|  |
| --- |
| Subject: |
| Application to robotics & IOT |

|  |
| --- |
| Document title: |
| **Automated Greenhouse System (Mockup)** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prepared by: |  |  |  |  |
| Name |  | M&M |  | 15/02/2023 |
| Course |  | 4ºA |  | Date |
|  |  |  |  |  |
| Reviewed: |  |  |  |  |
| Name |  |  |  |  |
| Prepared by: |  |  |  | Date |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| File name: |  | Date: |  | Edition: |  | Nº pages: |
| **M&M.pdf** |  | **15/02/2023** |  | **1** |  | **10** |

**Documentary control**

|  |  |
| --- | --- |
| Authors | Miguel Vilanova Fenollar  Miguel Monge Rodríguez |
| Project Name | Automated Greenhouse System (mockup) |

**Version control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Description | Date |
| 0.1 | Template: Cover, Documentary Control | First version of the document | 15/02/2023 |
| 0.2 | Introduction | Added the Introduction, objectives, motivation details | 18/02/2023 |
| 0.3 | Planification | Added the Planification details | 19/02/2023 |
| 0.4 | Design | Added the Design phase details | 28/02/2023 |
| 0.5 | SW Design | Explanation | 08/03/2023 |
|  |  |  |  |

\* N/A = Not applicable

Index

[Introduction 4](#_Toc128950792)

[Planification 5](#_Toc128950793)

[Determine the requirements 6](#_Toc128950794)

[URD 6](#_Toc128950795)

[Risk planning 7](#_Toc128950796)

[Hardware 11](#_Toc128950797)

[Select sensors and actuators 11](#_Toc128950798)

[Design physical model 11](#_Toc128950799)

[Software 12](#_Toc128950800)

[Design NodeMCU interface 12](#_Toc128950801)

[Develop control algorithm 12](#_Toc128950802)

[Develop the software 12](#_Toc128950803)

[References 14](#_Toc128950804)

Figures

[1: Gantt Project, file attached. 5](#_Toc128952299)

[2: NodeMCU e-diagram 9](#_Toc128952300)

[3: Node Red interface example 10](#_Toc128952301)

[4: MQTT Explorer (MQTT broker portable) 10](#_Toc128952302)

Tables

[i: URD 6](#_Toc128952303)

[ii: Risk Planning 7](#_Toc128952304)

# Introduction

We intend to create and implement an automated greenhouse system that optimizes plant growth, maximizes resource efficiency, and promotes sustainability. By integrating sensors, actuators, and control algorithms, we aim to create an intelligent system that can adjust the environmental conditions in real-time to meet the changing needs of the plants.

1. Sensors collect data: The sensors placed inside the greenhouse collect data on the environmental conditions such as temperature, humidity, light, soil moisture, and CO2 levels.
2. NodeMCU collects and sends data: The NodeMCU board, which is connected to the sensors, collects the data and sends it to the cloud-based platform, Node-RED, using Wi-Fi connectivity.
3. Node-RED processes the data: Node-RED is a visual programming tool that can process the data received from NodeMCU in real-time. It can analyze the data and trigger certain actions based on pre-defined conditions. For example, if the temperature in the greenhouse is too high, Node-RED can activate the cooling system.
4. Data storage and analysis: The processed data from Node-RED is then stored on a Raspberry Pi. The Raspberry Pi can be used to analyze and mine the data for valuable insights, such as identifying patterns in the growth of the plants, optimizing resource usage, and predicting future growth.

By using NodeMCU, Node-RED, and Raspberry Pi, you can create an automated greenhouse system that is both intelligent and efficient. The system can help you optimize the growth conditions for plants, reduce resource usage, and increase productivity.

Ultimately, our goal is to create a sustainable and efficient automated greenhouse system that can serve as a model for future agricultural practices. We believe that by combining technology, innovation, and ethics, we can create a better world for both plants and humans.

# Planification

* Design: This phase involves defining the objectives of the project and designing the hardware and software components. It starts on 13/2/23 and ends on 10/3/23.
* Building and Testing: This phase involves building the physical model of the automated greenhouse system, integrating the sensors and actuators, and programming the software. It starts on 13/3/23 and ends on 21/4/23.
* Sensor Testing and Feedback: This phase involves testing the sensors and collecting feedback through Telegram. It starts on 10/4/23 and ends on 21/4/23.
* Project Documentation: This phase involves documenting the project history and the state of the art of the automated greenhouse system. It starts on 24/4/23 and ends on 5/5/23.
* Final Assignment: This phase involves the final assignment submission. It takes place on 8/5/23.

Gráfico, Gráfico en cascada

Descripción generada automáticamente

1: Gantt Project, file attached.

## Determine the requirements

We can use humidity sensors to monitor the soil moisture levels of the plants. This can help to determine when to water the plants and avoid overwatering or underwatering.

Use the temperature sensor to monitor the temperature inside the greenhouse. This can be used to adjust the temperature based on the needs of the plants and optimize growth conditions.

Use the NodeMCU board to collect data from the sensors and send it to the cloud-based platform, Node-RED. We can process the data and trigger certain actions based on pre-defined conditions.

We can use a Raspberry Pi or a cloud-based database to store and analyze the data collected from the sensors. This can help to identify patterns and trends in plant growth and optimize resource usage.

Finally, we need humidity sensors to monitor the humidity levels inside the greenhouse. To adjust the humidity based on the needs of the plants and prevent plant diseases or pests.

### URD

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Description | Priority |
| 0 | Temperature Sensor | Measures temperature inside the greenhouse | High |
| 1 | Humidity Sensor | Measures humidity inside the greenhouse | High |
| 2 | Soil Moisture Sensor | Measures soil moisture levels of the plants | High |
| 3 | CO2 Sensor | Measures CO2 levels inside the greenhouse | Low |
| 4 | Lighting Controls | Adjusts the intensity and duration of light for the plants | High |
| 5 | Irrigation System | Delivers water to the plants based on their needs | Medium |
| 6 | Ventilation System | Regulates air flow and temperature inside the greenhouse | Medium |
| 7 | NodeMCU + Wi-Fi Mod | Collects data from the sensors and sends it to the cloud-based platform | High |
| 8 | Raspberry Pi or Cloud | Stores and analyzes the data collected from the sensors | Medium |
| 9 | Node-RED Platform | Processes the data received from the sensors and triggers certain actions based on pre-defined conditions | High |

i: URD

### Risk planning

It's important to anticipate potential difficulties and develop contingency plans to address them.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Risk | Contingency Plan | Difficulty |
| 0 | Sensor malfunction or inaccurate readings | Regularly calibrate and test the sensor, have backup sensors available | Low |
| 1 | Sensor malfunction or inaccurate readings | Regularly calibrate and test the sensor, have backup sensors available | Low |
| 2 | Sensor malfunction or inaccurate readings | Regularly calibrate and test the sensor, have backup sensors available | Low |
| 3 | System failure or power outage | Have a backup power source available, manually adjust the lighting if necessary | Low |
| 4 | System failure or overwatering/underwatering | Backup irrigation components available, manually water the plants if necessary | High |
| 5 | System failure or inadequate air flow | Backup ventilation components available, monitor the temperature air flow manually | High |
| 6 | Wi-Fi or connectivity issues | Monitor connectivity, have backup communication methods available | Low |
| 7 | Data loss or corruption | Regularly back up the data, have backup storage solutions available | High |
| 8 | System failure or programming errors | Regularly test and debug the system, have backup programming solutions available | Low |
| 9 | Sensor malfunction or inaccurate readings | Regularly calibrate and test the sensor, have backup sensors available | Medium |

ii: Risk Planning

Design

## Hardware

### Select sensors and actuators

Based on the plant requirements, we will select the appropriate sensors and actuators. For example, we might use temperature and humidity sensors, irrigation actuators, lighting controls.

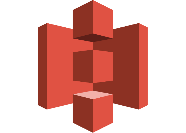
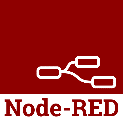
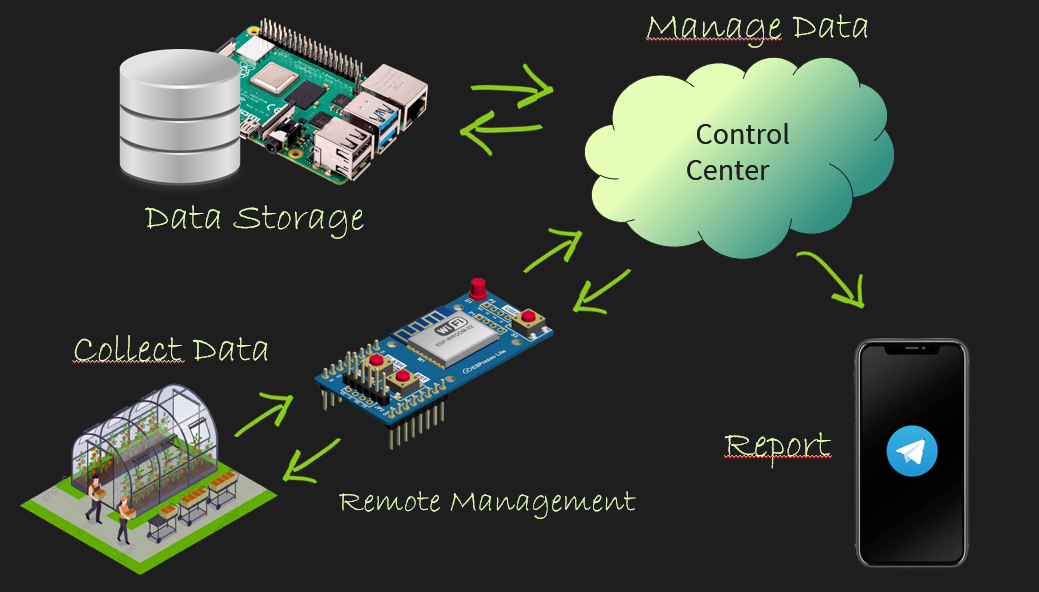
### Design physical model

We will create a 3D model of the automated greenhouse system, including the layout of the sensors and actuators, and the positioning of the plants. This will help us visualize the system and ensure that it meets the objectives and plant requirements.

## Software

SW processes:

1. NodeMCU acts as the interface between the sensors and actuators.
2. Sensors collect data on the conditions inside the greenhouse and send it to the NodeMCU.
3. The NodeMCU then sends the data to Node-RED, which processes the information and can trigger certain actions based on predetermined conditions.
4. The processed data from Node-RED is then stored on a Raspberry Pi or cloud storage, where it can be analyzed and mined for valuable insights.



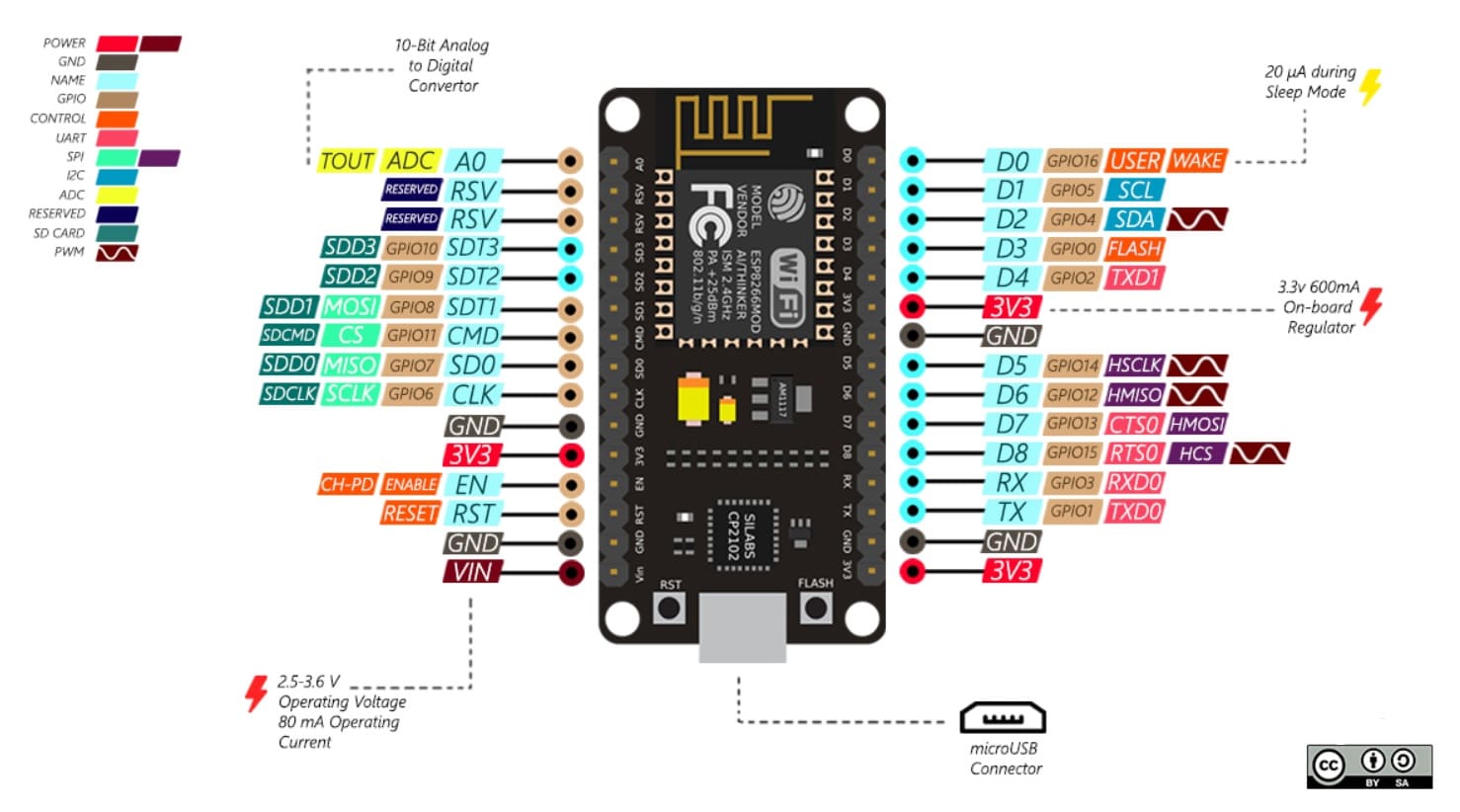
*By following these steps, we can ensure that the design phase of the automated greenhouse system is thorough and well-planned, and that it meets the objectives and plant requirements. This will set us up for success during the building and testing phase.*

### Design NodeMCU interface

We will design the NodeMCU board interface that will collect data from the sensors and control the actuators. This might involve selecting the appropriate Wi-Fi module and programming the board to communicate with the cloud-based platform, Node-RED.

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

The firmware uses the Lua scripting language. It is based on the eLua project and built on the Espressif Non-OS SDK for ESP8266. It uses many open-source projects, such as lua-cjson, and spiffs.

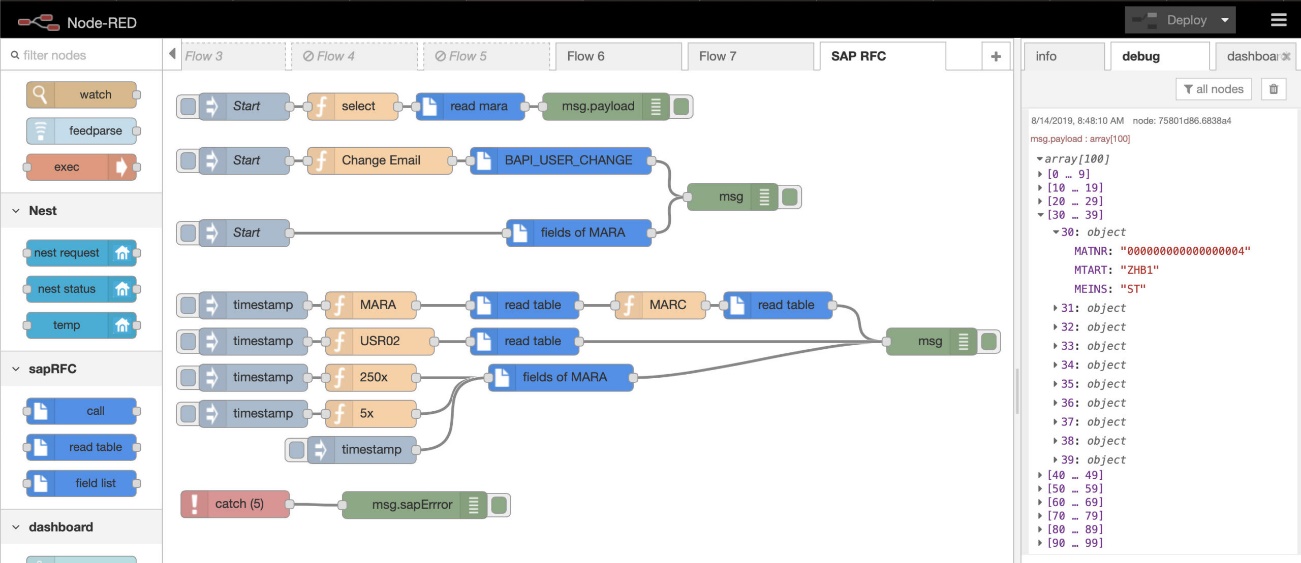


2: NodeMCU e-diagram

Node-RED is a programming tool for wiring together hardware devices, APIs, and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows. Used in a lot with Home Assistant platform projects.

It was developed by IBM and is now an open-source project. It is primarily used to integrate different IoT devices and services and can be used to process and visualize data streams in real time.

Node-RED uses a visual programming language called "flows" to allow users to wire together code blocks, called "nodes", to perform a task. It is often used in conjunction with hardware such as Raspberry Pi and other microcontrollers, as well as cloud-based services like AWS and Azure.



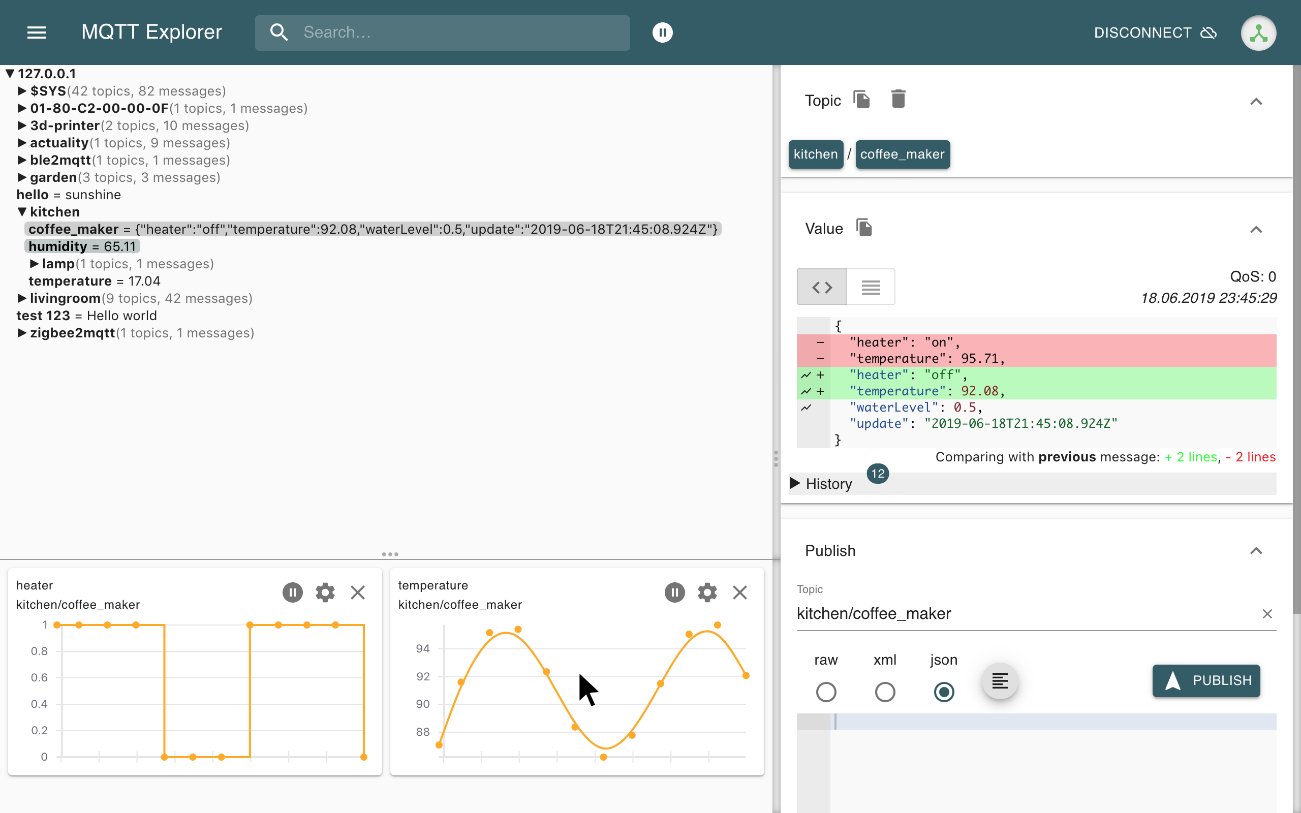
3: Node Red interface example

MQTT (Message Queuing Telemetry Transport) is a lightweight publish-subscribe messaging protocol that is used to send messages between devices. It was designed to be used with low-power devices and constrained networks, such as those found on the Internet of Things (IoT).

In MQTT, there are two main components:

* The message broker is responsible for receiving all messages and distributing them to the clients that are subscribed to the topic.
* The clients can be both publishers and subscribers, meaning they can both send and receive messages.

MQTT is a widely used protocol in the IoT and is supported by many devices and platforms, including Node-RED, which makes it easy to integrate with other systems.



4: MQTT Explorer (MQTT broker portable)

### Develop the software

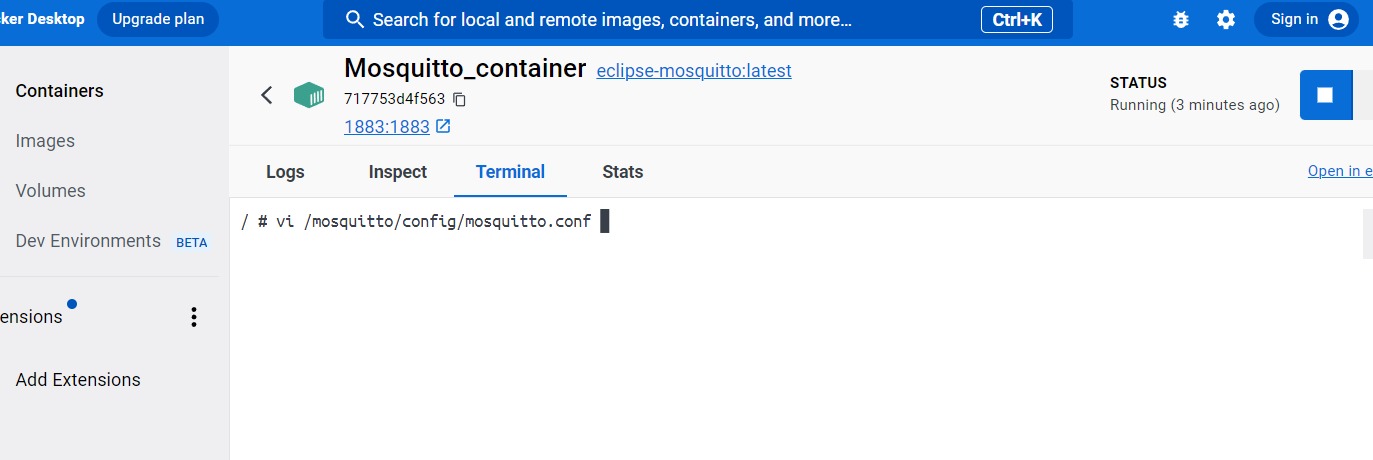
Finally, we will develop the software components, including the Node-RED platform and the data storage and analysis tools. This will allow us to process and analyze the data collected from the sensors and provide insights into the growth and efficiency of the plants.

docker pull eclipse-mosquitto

CMD

#### Setup MQTT server (Docker)

We need to download the latest [eclipse-mosquitto](https://hub.docker.com/_/eclipse-mosquitto) image:



We need to edit the [configuration](https://mosquitto.org/man/mosquitto-conf-5.html) file to allow anonymous connections and listen on port 1883, also we can create a config file and run the docker with that configuration.

Imagen de la pantalla de un celular con letras

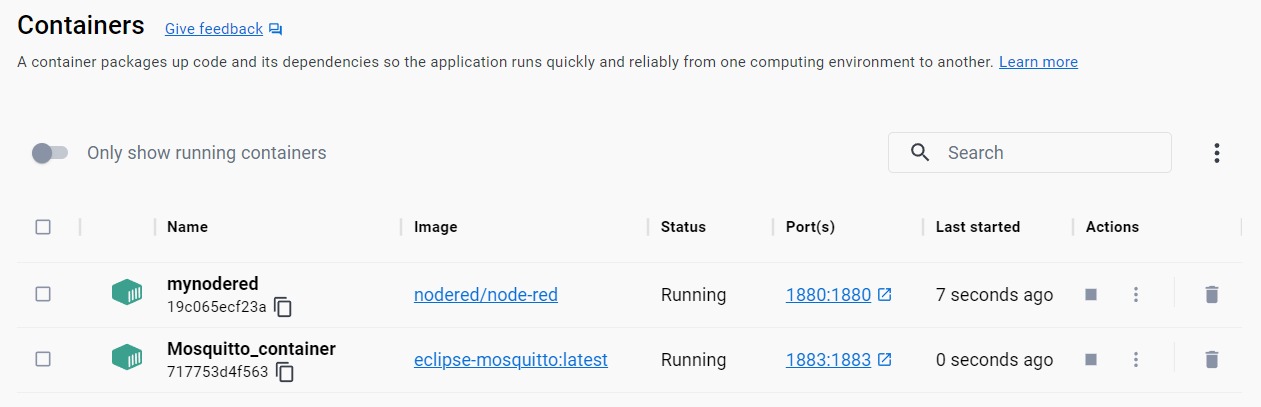
Descripción generada automáticamente con confianza baja

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

docker run -it --name mosquitto -p 1883:1883 -v mosquitto-config:/mosquitto/config eclipse-mosquitto

CMD

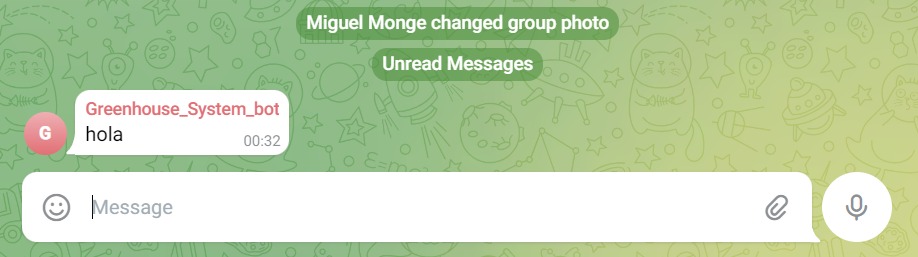
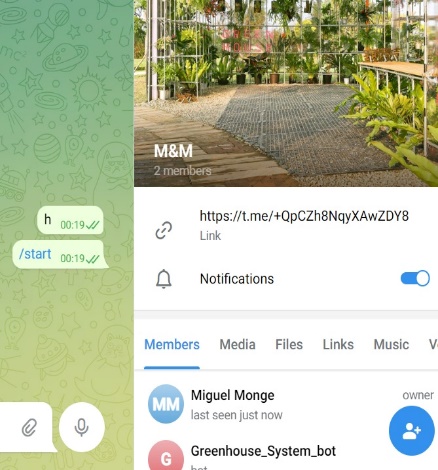
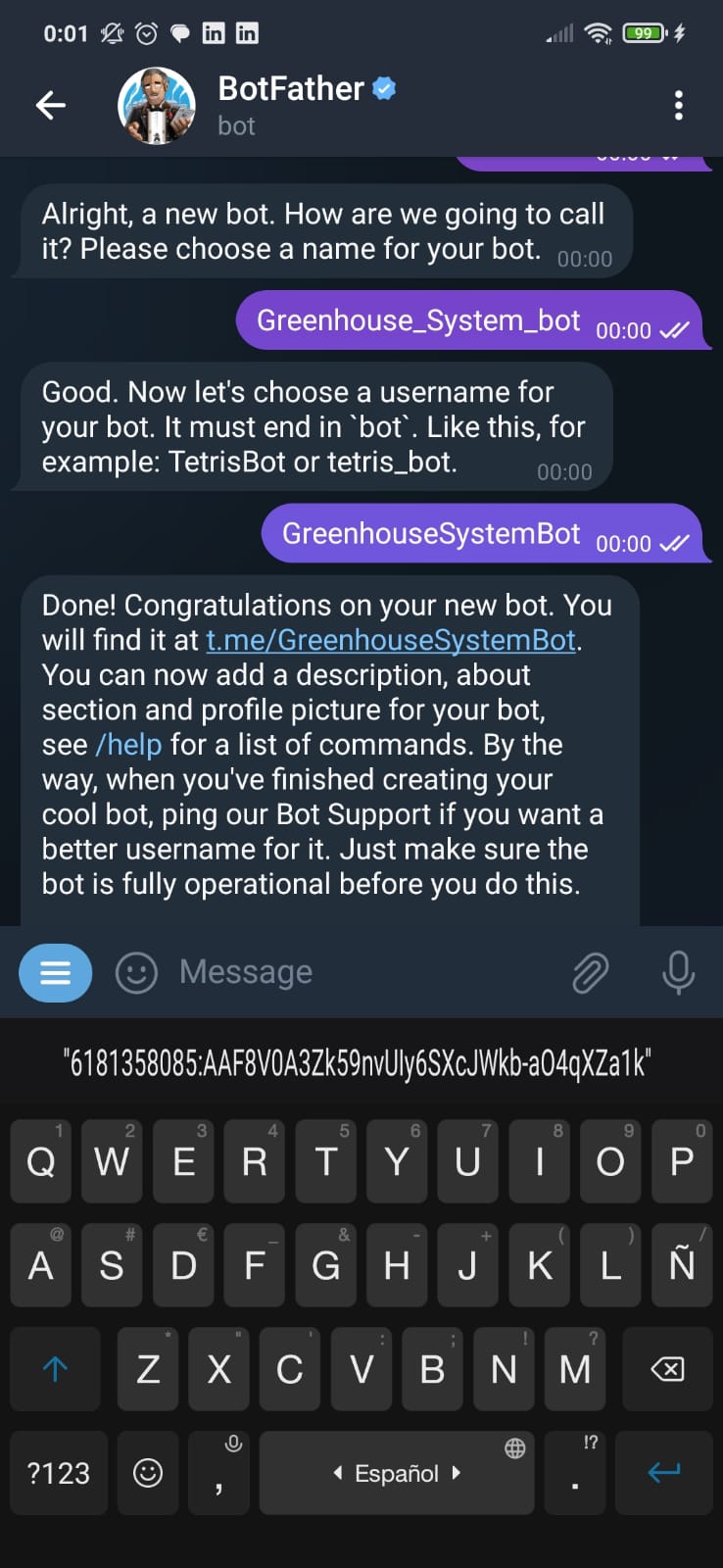
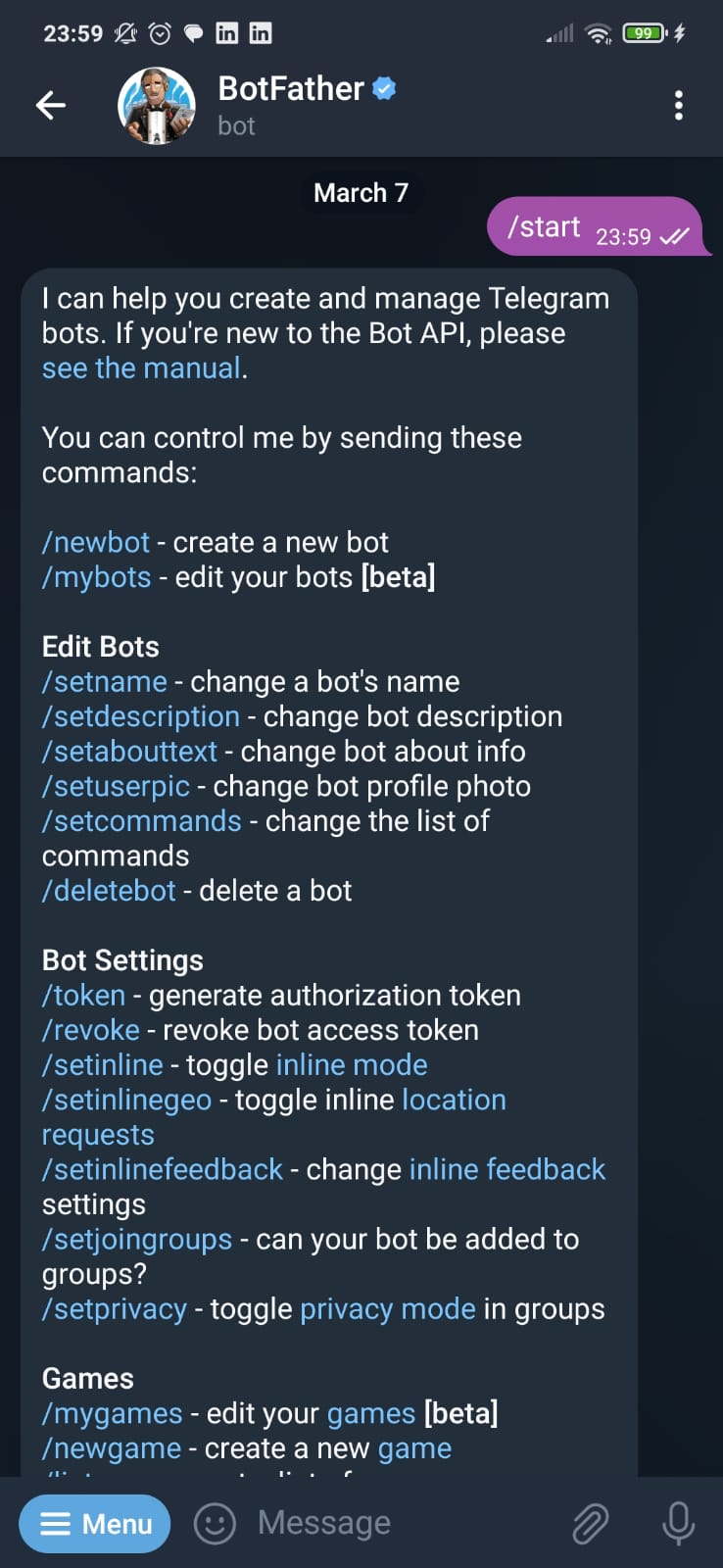


With that done we can connect to the server; we are going to use the portable version of mosquito explorer as said.

#### Setup Telegram Bots

Crafting and configuring bots on Telegram is remarkably straightforward:

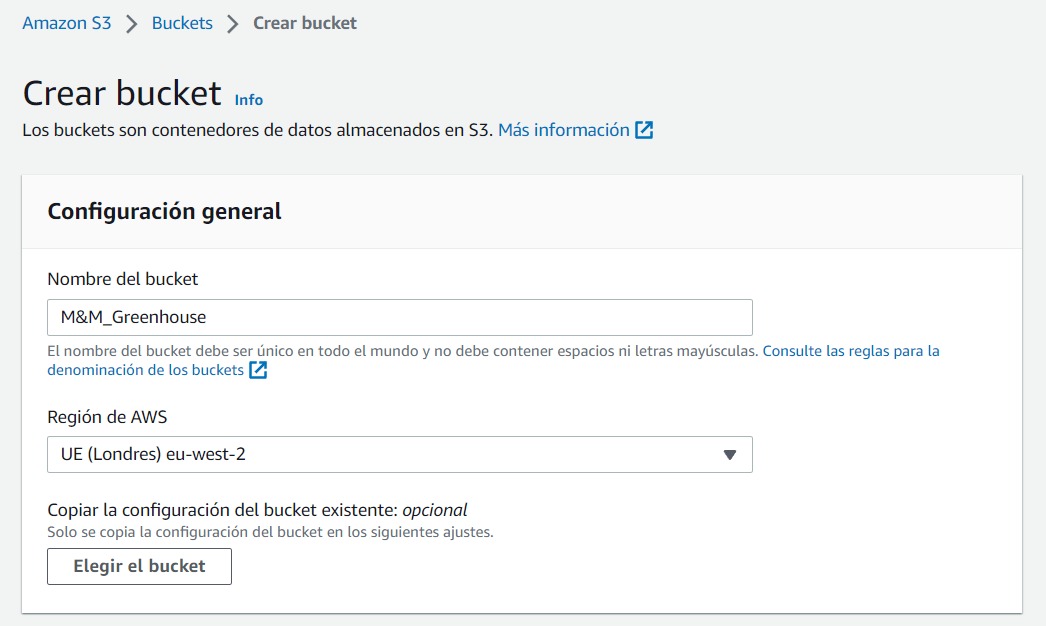
1. Search for BotFather in the search bar.
2. Start a chat with BotFather by clicking on the "Start" button.
3. Send the command "/newbot" to BotFather and follow the prompts to create a new bot, enter the bot’s name: “GreenhouseSystemBot”.
4. Once you have created your bot, BotFather will provide you with a token.
5. We then added the bot to a Group Chat named “M&M”.



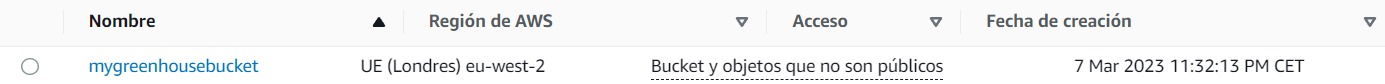
We can add more bots like this to serve additional purposes or functions.

#### Setup AWS S3

We need to create a bucket, a bucket is a cloud-based storage container provided by Amazon Web Services (AWS) for storing and retrieving data.



You can enable versioning, encryption, object level logging, and other options to the bucket but we don’t need it for this project.

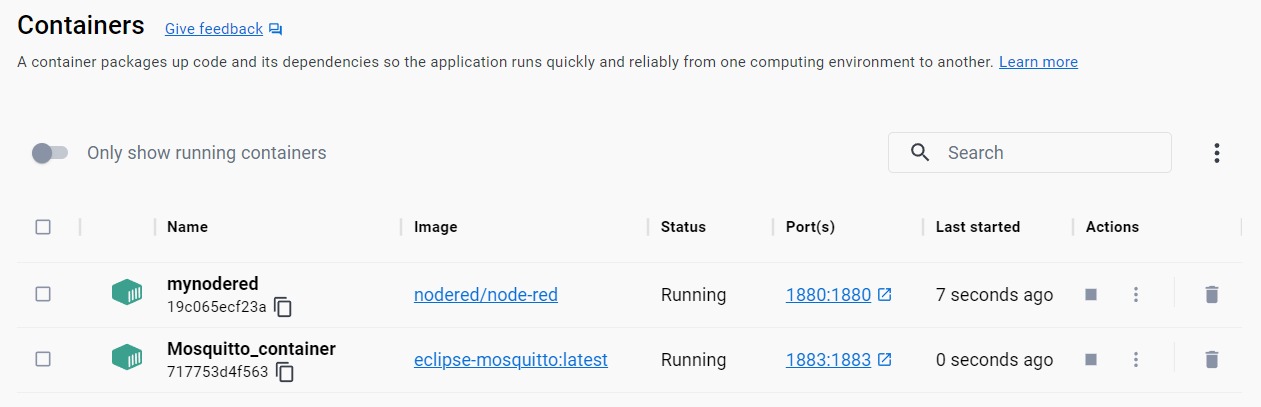


#### Setup Node Red (Docker)

Since we need to create and delete files in a container, we create a data volume within the container. A data volume is a specially designated directory within one or more containers that bypasses the container's union file system and provides a way for persistent data to be stored and shared among containers.

docker run -it -p 1880:1880 -v ~/nodereddata:/data --name mynodered nodered/node-red

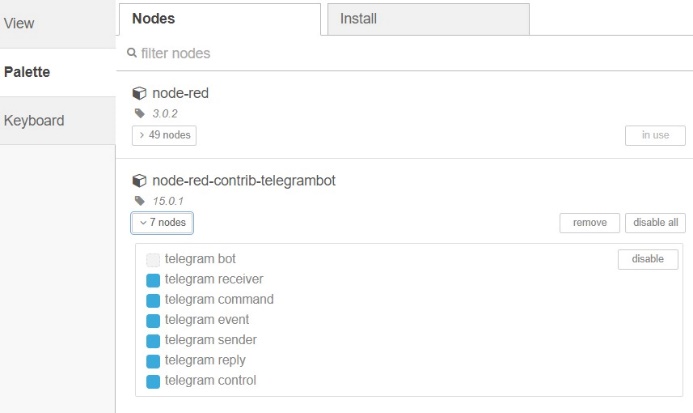
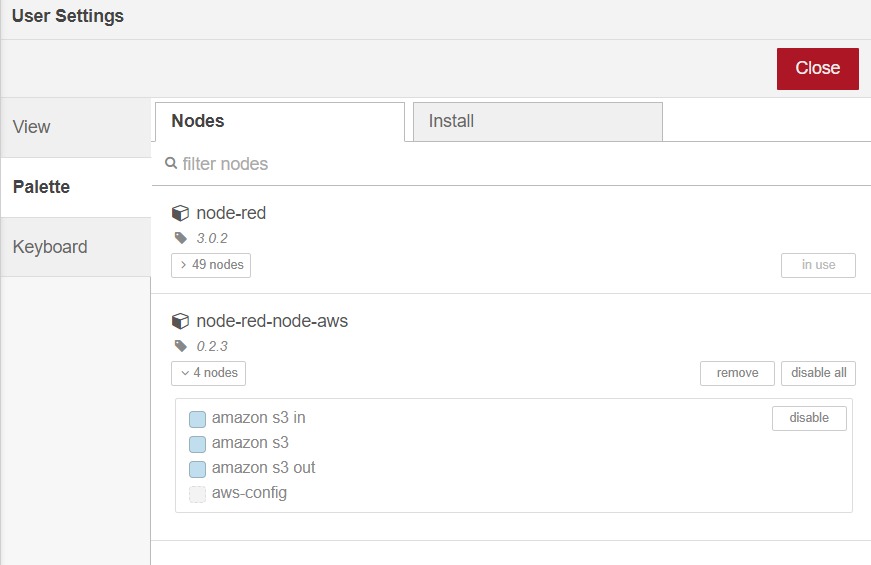
CMD



Now you can easily create files in the data volume inside Node Red container, we will need this to upload logs to AWS S3 bucket (and then maybe process it with EC2).

Click on manage palette and seach for:

node-red-contrib-telegrambot & node-red-node-aws:



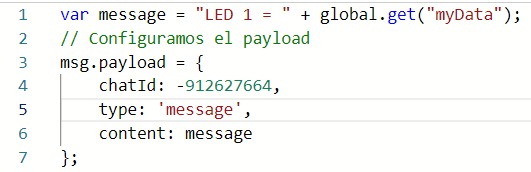
You can also use node-red-contrib-telegrambot-home, is easier to use and adds more functionality.

#### SW test

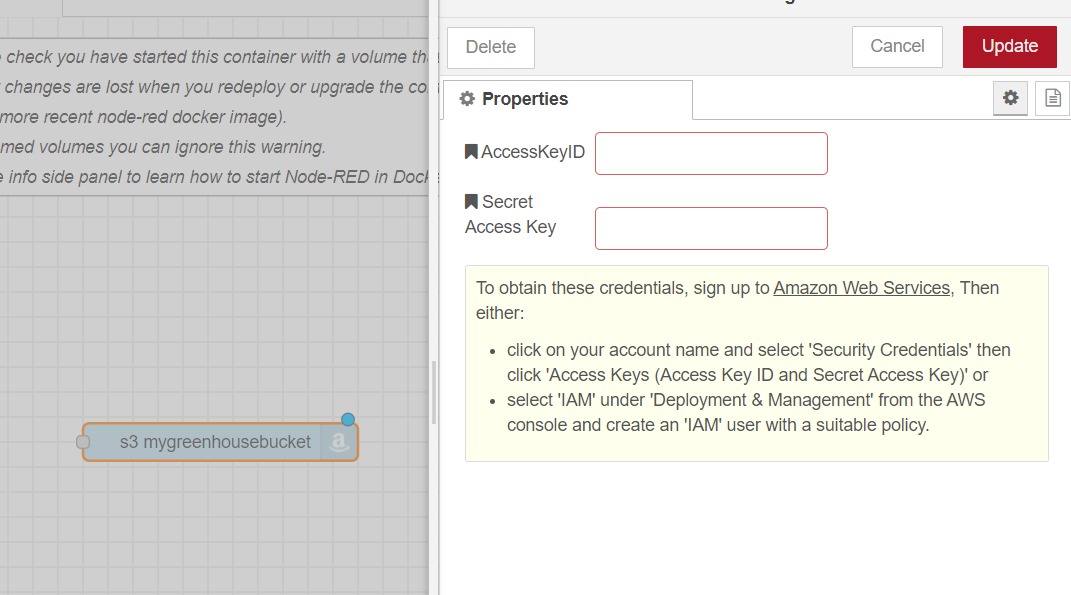
Now it is time to check that is all correct.

1. Connect to MQTT server.
   1. Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

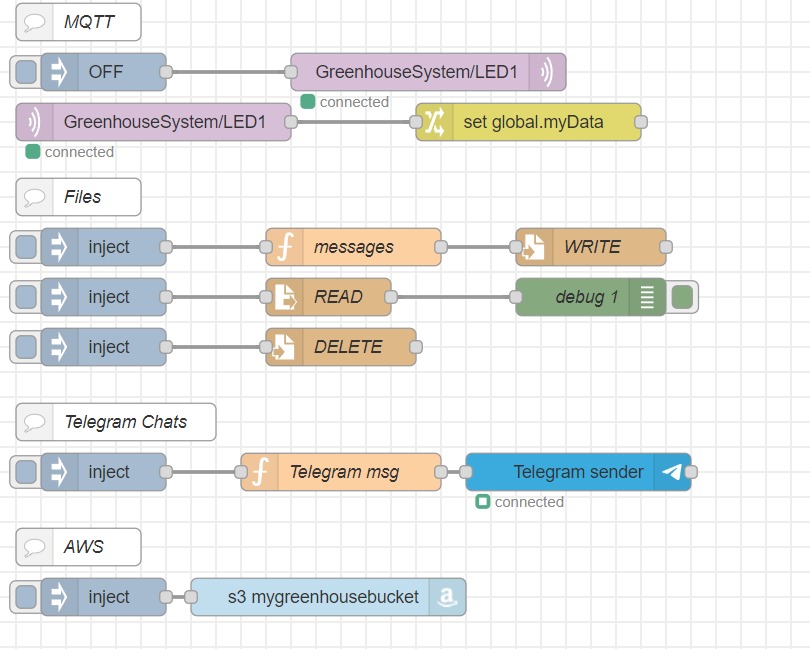
      Descripción generada automáticamenteUse your local Ip, localhost:1883 does not work since is a container.
2. Make sure the telegram bot token and chat IDs are correct.
   1. Parse msg.payload as follows to tell the bot which chat to type in:

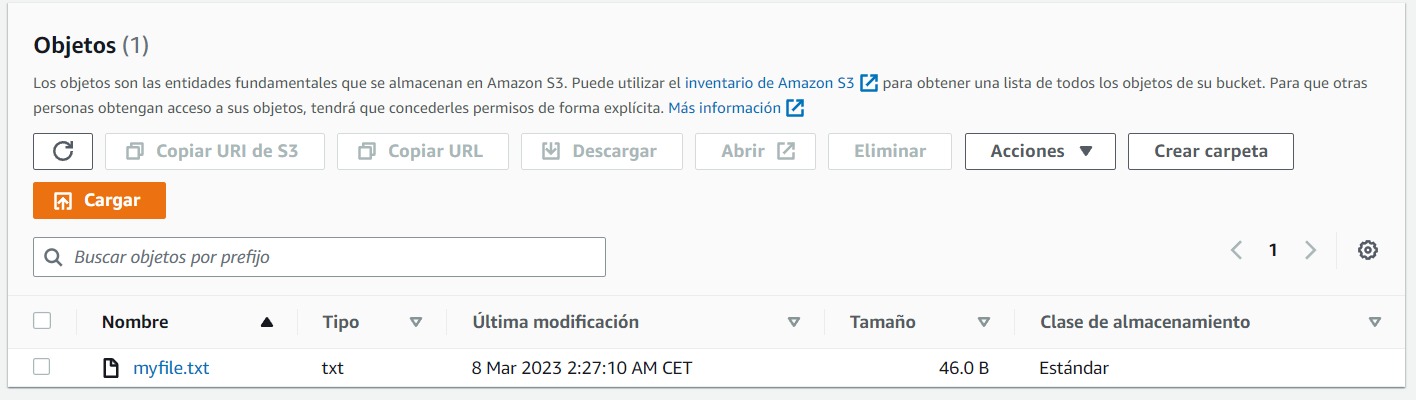


1. Check that S3 bucket is connected:
   1. You will need to create access keys and put them into the node properties.

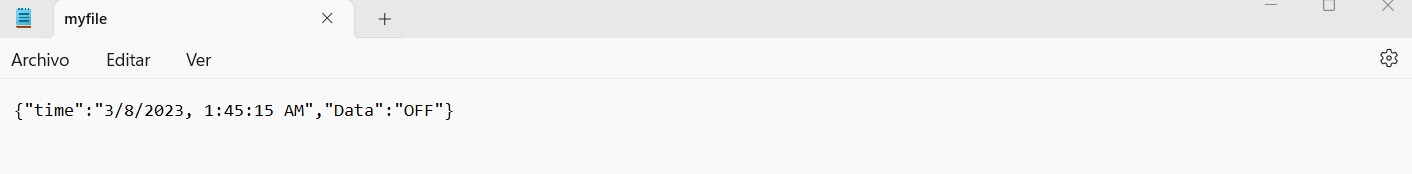


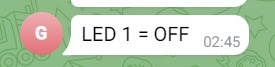
Now you can create a test flow as follows:



As we can see the test file is created in local storage, the data volume we created previously, and now we send that file to the AWS bucket.

Inside this file we can see that the data collected is “OFF” with a timestamp.

This is because we created a topic called LED1 inside GreenhouseSystem general topic that monitors the state of a led in the Arduino board (just test). 

Then we store the string “OFF” as a global or flow variable and we use a javascript function to send it to telegram:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteAnother function appends the msg to a file.txt and store in data volume.

Then this file is sent to S3.

### Develop control algorithm

We will develop the control algorithms that will adjust the environmental conditions based on the data collected from the sensors. This might include using fuzzy logic or other machine learning techniques to optimize resource usage.

# Building and Testing

## Buy & Check

## Building the 3D mockup

## Programming

# Testing

# State of Art (Professional Approach)

Conclusions

# References

**No hay ninguna fuente en el documento actual.**