump = 16;
int water_x = 0;
```

```
//SHT 40
SHTSensor sht;
float temperature;
//SERVO
Servo myservo;
int servo_pos = 0;
int servo_x;
long servo_period = 350;
unsigned long servo_time_now = 0;

//MAGNETIC LOCKS
const int sensor = 4;

//TIMERS
long lastMsg = 0;
long now_period = 0;
long now_period5 = 0;
char msg[50];

//Buzzer
int buzzer = 14;


// // Define NTP Client to get time
// WiFiUDP ntpUDP;
// NTPClient timeClient(ntpUDP);

//A/C
const uint16_t kIrLed = 15;  // ESP8266 GPIO pin to use. Recommended: 4 (D2).
IRGreeAC ac(kIrLed);  // Set the GPIO to be used for sending messages.
int ac_temp;
```

```
64  //MQ2
65  int const Gas_analog = 34;
66  float gassensorAnalog;
67  //int Gas_digital = 14;
68
69  //FIRE DETECTOR
70  int Fire_analog = 35;    // used for ESP32
71  //int Fire_digital = 12;   // used for ESP32
72
73  //Water level pin
74  const int WaterPin = 32;
75  float waterValue;
76  float edited_fire;
77  // DEFINING VALUES
78  int state ;
79  int value = 0;
80  int alarm_time = 0;
81  int alarm_trigger = 0;
82  int ledState = LOW;
83  int lightState = LOW;
84  int alarmState = LOW;
85  int pumpState = LOW;
86  int buzzerState = LOW;
87  int Humansensor = 33;
88  int Petsensor = 26;
89
90  //pir states
91  int Humanstate = LOW;
92  int Petstate = LOW;
93  int Humanval = 0;
94  int Petval = 0;
95  int pet_w = 0;
96  int pet_warning = 300;
97  String lastState = " ";
98
99  //Warning variables
100 int water_warning = 900;
101 int alarm_on_warning = 0;
102 int alarm_off_warning = 0;
103 int fire_warning = 0;
104 int gas_warning = 0;
105 int alarm_trigger_warning = 0;
106 int alarm_buzzing_warning = 0;
107 int pet_feed_auto = 0;
```

```
byte keyTagUID[4] = {0x09, 0x12, 0x9A, 0xC1};

void setup() {
  //RFID
  SPI.begin(18,19,23); // init SPI bus
  Serial.begin(115200);
  rfid.PCD_Init(); // init MFRC522



  //I2C SHT40
  Wire.begin(21,22);

  //A/C
  ac.begin();
  ac.setFan(1);
  ac.setMode(kGreeHeat);
  ac.setLight(true);
  ac.setXFan(false);
  ac.setLight(true);
  ac.setSleep(false);
  ac.setTurbo(false);

  //Servo
  myservo.attach(13);

  //OUTPUTS AND INPUTS
  //LED
  pinMode(led,OUTPUT);

  //LAMP
  pinMode(light,OUTPUT);

  //WATER PUMP
  pinMode(pump, OUTPUT);

  //BUZZER
  pinMode(buzzer, OUTPUT);

  //GAS
  pinMode (Gas_analog, INPUT);
```

```
150     //MAGNETIC LOCK
151     pinMode(sensor, INPUT_PULLUP);
152
153     //PIRS
154     pinMode(Humansensor, INPUT);
155     pinMode(Petsensor, INPUT);
156
157     digitalWrite(led,ledState);
158     digitalWrite(light,lightState);
159     digitalWrite(pump,pumpState);
160     digitalWrite(buzzer,buzzerState);
161
162
163     //SETTING UP WIFI
164     setup_wifi();
165     client.setServer(mqtt_server, 1883);
166     client.setCallback(callback);
167     // timeClient.begin();
168     // timeClient.setTimeOffset(7200);
169     //CHECK IF SHT WORKS
170     if (sht.init()) {
171         Serial.print("init(): success\n");
172     } else {
173         Serial.print("init(): failed\n");
174     }
175 }
176
177 void setup_wifi() {
178     delay(10);
179
180     // CONNECTING TO WIFI
181     Serial.println();
182     Serial.print("Connecting to ");
183     Serial.println(ssid);
184
185     WiFi.begin(ssid, password);
186
187     while (WiFi.status() != WL_CONNECTED) {
188       delay(500);
189       Serial.print(".");
190     }
191
```

```
192     Serial.println("");
193     Serial.println("WiFi connected");
194     Serial.println("IP address: ");
195     Serial.println(WiFi.localIP());
196   }
197
198   //WHEN MESSAGE ARRIVES
199   void callback(char* topic, byte* message, unsigned int length) {
200     Serial.print("Message arrived on topic: ");
201     Serial.print(topic);
202     Serial.print(". Message: ");
203
204     String messageLed;
205     String messageLight;
206     String messageAlarm;
207     String messageFood;
208     String messageAc;
209     String messageAcheat;
210     String messageAccool;
211     String messageActemp;
212     String messageAcauto;
213     //LED TOPIC
214     if (String(topic) == "michos_test/led"){
215       for (int i = 0; i < length; i++) {
216       Serial.print((char)message[i]);
217       messageLed += (char)message[i];
218       }
219     Serial.println();
220     if (messageLed == "on"){
221       ledState = HIGH;
222       } else if (messageLed == "off"){
223       ledState = LOW;
224         }
225     }
226
227     //LIGHT TOPIC
228     else if (String(topic) == "michos_test/light"){
229       for (int i = 0; i < length; i++) {
230       Serial.print((char)message[i]);
231       messageLight += (char)message[i];
232       }
233     Serial.println();
```

```
234     if (messageLight == "on"){
235       lightState = HIGH;
236       } else if (messageLight == "off"){
237       lightState = LOW;
238         }
239     }
240
241     //ALARM TOPIC
242     else if(String(topic) == "michos_test/alarm"){
243       for (int i = 0; i < length; i++) {
244       Serial.print((char)message[i]);
245       messageAlarm += (char)message[i];
246       }
247     Serial.println();
248     if (messageAlarm == "on"){
249       alarmState = HIGH;
250       } else if (messageAlarm == "off"){
251       alarmState = LOW;
252         }
253     }
254     //A/C TOPICS
255     else if(String(topic) == "michos_test/ac_heat"){
256       for (int i = 0; i < length; i++) {
257       Serial.print((char)message[i]);
258       messageAcheat += (char)message[i];
259       }
260     Serial.println();
261     if (messageAcheat == "on"){
262       ac.setMode(kGreeHeat);
263       ac.send();
264         }
265       }
266
267     else if(String(topic) == "michos_test/ac_cool"){
268       for (int i = 0; i < length; i++) {
269       Serial.print((char)message[i]);
270       messageAccool += (char)message[i];
271       }
272     Serial.println();
273     if (messageAccool == "on"){
274       ac.setMode(kGreeCool);
275       ac.send();
```

```
    }
  }

else if(String(topic) == "michos_test/ac_cool"){
  for (int i = 0; i < length; i++) {
  Serial.print((char)message[i]);
  messageAccool += (char)message[i];
  }
Serial.println();
if (messageAccool == "on"){
  ac.setMode(kGreeCool);
  ac.send();
    }
  }

else if(String(topic) == "michos_test/ac_switch"){
  for (int i = 0; i < length; i++) {
  Serial.print((char)message[i]);
  messageAc += (char)message[i];
  }
Serial.println();
if (messageAc == "on"){
  ac.on();
  ac.send();
  } else if (messageAc == "off"){
  ac.off();
  ac.send();
    }
  }

else if(String(topic) == "michos_test/ac_auto"){
  for(int i = 0; i < length; i++){
    Serial.print((char)message[i]);
    messageAcauto += (char)message[i];
  }
Serial.println();
if(messageAcauto == "on"){
  if(temperature < 22){
    ac.setMode(kGreeHeat);
    ac.setTemp(24);
    ac.on();
    ac.send();
    if(temperature > 24){
      ac.off();
```

```
        ac.send();
        }
      }
    }
  else if(messageAcauto == "off"){
    ac.off();
    ac.send();
  }
  }

  else if(String(topic) == "michos_test/ac_temp"){
    for (int i = 0; i < length; i++) {
      Serial.print((char)message[i]);
      ac.setTemp((char)message[i]);
      messageActemp += (char)message[i];
      int messageAcint = messageActemp.toInt();
      ac.setTemp(messageAcint);
      ac.send();
    }
  Serial.println();
  }

  else if(String(topic) == "michos_test/food"){
    for (int i = 0; i<length; i++){
      Serial.print((char)message[i]);
      messageFood += (char)message[i];
    }
  Serial.println();
  if(messageFood == "on"){
    myservo.attach(13);
    pet_feed_auto = 0;
    for(servo_x = 0; servo_x<1; servo_x++){
      for (servo_pos=0; servo_pos<=180; servo_pos+=1){
      myservo.write(servo_pos);
      delay(7.5);
      }
      for (servo_pos = 180; servo_pos >= 0; servo_pos -= 1) { // goes from 180 deg
      myservo.write(servo_pos);              // tell servo to go to position in va
      delay(0.05);
      myservo.detach();                       // waits 15ms for the servo to reach
        }
      }
```

```
    }
  }
}



void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      // Subscribe
      client.subscribe("michos_test/led");
      client.subscribe("michos_test/light");
      client.subscribe("michos_test/alarm");
      client.subscribe("michos_test/food");
      client.subscribe("michos_test/ac_switch");
      client.subscribe("michos_test/ac_temp");
      client.subscribe("michos_test/ac_heat");
      client.subscribe("michos_test/ac_cool");
      client.subscribe("michos_test/ac_auto");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      //delay(1000);
    }
  }
}


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

  //PIR
```

```
393   //MAGNETIC LOCK
394   state = digitalRead(sensor);
395
396   //LED
397   digitalWrite(led,ledState);
398
399   //LAMP
400   digitalWrite(light,lightState);
401
402   Humanval = digitalRead(Humansensor);
403   Petval = digitalRead(Petsensor);
404
405
406   //SHT READS SAMPLES
407   sht.readSample();
408   //TIMING WITH MILLIS
409   long now = millis();
410
411
412   //3 SECOND SHOW
413   if (now - lastMsg > 3000) {
414     lastMsg = now;
415
416     //READ GAS
417     gassensorAnalog = analogRead(Gas_analog);
418
419     //READ FIRE
420     float firesensorAnalog = analogRead(Fire_analog);
421     edited_fire = 4095-firesensorAnalog;
422
423     //READ HUM AND TEMP
424     float humidity = sht.getHumidity();
425     temperature = sht.getTemperature();
426
427     //MAKE CELSIUS TO FAHR
428     float temperature_far = (temperature * 1.8) +32;
429
430     //READ WATER LEVELS
431     waterValue = analogRead(WaterPin);
432
433     // Compute heat index with Fahr
434     float heat = -42.379 + (2.04901523) * (temperature_far) + (10.14333127) * (hum
435     - 0.00683783 * (temperature_far*temperature_far) - 0.05481717 * (humidity*humi
436     + 0.00085282 * temperature_far * (humidity*humidity) - 0.00000199* (temperatur
```

```
438     //HEAT INDEX FROM FAHR TO C
439     float hic = (heat-32)*0.5556;
440     if (Petval == HIGH){
441       if(Humanval == HIGH){
442         if(Petstate == LOW && Humanstate == LOW){
443           Serial.println("HUMAN");
444           lastState = "HUMAN";
445           Humanstate = HIGH;
446           Petstate = HIGH;
447
448         }
449       }
450       else if(Humanval == LOW){
451         if(Petstate == LOW){
452           Serial.println("PET");
453           lastState = "PET";
454           Petstate = HIGH;
455           if(state == LOW && (pet_w == 0 || pet_w == pet_warning)){
456             client.publish("michos_test/pet_near_door","WALK");
457             pet_w ++;
458         }
459           else if(state == HIGH){
460             client.publish("michos_test/pet_door_open","DOOR");
461           }
462         }
463       }
464     }
465     else if (Humanval == HIGH){
466       if(Petval == HIGH || Petval == LOW){
467         if(Humanstate == LOW && Petstate == LOW){
468           Serial.println("HUMAN");
469           lastState = "HUMAN";
470           Humanstate = HIGH;
471
472         }
473       }
474     }
475     Serial.println(lastState);
476
477
478
479
480     //TEMPERATURE
481     char tempString[8];
```

```
482     dtostrf(temperature, 1, 2, tempString);
483     //Serial.print("Temperature: ");
484     //Serial.println(tempString);
485     client.publish("michos_test/temp", tempString);
486
487     //HUMIDITY
488     char humString[8];
489     dtostrf(humidity, 1, 2, humString);
490     //Serial.print("Humidity: ");
491     //Serial.println(humString);
492     client.publish("michos_test/hum", humString);
493
494
495     //HEAT
496     char heatString[8];
497     dtostrf(hic, 1, 2, heatString);
498     //Serial.print("Heat: ");
499     //Serial.println(heatString);
500     client.publish("michos_test/heat", heatString);
501
502     //GAS
503     char gasString[8];
504     dtostrf(gassensorAnalog, 1, 2, gasString);
505     //Serial.print("Gas: ");
506     //Serial.println(gasString);
507     client.publish("michos_test/gas", gasString);
508
509     //FIRE
510     char fireString[8];
511     dtostrf(edited_fire, 1, 2, fireString);
512     //Serial.print("Fire: ");
513     //Serial.println(fireString);
514     client.publish("michos_test/fire", fireString);
515
516     //LED
517     //Serial.print("Led: ");
518     //Serial.println(ledState);
519
520     //WATER
521     char waterString[8];
522     dtostrf(waterValue, 1, 2, waterString);
523     //Serial.print("Water : " );
```

```
524       //Serial.println(waterValue);
525       client.publish("michos_test/water", waterString);
526
527       //LIGHT
528       //Serial.print("Light: ");
529       //Serial.println(lightState);
530
531       //DOOR
532       //Serial.print("Door: ");
533       //Serial.println(state);
534
535       //ALARM
536       //Serial.print("Alarm: ");
537       //Serial.println(alarmState);
538     }
539
540
541
542     if (Humanstate == HIGH){
543           Serial.println("Human Motion stopped!");
544           Humanstate = LOW;       // update variable state to LOW
545           lastState = "";
546       }
547     else if(Petstate == HIGH){
548         Serial.println("Pet Motion stopped");
549         Petstate = LOW;
550         lastState = "";
551         pet_w = 0;
552         pet_warning = 300;
553         }
554
555
556
557     if (state == LOW){
558       client.publish("michos_test/door_off", "CLOSED");
559     }
560     else if (state == HIGH){
561       client.publish("michos_test/door_on", "OPEN");
562     }
563
564     if(ledState == LOW){
565       client.publish("michos_test/led_off", "OFF");
```

```
566     }
567     else if (ledState == HIGH){
568       client.publish("michos_test/led_on", "ON");
569     }
570
571     if(lightState == LOW){
572       client.publish("michos_test/light_off", "OFF");
573     }
574     else if (lightState == HIGH){
575       client.publish("michos_test/light_on", "ON");
576     }
577
578
579     if(alarmState == LOW){
580       alarm_trigger = 0;
581       alarm_time = 0;
582       client.publish("michos_test/alarm_off", "OFF");
583       alarm_off_warning++;
584       alarm_on_warning = 0;
585
586       if (alarm_off_warning==5){
587         client.publish("michos_test/alarm_disabled","Alarm is off");
588       }
589     }
590     else if (alarmState == HIGH){
591       client.publish("michos_test/alarm_on", "ON");
592       alarm_on_warning++;
593       alarm_off_warning = 0;
594       if(alarm_on_warning == 5){
595         client.publish("michos_test/alarm_enabled","Alarm is on");
596       }
597       //CHECK IF DOOR IS OPEN TO TRIGGER THE 30-SECONDS TIMING
598       if(state == HIGH){
599           alarm_trigger = 1;
600         }
601     }
602
603     //1 SECOND SHOW
604     if(now - now_period >= 1000){
605       now_period = millis();
606       pet_feed_auto++;
607       if(pet_feed_auto == 21600){
```

```
608         myservo.attach(13);
609         pet_feed_auto = 0;
610       for(servo_x = 0; servo_x<1; servo_x++){
611         for (servo_pos=0; servo_pos<=180; servo_pos+=1){
612           myservo.write(servo_pos);
613           delay(7.5);
614         }
615         for (servo_pos = 180; servo_pos >= 0; servo_pos -= 1) { // goes from 180 degrees to 0 degrees
616           myservo.write(servo_pos);              // tell servo to go to position in variable 'pos'
617           delay(0.05);
618           myservo.detach();                       // waits 15ms for the servo to reach the position
619           }
620         }
621       }
622       //FIRE WARNING
623       if(edited_fire>=2000){
624         digitalWrite(buzzer,HIGH);
625         //Serial.println("FIRE!");
626         fire_warning ++;
627         if(fire_warning == 1 || fire_warning == 2 || fire_warning == 3){
628           client.publish("michos_test/fire_alarm","FIRE");
629         }
630       }
631       else{
632         digitalWrite(buzzer,LOW);
633         fire_warning = 0;
634       }
635
636       //GAS ALARM
637       if(gassensorAnalog>=2000){
638         digitalWrite(buzzer,HIGH);
639         //Serial.println("GAS!");
640         gas_warning ++;
641         if(gas_warning == 1 || gas_warning == 2 || gas_warning == 3)
642         client.publish("michos_test/gas_alarm","GAS");
643       }
644       else{
645         gas_warning = 0;
646         digitalWrite(buzzer,LOW);
647       }
648
649       //WATER PUMPS FOR 5 SEC IF LOW WATER LEVELS
```

```
650     if(waterValue <= 2800){
651       digitalWrite(pump,HIGH);
652       water_x++;
653       if(water_x > 5){
654         digitalWrite(pump,LOW);
655       }
656     }
657     else if(waterValue>2800){
658       digitalWrite(pump,LOW);
659       water_x = 0;
660       water_warning = 900;
661     }
662 
663 
664     // Serial.print("water_time");
665     // Serial.println(water_x);
666 
667     //WARN USER FOR LOW LEVEL WATERS
668     if(water_x==water_warning){
669       //Serial.println("The pet does not have the drink!");
670       client.publish("michos_test/empty_drink","WARNING!");
671       water_warning += 3600;
672     }
673 
674     if(alarm_trigger == 1){
675       alarm_time++;
676     }
677     if(alarm_time == 1){
678       client.publish("michos_test/Alarm_trigger","The alarm is triggered");
679     }
680     if(alarm_time == 30 || alarm_time == 31 || alarm_time == 32){
681       client.publish("michos_test/ALARM_BUZZING","ALARM");
682     }
683 
684     if(alarm_time>=30){
685       //Serial.println("ALARM");
686       digitalWrite(buzzer,HIGH);
687     }
688 
689   }
```

```
691   //RFID
692   if (rfid.PICC_IsNewCardPresent()) { // new tag is available
693     if (rfid.PICC_ReadCardSerial()) { // NUID has been readed
694       MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
695       if (rfid.uid.uidByte[0] == keyTagUID[0] &&
696           rfid.uid.uidByte[1] == keyTagUID[1] &&
697           rfid.uid.uidByte[2] == keyTagUID[2] &&
698           rfid.uid.uidByte[3] == keyTagUID[3] ) {
699             Serial.println("Access is granted");
700             alarm_trigger = 0;
701             alarm_time = 0;
702             digitalWrite(buzzer,LOW);
703             client.publish("michos_test/trigger_defused","Trigger is defused");
704             Serial.println();
705             }
706       else{
707         Serial.print("Access denied");
708         Serial.println();
709       }
710       rfid.PICC_HaltA();
711       rfid.PCD_StopCrypto1();
712     }
713   }
714 }
```

Appendix A: NODE RED BACKEND

```
    {
        "id": "b2012bc619b95d6e",
        "type": "debug",
        "z": "283dfd7caffc1729",
        "name": "",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 290,
        "y": 100,
        "wires": []
    },
    {
        "id": "a824636f9020c5fe",
        "type": "ui_gauge",
        "z": "283dfd7caffc1729",
        "name": "",
        "group": "61285987.c20328",
        "order": 2,
        "width": 0,
        "height": 0,
        "gtype": "donut",
        "title": "Humidity",
        "label": "%",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#ff0000",
            "#04fb4e",
            "#0022ff"
        ],
        "seg1": "40",
        "seg2": "60",
        "className": "",
        "x": 280,
        "y": 140,
        "wires": []
    },
```

```
    {
        "id": "60f5822fee61c1ea",
        "type": "mqtt out",
        "z": "283dfd7caffc1729",
        "name": "",
        "topic": "michos_test/led",
        "qos": "",
        "retain": "",
        "respTopic": "",
        "contentType": "",
        "userProps": "",
        "correl": "",
        "expiry": "",
        "broker": "10e78a89.5b4fd5",
        "x": 760,
        "y": 100,
        "wires": []
    },
    {
        "id": "39b30bdf70b3b706",
        "type": "ui_switch",
        "z": "283dfd7caffc1729",
        "name": "",
        "label": "LED CONTROL",
        "tooltip": "",
        "group": "2aac9bc622199dd5",
        "order": 1,
        "width": 0,
        "height": 0,
        "passthru": true,
        "decouple": "false",
        "topic": "michos_test/led",
        "topicType": "msg",
        "style": "",
        "onvalue": "on",
        "onvalueType": "str",
        "onicon": "",
        "oncolor": "",
        "offvalue": "off",
        "offvalueType": "str",
        "officon": "",
        "offcolor": "",
        "animate": false,
```

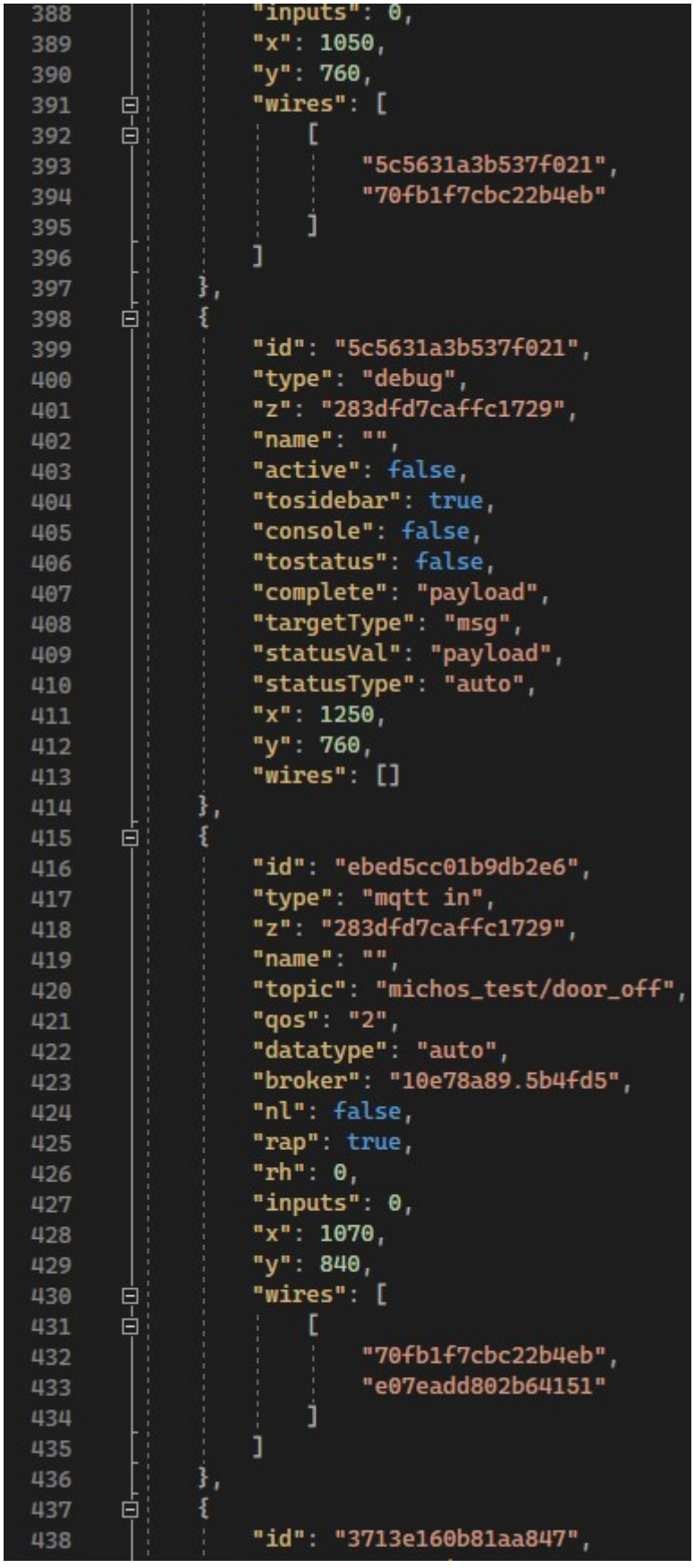

"className": "",
"x": 560,
"y": 100,
"wires": [
[
"60f5822fee61c1ea"
]
]
},
{
"id": "799b44e833ea78aa",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/heat",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 100,
"y": 200,
"wires": [
[
"abfae2e450536bbe",
"e69955a49aec34da"
]
]
},
{
"id": "abfae2e450536bbe",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "",
"active": false,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"statusVal": "",
"statusType": "auto",
"x": 270,
"y": 180,
"wires": []
},


{
"id": "e69955a49aec34da",
"type": "ui_gauge",
"z": "283dfd7caffc1729",
"name": "",
"group": "61285987.c20328",
"order": 3,
"width": 0,
"height": 0,
"gtype": "gage",
"title": "Heat",
"label": "units",
"format": "{{value}}",
"min": "20",
"max": "54",
"colors": [
"#00ffaa",
"#ff8800",
"#fa0000"
],
"seg1": "26",
"seg2": "41",
"className": "",
"x": 290,
"y": 240,
"wires": []
},
{
"id": "6ce8e0f047e36070",
"type": "ui_gauge",
"z": "283dfd7caffc1729",
"name": "",
"group": "61285987.c20328",
"order": 1,
"width": 0,
"height": 0,
"gtype": "gage",
"title": "Temperature",
"label": "Celsius",
"format": "{{value}}",
"min": "0",
"max": "40",
"colors": [
"#00f5a3",
"#fff700",
"#fa0000"
],
"seg1": "20",
"seg2": "27",
"className": "",
"x": 290,


"y": 60,
"wires": []
},
{
"id": "1c822f9fe9c42ba0",
"type": "mqtt out",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/light",
"qos": "",
"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "10e78a89.5b4fd5",
"x": 770,
"y": 160,
"wires": []
},
{
"id": "d6a9b7bb3c2b2e34",
"type": "ui_switch",
"z": "283dfd7caffc1729",
"name": "",
"label": "LIGHT CONTROL",
"tooltip": "",
"group": "2aac9bc622199dd5",
"order": 3,
"width": 0,
"height": 0,
"passthru": true,
"decouple": "false",
"topic": "michos_test/light",
"topicType": "msg",
"style": "",
"onvalue": "on",
"onvalueType": "str",
"onicon": "",
"oncolor": "",
"offvalue": "off",
"offvalueType": "str",
"officon": "",
"offcolor": "",
"animate": false,
"className": "",
"x": 570,
"y": 160,


"wires": [
[
"1c822f9fe9c42ba0"
]
]
},
{
"id": "5e167585bda8fb63",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/gas",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 100,
"y": 300,
"wires": [
[
"c9f7948093047a3e",
"453b6c0929741317"
]
]
},
{
"id": "c9f7948093047a3e",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "",
"active": false,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"statusVal": "",
"statusType": "auto",
"x": 310,
"y": 280,
"wires": []
},
{
"id": "453b6c0929741317",
"type": "ui_chart",
"z": "283dfd7caffc1729",
"name": "",


"group": "a55c6f726e71ceba",
"order": 1,
"width": 0,
"height": 0,
"label": "Gas",
"chartType": "line",
"legend": "false",
"xformat": "HH:mm:ss",
"interpolate": "linear",
"nodata": "",
"dot": false,
"ymin": "0",
"ymax": "5000",
"removeOlder": "10",
"removeOlderPoints": "",
"removeOlderUnit": "60",
"cutout": 0,
"useOneColor": false,
"useUTC": false,
"colors": [
"#eeff00",
"#aec7e8",
"#ff7f0e",
"#2ca02c",
"#98df8a",
"#d62728",
"#ff9896",
"#9467bd",
"#c5b0d5"
],
"outputs": 1,
"useDifferentColor": false,
"className": "",
"x": 290,
"y": 320,
"wires": [
[]
]
},
{
"id": "f074cb7810e773a5",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/door_on",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,


"inputs": 0,
"x": 1050,
"y": 760,
"wires": [
[
"5c5631a3b537f021",
"70fb1f7cbc22b4eb"
]
]
},
{
"id": "5c5631a3b537f021",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "",
"active": false,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"statusVal": "payload",
"statusType": "auto",
"x": 1250,
"y": 760,
"wires": []
},
{
"id": "ebed5cc01b9db2e6",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/door_off",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 1070,
"y": 840,
"wires": [
[
"70fb1f7cbc22b4eb",
"e07eadd802b64151"
]
]
},
{
"id": "3713e160b81aa847",


"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/led_on",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 1070,
"y": 620,
"wires": [
[
"fd9248aa06ebc9d2",
"11fdd234a5b09107"
]
]
},
{
"id": "48967a7e1dcedfd3",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/led_off",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 1070,
"y": 680,
"wires": [
[
"9dd2fa74f1e5ebea",
"11fdd234a5b09107"
]
]
},
{
"id": "fd9248aa06ebc9d2",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "",
"active": false,
"tosidebar": true,
"console": false,
"tostatus": false,


"complete": "payload",
"targetType": "msg",
"statusVal": "",
"statusType": "auto",
"x": 1250,
"y": 600,
"wires": []
},
{
"id": "9dd2fa74f1e5ebea",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "",
"active": false,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"statusVal": "payload",
"statusType": "auto",
"x": 1250,
"y": 700,
"wires": []
},
{
"id": "11fdd234a5b09107",
"type": "ui_text",
"z": "283dfd7caffc1729",
"group": "2aac9bc622199dd5",
"order": 2,
"width": 0,
"height": 0,
"name": "",
"label": "LED",
"format": "{{msg.payload}}",
"layout": "row-left",
"className": "",
"x": 1250,
"y": 660,
"wires": []
},
{
"id": "ac8af0085fc1a368",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/light_on",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",


"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 1050,
"y": 460,
"wires": [
[
"0e68873069682ed4",
"6a85ec0c05c2ed40"
]
]
},
{
"id": "e93ed8e979bf8d7e",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/light_off",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 1050,
"y": 540,
"wires": [
[
"3d7e5098a4051370",
"6a85ec0c05c2ed40"
]
]
},
{
"id": "0e68873069682ed4",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "",
"active": false,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"statusVal": "",
"statusType": "auto",
"x": 1270,
"y": 460,
"wires": []


},
{
"id": "3d7e5098a4051370",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "",
"active": false,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"statusVal": "payload",
"statusType": "auto",
"x": 1250,
"y": 540,
"wires": []
},
{
"id": "6a85ec0c05c2ed40",
"type": "ui_text",
"z": "283dfd7caffc1729",
"group": "2aac9bc622199dd5",
"order": 4,
"width": 0,
"height": 0,
"name": "",
"label": "LIGHT",
"format": "{{msg.payload}}",
"layout": "row-left",
"className": "",
"x": 1250,
"y": 500,
"wires": []
},
{
"id": "70fb1f7cbc22b4eb",
"type": "ui_text",
"z": "283dfd7caffc1729",
"group": "c021dfb154dab37b",
"order": 3,
"width": 0,
"height": 0,
"name": "",
"label": "Door",
"format": "{{msg.payload}}",
"layout": "row-left",
"className": "",
"x": 1250,
"y": 820,
"wires": []

```
    },
    {
        "id": "f05ca86a75e28550",
        "type": "debug",
        "z": "283dfd7caffc1729",
        "name": "",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 290,
        "y": 20,
        "wires": []
    },
    {
        "id": "e07eadd802b64151",
        "type": "debug",
        "z": "283dfd7caffc1729",
        "name": "",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 1250,
        "y": 880,
        "wires": []
    },
    {
        "id": "1ba820406ab2aea2",
        "type": "comment",
        "z": "283dfd7caffc1729",
        "name": "Remote access using Smartphone",
        "info": "",
        "x": 140,
        "y": 1080,
        "wires": []
    },
    {
        "id": "dfb31f4285c5f82e",
        "type": "comment",
        "z": "283dfd7caffc1729",
        "name": "Taking information from devices",
        "info": "",
```

```
            "x": 150,
            "y": 540,
            "wires": []
        },
        {
            "id": "bede3abf3f203d23",
            "type": "comment",
            "z": "283dfd7caffc1729",
            "name": "Remote Control Devices",
            "info": "",
            "x": 630,
            "y": 600,
            "wires": []
        },
        {
            "id": "f80311ba76bc630d",
            "type": "comment",
            "z": "283dfd7caffc1729",
            "name": "Confirming the states of devices",
            "info": "",
            "x": 1030,
            "y": 1140,
            "wires": []
        },
        {
            "id": "6f57538665569c7b",
            "type": "ui_switch",
            "z": "283dfd7caffc1729",
            "name": "",
            "label": "ALARM CONTROL",
            "tooltip": "",
            "group": "c021dfb154dab37b",
            "order": 1,
            "width": 0,
            "height": 0,
            "passthru": true,
            "decouple": "false",
            "topic": "michos_test/alarm",
            "topicType": "msg",
            "style": "",
            "onvalue": "on",
            "onvalueType": "str",
            "onicon": "",
            "oncolor": "",
            "offvalue": "off",
            "offvalueType": "str",
            "officon": "",
            "offcolor": "",
            "animate": true,
            "className": "",
            "x": 570,
```


"y": 240,
"wires": [
[
"5bd8232f219adfb3"
]
]
},
{
"id": "5f1ae543ed012986",
"type": "remote-access",
"z": "283dfd7caffc1729",
"configNode": "bd7da532ce73cef0",
"name": "",
"verbose": 0,
"x": 100,
"y": 1020,
"wires": [
[]
]
},
{
"id": "5bd8232f219adfb3",
"type": "mqtt out",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/alarm",
"qos": "",
"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "10e78a89.5b4fd5",
"x": 790,
"y": 240,
"wires": []
},
{
"id": "8be48a450502d073",
"type": "ui_button",
"z": "283dfd7caffc1729",
"name": "",
"group": "72c8164bb9389191",
"order": 1,
"width": 0,
"height": 0,
"passthru": true,
"label": "Give Food",
"tooltip": "",
"color": "",


"bgcolor": "",
"className": "",
"icon": "",
"payload": "on",
"payloadType": "str",
"topic": "michos_test/food",
"topicType": "msg",
"x": 550,
"y": 40,
"wires": [
[
"ddfd097e3d7f3976"
]
]
},
{
"id": "ddfd097e3d7f3976",
"type": "mqtt out",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/food",
"qos": "",
"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "10e78a89.5b4fd5",
"x": 790,
"y": 40,
"wires": []
},
{
"id": "07c57ea5fbe2a133",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/alarm_on",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 1040,
"y": 960,
"wires": [
[
"2ea5089d214c9046",


"7c5d9d6bcf0836bf"
]
]
},
{
"id": "a31975bbc0b0f997",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/alarm_off",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 1040,
"y": 1060,
"wires": [
[
"2ea5089d214c9046",
"4a103d9a735360f9"
]
]
},
{
"id": "2ea5089d214c9046",
"type": "ui_text",
"z": "283dfd7caffc1729",
"group": "c021dfb154dab37b",
"order": 2,
"width": 0,
"height": 0,
"name": "",
"label": "Alarm",
"format": "{{msg.payload}}",
"layout": "row-left",
"className": "",
"x": 1250,
"y": 1000,
"wires": []
},
{
"id": "7c5d9d6bcf0836bf",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "",
"active": false,
"tosidebar": true,
"console": false,

```
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 1250,
        "y": 940,
        "wires": []
    },
    {
        "id": "4a103d9a735360f9",
        "type": "debug",
        "z": "283dfd7caffc1729",
        "name": "",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 1250,
        "y": 1080,
        "wires": []
    },
    {
        "id": "8d672610f21d2fed",
        "type": "mqtt in",
        "z": "283dfd7caffc1729",
        "name": "",
        "topic": "michos_test/fire",
        "qos": "2",
        "datatype": "auto",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 100,
        "y": 360,
        "wires": [
            [
                "759663e507fce3bc",
                "5c101efc8974b85b"
            ]
        ]
    },
    {
        "id": "759663e507fce3bc",
        "type": "ui_chart",
```


"z": "283dfd7caffc1729",
"name": "",
"group": "a55c6f726e71ceba",
"order": 3,
"width": 0,
"height": 0,
"label": "Fire",
"chartType": "line",
"legend": "false",
"xformat": "HH:mm:ss",
"interpolate": "linear",
"nodata": "",
"dot": false,
"ymin": "0",
"ymax": "5000",
"removeOlder": "10",
"removeOlderPoints": "",
"removeOlderUnit": "60",
"cutout": 0,
"useOneColor": false,
"useUTC": false,
"colors": [
"#ff0000",
"#aec7e8",
"#ff7f0e",
"#2ca02c",
"#98df8a",
"#d62728",
"#ff9896",
"#9467bd",
"#c5b0d5"
],
"outputs": 1,
"useDifferentColor": false,
"className": "",
"x": 290,
"y": 400,
"wires": [
[]
]
},
{
"id": "e97456b0cbd74741",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/water",
"qos": "2",
"datatype": "auto",
"broker": "10e78a89.5b4fd5",
"nl": false,

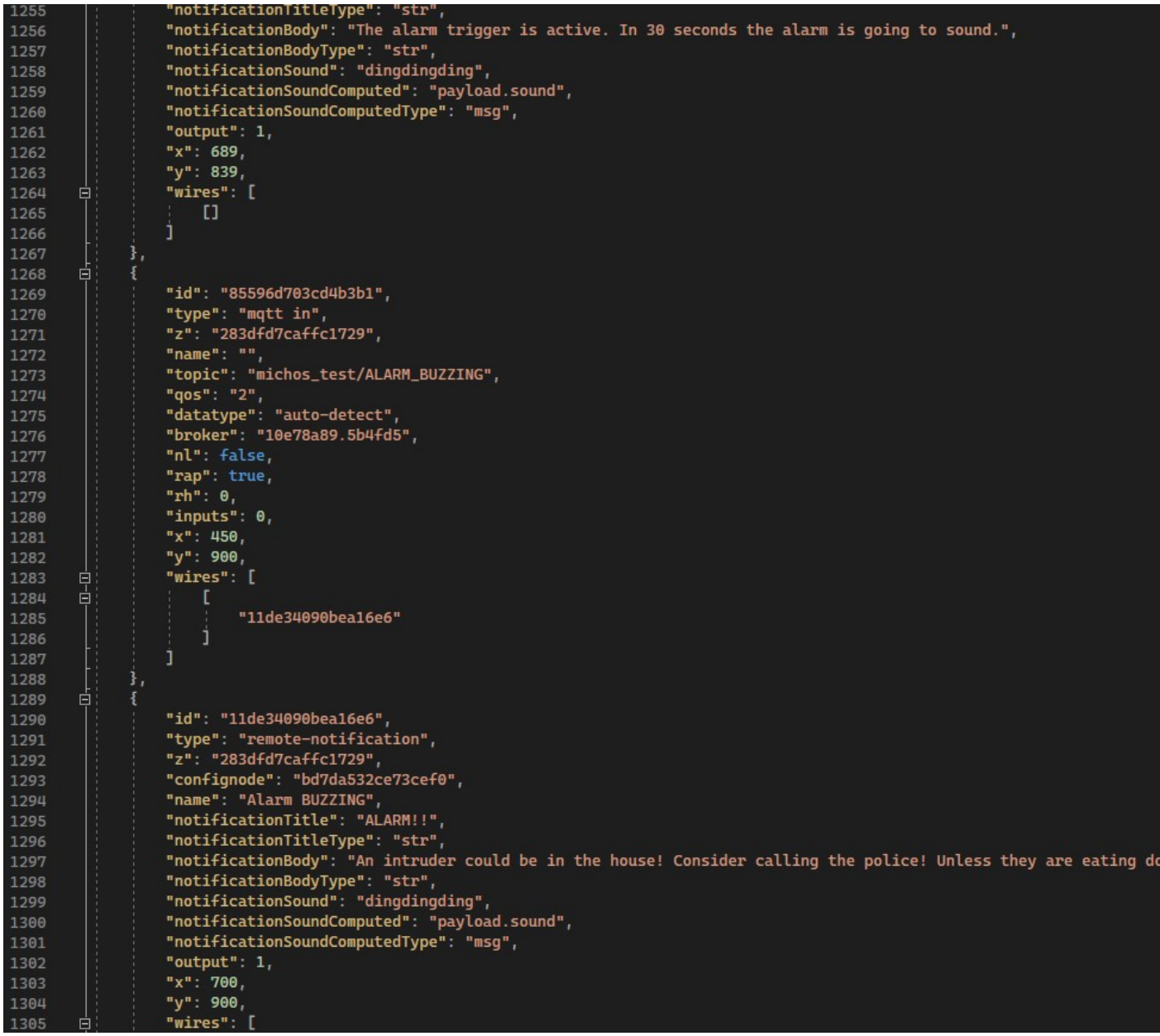

"rap": true,
"rh": 0,
"inputs": 0,
"x": 110,
"y": 440,
"wires": [
[
"cb1981b09f925093",
"1086dc3d6e0ae830"
]
]
},
{
"id": "cb1981b09f925093",
"type": "ui_gauge",
"z": "283dfd7caffc1729",
"name": "",
"group": "2d0c2ebd8915f48e",
"order": 1,
"width": 0,
"height": 0,
"gtype": "donut",
"title": "Water",
"label": "units",
"format": "{{value}}",
"min": 0,
"max": "4095",
"colors": [
"#ff0000",
"#00faa7",
"#0091ff"
],
"seg1": "",
"seg2": "",
"className": "",
"x": 290,
"y": 480,
"wires": []
},
{
"id": "b97bc1c0.cb899",
"type": "ui_template",
"z": "283dfd7caffc1729",
"group": "c982fbb8.1deb38",
"name": "Clock Toolbar",
"order": 2,
"width": "0",
"height": "0",
"format": "<script id=\"titleScript\"
"storeOutMessages": false,
"fwdInMessages": false,

```
        "resendOnRefresh": false,
        "templateScope": "global",
        "className": "",
        "x": 100,
        "y": 920,
        "wires": [
            []
        ]
    },
    {
        "id": "e10bf2e1f4e5ab18",
        "type": "mqtt in",
        "z": "283dfd7caffc1729",
        "name": "",
        "topic": "michos_test/empty_drink",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 430,
        "y": 660,
        "wires": [
            [
                "fd199e08d7b267a5"
            ]
        ]
    },
    {
        "id": "fd199e08d7b267a5",
        "type": "remote-notification",
        "z": "283dfd7caffc1729",
        "confignode": "bd7da532ce73cef0",
        "name": "Water Warning",
        "notificationTitle": "WARNING!",
        "notificationTitleType": "str",
        "notificationBody": "The pet does not have the drink!",
        "notificationBodyType": "str",
        "notificationSound": "pristine",
        "notificationSoundComputed": "payload.sound",
        "notificationSoundComputedType": "msg",
        "output": 1,
        "x": 680,
        "y": 660,
        "wires": [
            []
        ]
    },
    {
```

```
        "id": "150a524cd38623c6",
        "type": "comment",
        "z": "283dfd7caffc1729",
        "name": "Clock",
        "info": "",
        "x": 90,
        "y": 960,
        "wires": []
    },
    {
        "id": "ad08118a05768744",
        "type": "mqtt in",
        "z": "283dfd7caffc1729",
        "name": "",
        "topic": "michos_test/alarm_disabled",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 440,
        "y": 720,
        "wires": [
            [
                "26ac02aba968c512"
            ]
        ]
    },
    {
        "id": "65fb27108df1ecb9",
        "type": "mqtt in",
        "z": "283dfd7caffc1729",
        "name": "",
        "topic": "michos_test/alarm_enabled",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 440,
        "y": 780,
        "wires": [
            [
                "9d8d99753e86d5a8"
            ]
        ]
    },
```

```
    {
        "id": "26ac02aba968c512",
        "type": "remote-notification",
        "z": "283dfd7caffc1729",
        "confignode": "bd7da532ce73cef0",
        "name": "Alarm Disabled",
        "notificationTitle": "UPDATE",
        "notificationTitleType": "str",
        "notificationBody": "The alarm is disabled!",
        "notificationBodyType": "str",
        "notificationSound": "pristine",
        "notificationSoundComputed": "payload.sound",
        "notificationSoundComputedType": "msg",
        "output": 1,
        "x": 680,
        "y": 720,
        "wires": [
            []
        ]
    },
    {
        "id": "9d8d99753e86d5a8",
        "type": "remote-notification",
        "z": "283dfd7caffc1729",
        "confignode": "bd7da532ce73cef0",
        "name": "Alarm Enabled",
        "notificationTitle": "UPDATE",
        "notificationTitleType": "str",
        "notificationBody": "The alarm is enabled!",
        "notificationBodyType": "str",
        "notificationSound": "pristine",
        "notificationSoundComputed": "payload.sound",
        "notificationSoundComputedType": "msg",
        "output": 1,
        "x": 680,
        "y": 780,
        "wires": [
            []
        ]
    },
    {
        "id": "5c101efc8974b85b",
        "type": "debug",
        "z": "283dfd7caffc1729",
        "name": "",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
```


"statusVal": "",
"statusType": "auto",
"x": 293.6666564941406,
"y": 360.6666564941406,
"wires": []
},
{
"id": "1086dc3d6e0ae830",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "",
"active": false,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"statusVal": "",
"statusType": "auto",
"x": 300.6666564941406,
"y": 437.6666564941406,
"wires": []
},
{
"id": "7bcd7d2a47e08a0c",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/Alarm_trigger",
"qos": "2",
"datatype": "auto-detect",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 430,
"y": 840,
"wires": [
[
"513fa97a72195670"
]
]
},
{
"id": "513fa97a72195670",
"type": "remote-notification",
"z": "283dfd7caffc1729",
"confignode": "bd7da532ce73cef0",
"name": "Alarm Triggered",
"notificationTitle": "WARNING!",


"notificationTitleType": "str",
"notificationBody": "The alarm trigger is active. In 30 seconds the alarm is going to sound.",
"notificationBodyType": "str",
"notificationSound": "dingdingding",
"notificationSoundComputed": "payload.sound",
"notificationSoundComputedType": "msg",
"output": 1,
"x": 689,
"y": 839,
"wires": [
[]
]
},
{
"id": "85596d703cd4b3b1",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/ALARM_BUZZING",
"qos": "2",
"datatype": "auto-detect",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 450,
"y": 900,
"wires": [
[
"11de34090bea16e6"
]
]
},
{
"id": "11de34090bea16e6",
"type": "remote-notification",
"z": "283dfd7caffc1729",
"confignode": "bd7da532ce73cef0",
"name": "Alarm BUZZING",
"notificationTitle": "ALARM!!",
"notificationTitleType": "str",
"notificationBody": "An intruder could be in the house! Consider calling the police! Unless they are eating do
"notificationBodyType": "str",
"notificationSound": "dingdingding",
"notificationSoundComputed": "payload.sound",
"notificationSoundComputedType": "msg",
"output": 1,
"x": 700,
"y": 900,
"wires": [

```
            []
        ]
    },
    {
        "id": "83feede57dffdf35",
        "type": "mqtt in",
        "z": "283dfd7caffc1729",
        "name": "",
        "topic": "michos_test/trigger_defused",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 420,
        "y": 960,
        "wires": [
            [
                "dd4feef83e7e71dc"
            ]
        ]
    },
    {
        "id": "dd4feef83e7e71dc",
        "type": "remote-notification",
        "z": "283dfd7caffc1729",
        "confignode": "bd7da532ce73cef0",
        "name": "Trigger Defused",
        "notificationTitle": "UPDATE",
        "notificationTitleType": "str",
        "notificationBody": "The alarm trigger is defused!",
        "notificationBodyType": "str",
        "notificationSound": "pristine",
        "notificationSoundComputed": "payload.sound",
        "notificationSoundComputedType": "msg",
        "output": 1,
        "x": 680,
        "y": 960,
        "wires": [
            []
        ]
    },
    {
        "id": "0f49d7bb71b66e6b",
        "type": "ui_slider",
        "z": "283dfd7caffc1729",
        "name": "",
        "label": "A/C Temperature",
        "tooltip": "",
```


"group": "39645e65241f69c9",
"order": 3,
"width": 0,
"height": 0,
"passthru": true,
"outs": "end",
"topic": "michos_test/ac_temp",
"topicType": "msg",
"min": "20",
"max": "27",
"step": 1,
"className": "",
"x": 550,
"y": 480,
"wires": [
[
"21d7b6adb780b000"
]
]
},
{
"id": "5716ebe44cc97fba",
"type": "ui_switch",
"z": "283dfd7caffc1729",
"name": "",
"label": "A/C",
"tooltip": "",
"group": "39645e65241f69c9",
"order": 1,
"width": 0,
"height": 0,
"passthru": false,
"decouple": "false",
"topic": "topic",
"topicType": "msg",
"style": "",
"onvalue": "on",
"onvalueType": "str",
"onicon": "",
"oncolor": "",
"offvalue": "off",
"offvalueType": "str",
"officon": "",
"offcolor": "",
"animate": false,
"className": "",
"x": 550,
"y": 420,
"wires": [
[
"e03fcfbb278754e5"


]
]
},
{
"id": "ab2b547f38fe887c",
"type": "ui_button",
"z": "283dfd7caffc1729",
"name": "",
"group": "39645e65241f69c9",
"order": 5,
"width": 0,
"height": 0,
"passthru": false,
"label": "Heat",
"tooltip": "",
"color": "",
"bgcolor": "red",
"className": "",
"icon": "",
"payload": "on",
"payloadType": "str",
"topic": "topic",
"topicType": "msg",
"x": 550,
"y": 360,
"wires": [
[
"b5c2a3c36530954b"
]
]
},
{
"id": "7e055bcd1feeb16e",
"type": "ui_button",
"z": "283dfd7caffc1729",
"name": "",
"group": "39645e65241f69c9",
"order": 4,
"width": 0,
"height": 0,
"passthru": false,
"label": "Cool",
"tooltip": "",
"color": "",
"bgcolor": "#00E1FF",
"className": "",
"icon": "",
"payload": "on",
"payloadType": "str",
"topic": "topic",
"topicType": "msg",


"x": 550,
"y": 540,
"wires": [
[
"984b01b5c189b5cf"
]
]
},
{
"id": "21d7b6adb780b000",
"type": "mqtt out",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/ac_temp",
"qos": "",
"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "10e78a89.5b4fd5",
"x": 780,
"y": 480,
"wires": []
},
{
"id": "e03fcfbb278754e5",
"type": "mqtt out",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/ac_switch",
"qos": "",
"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "10e78a89.5b4fd5",
"x": 740,
"y": 420,
"wires": []
},
{
"id": "b5c2a3c36530954b",
"type": "mqtt out",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/ac_heat",
"qos": "",


"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "10e78a89.5b4fd5",
"x": 740,
"y": 360,
"wires": []
},
{
"id": "984b01b5c189b5cf",
"type": "mqtt out",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/ac_cool",
"qos": "",
"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "10e78a89.5b4fd5",
"x": 740,
"y": 540,
"wires": []
},
{
"id": "a9332b6c1f34b480",
"type": "ui_switch",
"z": "283dfd7caffc1729",
"name": "",
"label": "Auto",
"tooltip": "",
"group": "39645e65241f69c9",
"order": 2,
"width": 0,
"height": 0,
"passthru": true,
"decouple": "false",
"topic": "topic",
"topicType": "msg",
"style": "",
"onvalue": "on",
"onvalueType": "str",
"onicon": "",
"oncolor": "",
"offvalue": "off",
"offvalueType": "str",


"officon": "",
"offcolor": "",
"animate": false,
"className": "",
"x": 530,
"y": 300,
"wires": [
[
"bde9666bf80fe826"
]
]
},
{
"id": "bde9666bf80fe826",
"type": "mqtt out",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/ac_auto",
"qos": "",
"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "10e78a89.5b4fd5",
"x": 740,
"y": 300,
"wires": []
},
{
"id": "38648473f0e8dd16",
"type": "mqtt in",
"z": "283dfd7caffc1729",
"name": "",
"topic": "michos_test/gas_alarm",
"qos": "2",
"datatype": "auto-detect",
"broker": "10e78a89.5b4fd5",
"nl": false,
"rap": true,
"rh": 0,
"inputs": 0,
"x": 420,
"y": 1060,
"wires": [
[
"f2f0a56e0d5f8e39"
]
]
},

```
    {
        "id": "6409f207576abfac",
        "type": "mqtt in",
        "z": "283dfd7caffc1729",
        "name": "",
        "topic": "michos_test/fire_alarm",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 420,
        "y": 1120,
        "wires": [
            [
                "cfc8325a9bad4101"
            ]
        ]
    },
    {
        "id": "f2f0a56e0d5f8e39",
        "type": "remote-notification",
        "z": "283dfd7caffc1729",
        "confignode": "bd7da532ce73cef0",
        "name": "Gas ALARM",
        "notificationTitle": "GAS!",
        "notificationTitleType": "str",
        "notificationBody": "The house could be exposed to gas or chemicals",
        "notificationBodyType": "str",
        "notificationSound": "dingdingding",
        "notificationSoundComputed": "payload.sound",
        "notificationSoundComputedType": "msg",
        "output": 1,
        "x": 630,
        "y": 1060,
        "wires": [
            []
        ]
    },
    {
        "id": "cfc8325a9bad4101",
        "type": "remote-notification",
        "z": "283dfd7caffc1729",
        "confignode": "bd7da532ce73cef0",
        "name": "Fire Alarm",
        "notificationTitle": "FIRE!",
        "notificationTitleType": "msg",
        "notificationBody": "Your house could be on fire! Seek a fire department",
        "notificationBodyType": "str",
```

```
            "notificationSound": "dingdingding",
            "notificationSoundComputed": "payload.sound",
            "notificationSoundComputedType": "msg",
            "output": 1,
            "x": 630,
            "y": 1120,
            "wires": [
                []
            ]
        },
        {
            "id": "81d248d94ef01411",
            "type": "mqtt in",
            "z": "283dfd7caffc1729",
            "name": "",
            "topic": "michos_test/pet_near_door",
            "qos": "2",
            "datatype": "auto-detect",
            "broker": "10e78a89.5b4fd5",
            "nl": false,
            "rap": true,
            "rh": 0,
            "inputs": 0,
            "x": 410,
            "y": 1020,
            "wires": [
                [
                    "d6e6d32f41181589"
                ]
            ]
        },
        {
            "id": "d6e6d32f41181589",
            "type": "remote-notification",
            "z": "283dfd7caffc1729",
            "confignode": "bd7da532ce73cef0",
            "name": "Pet walk",
            "notificationTitle": "PET",
            "notificationTitleType": "str",
            "notificationBody": "Your pet is near the door and might need a walk",
            "notificationBodyType": "str",
            "notificationSound": "pristine",
            "notificationSoundComputed": "payload.sound",
            "notificationSoundComputedType": "msg",
            "output": 1,
            "x": 700,
            "y": 1020,
            "wires": [
                []
            ]
        },
```

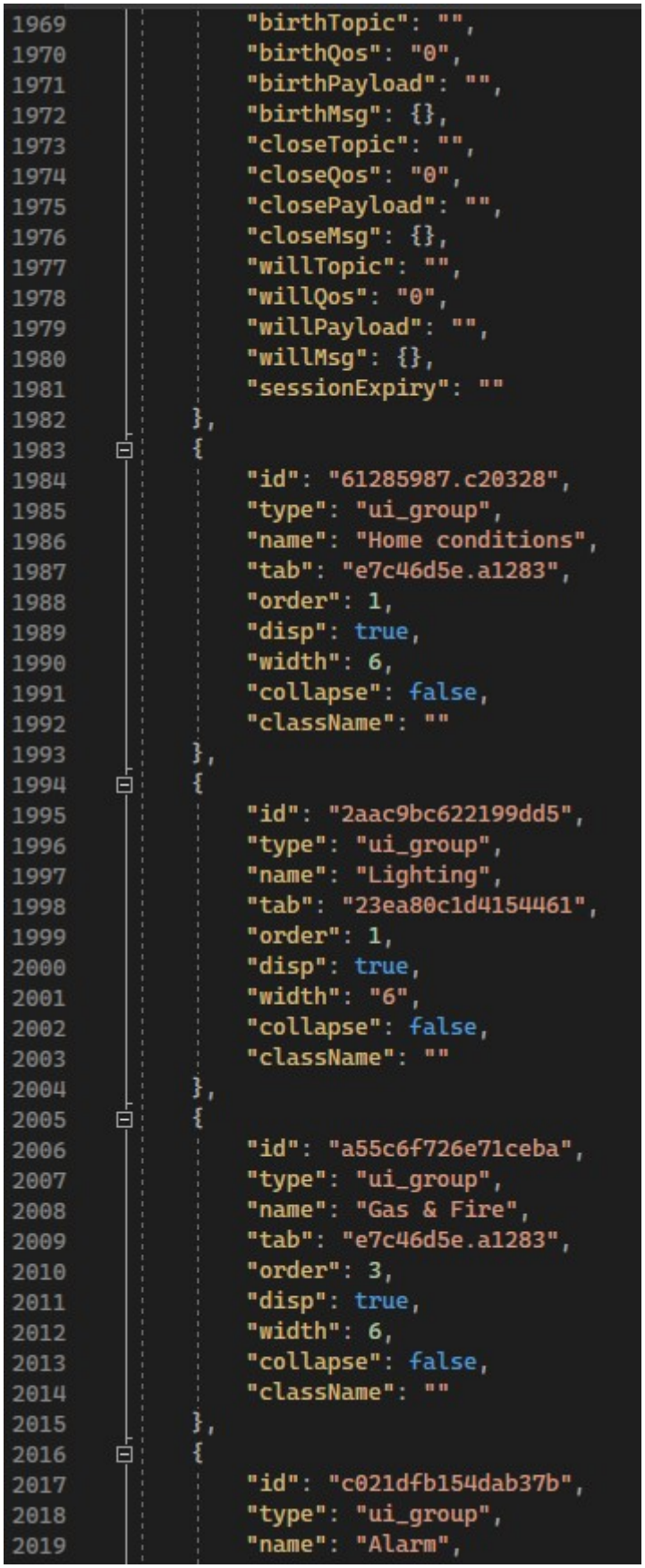
```
    {
        "id": "f93608ef6c5683d1",
        "type": "mqtt in",
        "z": "283dfd7caffc1729",
        "name": "",
        "topic": "michos_test/pet_door_open",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 420,
        "y": 1200,
        "wires": [
            [
                "f2421df498e1f1fb"
            ]
        ]
    },
    {
        "id": "f2421df498e1f1fb",
        "type": "remote-notification",
        "z": "283dfd7caffc1729",
        "confignode": "bd7da532ce73cef0",
        "name": "PET WARNING",
        "notificationTitle": "WARNING",
        "notificationTitleType": "str",
        "notificationBody": "Your pet is near an open door",
        "notificationBodyType": "str",
        "notificationSound": "dingdingding",
        "notificationSoundComputed": "payload.sound",
        "notificationSoundComputedType": "msg",
        "output": 1,
        "x": 700,
        "y": 1200,
        "wires": [
            []
        ]
    },
    {
        "id": "28970dc189832452",
        "type": "comment",
        "z": "283dfd7caffc1729",
        "name": "Remote Notifications",
        "info": "",
        "x": 450,
        "y": 1280,
        "wires": []
    },
```

```
    {
        "id": "2ee9e6f1401d9d64",
        "type": "openweathermap",
        "z": "283dfd7caffc1729",
        "name": "Node-RED",
        "wtype": "current",
        "lon": "",
        "lat": "",
        "city": "Thessaloniki",
        "country": "Greece",
        "language": "en",
        "x": 1370,
        "y": 180,
        "wires": [
            [
                "7e9ba3dd4a1314a0",
                "21c64f91409f2c7e",
                "059a76a40d33d2c7",
                "03bc278a321887e8",
                "0aca20814c42777a",
                "4de6455a43b075b5"
            ]
        ]
    },
    {
        "id": "3386ba1a758b444f",
        "type": "inject",
        "z": "283dfd7caffc1729",
        "name": "",
        "props": [
            {
                "p": "payload"
            },
            {
                "p": "topic",
                "vt": "str"
            }
        ],
        "repeat": "3600",
        "crontab": "",
        "once": true,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 1210,
        "y": 180,
        "wires": [
            [
                "2ee9e6f1401d9d64"
            ]
```


]
},
{
"id": "7e9ba3dd4a1314a0",
"type": "ui_text",
"z": "283dfd7caffc1729",
"group": "bc6f21e297bd8193",
"order": 1,
"width": 0,
"height": 0,
"name": "",
"label": "City",
"format": "{{location.city}}",
"layout": "row-spread",
"className": "",
"x": 1730,
"y": 100,
"wires": []
},
{
"id": "21c64f91409f2c7e",
"type": "ui_text",
"z": "283dfd7caffc1729",
"group": "bc6f21e297bd8193",
"order": 3,
"width": 0,
"height": 0,
"name": "",
"label": "Details",
"format": "{{payload.detail}}",
"layout": "row-spread",
"className": "",
"x": 1730,
"y": 180,
"wires": []
},
{
"id": "059a76a40d33d2c7",
"type": "ui_text",
"z": "283dfd7caffc1729",
"group": "bc6f21e297bd8193",
"order": 2,
"width": 0,
"height": 0,
"name": "",
"label": "Weather",
"format": "{{payload.weather}}",
"layout": "row-spread",
"className": "",
"x": 1720,
"y": 140,

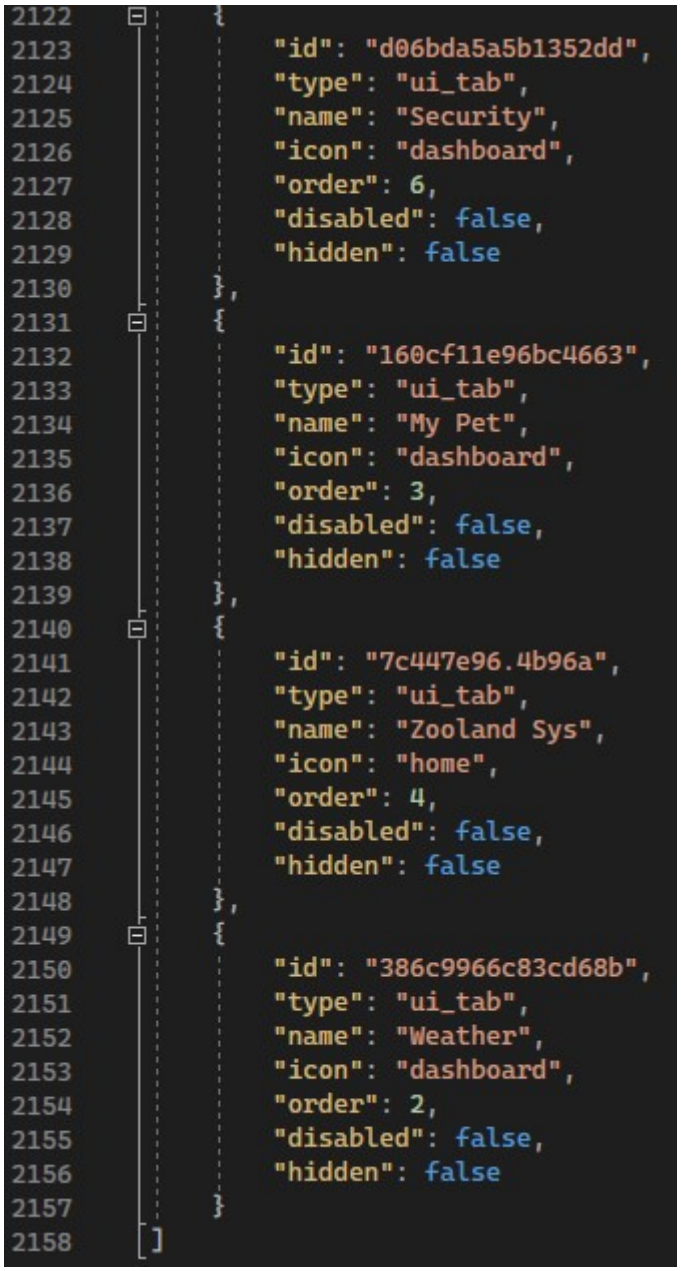

"wires": []
},
{
"id": "03bc278a321887e8",
"type": "ui_gauge",
"z": "283dfd7caffc1729",
"name": "",
"group": "bc6f21e297bd8193",
"order": 4,
"width": 0,
"height": 0,
"gtype": "gage",
"title": "Temperature",
"label": "Celsius",
"format": "{{payload.tempc}}",
"min": "-40",
"max": "60",
"colors": [
"#00fbff",
"#00e645",
"#fa0000"
],
"seg1": "",
"seg2": "",
"className": "",
"x": 1730,
"y": 240,
"wires": []
},
{
"id": "0aca20814c42777a",
"type": "ui_text",
"z": "283dfd7caffc1729",
"group": "1b99b8eee277865a",
"order": 1,
"width": 0,
"height": 0,
"name": "",
"label": "Wind Speed",
"format": "{{data.wind.speed}}",
"layout": "row-spread",
"className": "",
"x": 1750,
"y": 280,
"wires": []
},
{
"id": "4de6455a43b075b5",
"type": "ui_text",
"z": "283dfd7caffc1729",
"group": "1b99b8eee277865a",


"order": 2,
"width": 0,
"height": 0,
"name": "",
"label": "Degrees",
"format": "{{data.wind.deg}}",
"layout": "row-spread",
"className": "",
"x": 1720,
"y": 340,
"wires": []
},
{
"id": "e714319df6d3c36d",
"type": "debug",
"z": "283dfd7caffc1729",
"name": "debug 1",
"active": true,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"statusVal": "",
"statusType": "auto",
"x": 1400,
"y": 120,
"wires": []
},
{
"id": "8a9b3c21353a372d",
"type": "ui_spacer",
"z": "283dfd7caffc1729",
"name": "spacer",
"group": "a55c6f726e71ceba",
"order": 2,
"width": 6,
"height": 1
},
{
"id": "10e78a89.5b4fd5",
"type": "mqtt-broker",
"name": "",
"broker": "localhost",
"port": "1883",
"clientid": "",
"autoConnect": true,
"usetls": false,
"protocolVersion": "4",
"keepalive": "60",
"cleansession": true,


"birthTopic": "",
"birthQos": "0",
"birthPayload": "",
"birthMsg": {},
"closeTopic": "",
"closeQos": "0",
"closePayload": "",
"closeMsg": {},
"willTopic": "",
"willQos": "0",
"willPayload": "",
"willMsg": {},
"sessionExpiry": ""
},
{
"id": "61285987.c20328",
"type": "ui_group",
"name": "Home conditions",
"tab": "e7c46d5e.a1283",
"order": 1,
"disp": true,
"width": 6,
"collapse": false,
"className": ""
},
{
"id": "2aac9bc622199dd5",
"type": "ui_group",
"name": "Lighting",
"tab": "23ea80c1d4154461",
"order": 1,
"disp": true,
"width": "6",
"collapse": false,
"className": ""
},
{
"id": "a55c6f726e71ceba",
"type": "ui_group",
"name": "Gas & Fire",
"tab": "e7c46d5e.a1283",
"order": 3,
"disp": true,
"width": 6,
"collapse": false,
"className": ""
},
{
"id": "c021dfb154dab37b",
"type": "ui_group",
"name": "Alarm",

```
            "tab": "d06bda5a5b1352dd",
            "order": 1,
            "disp": true,
            "width": "6",
            "collapse": false,
            "className": ""
        },
        {
            "id": "bd7da532ce73cef0",
            "type": "remote-config",
            "name": "My Home",
            "host": "localhost",
            "protocol": "http",
            "port": "1880",
            "baseurl": "/ui",
            "instancehash": "7et42uw1u9vt6rmqdk8der4ai5e9o5griqqox22ganc70xwxlsez1yk6h
            "server": "nodered04.remote-red.com",
            "region": "de"
        },
        {
            "id": "72c8164bb9389191",
            "type": "ui_group",
            "name": "Group 1",
            "tab": "160cf11e96bc4663",
            "order": 1,
            "disp": false,
            "width": "6",
            "collapse": false,
            "className": ""
        },
        {
            "id": "2d0c2ebd8915f48e",
            "type": "ui_group",
            "name": "",
            "tab": "160cf11e96bc4663",
            "order": 2,
            "disp": true,
            "width": "6",
            "collapse": false,
            "className": ""
        },
        {
            "id": "c982fbb8.1deb38",
            "type": "ui_group",
            "name": "hidden_group",
            "tab": "7c447e96.4b96a",
            "order": 1,
            "disp": false,
            "width": "6",
            "collapse": false
        },
```


{
"id": "39645e65241f69c9",
"type": "ui_group",
"name": "A/C",
"tab": "e7c46d5e.a1283",
"order": 2,
"disp": true,
"width": 6,
"collapse": false,
"className": ""
},
{
"id": "bc6f21e297bd8193",
"type": "ui_group",
"name": "Group 1",
"tab": "386c9966c83cd68b",
"order": 1,
"disp": false,
"width": "6",
"collapse": false,
"className": ""
},
{
"id": "1b99b8eee277865a",
"type": "ui_group",
"name": "Group 2",
"tab": "386c9966c83cd68b",
"order": 2,
"disp": false,
"width": "6",
"collapse": false,
"className": ""
},
{
"id": "e7c46d5e.a1283",
"type": "ui_tab",
"name": "Home",
"icon": "dashboard",
"order": 1,
"disabled": false,
"hidden": false
},
{
"id": "23ea80c1d4154461",
"type": "ui_tab",
"name": "Devices",
"icon": "dashboard",
"order": 5,
"disabled": false,
"hidden": false
},


{
"id": "d06bda5a5b1352dd",
"type": "ui_tab",
"name": "Security",
"icon": "dashboard",
"order": 6,
"disabled": false,
"hidden": false
},
{
"id": "160cf11e96bc4663",
"type": "ui_tab",
"name": "My Pet",
"icon": "dashboard",
"order": 3,
"disabled": false,
"hidden": false
},
{
"id": "7c447e96.4b96a",
"type": "ui_tab",
"name": "Zooland Sys",
"icon": "home",
"order": 4,
"disabled": false,
"hidden": false
},
{
"id": "386c9966c83cd68b",
"type": "ui_tab",
"name": "Weather",
"icon": "dashboard",
"order": 2,
"disabled": false,
"hidden": false
}
]