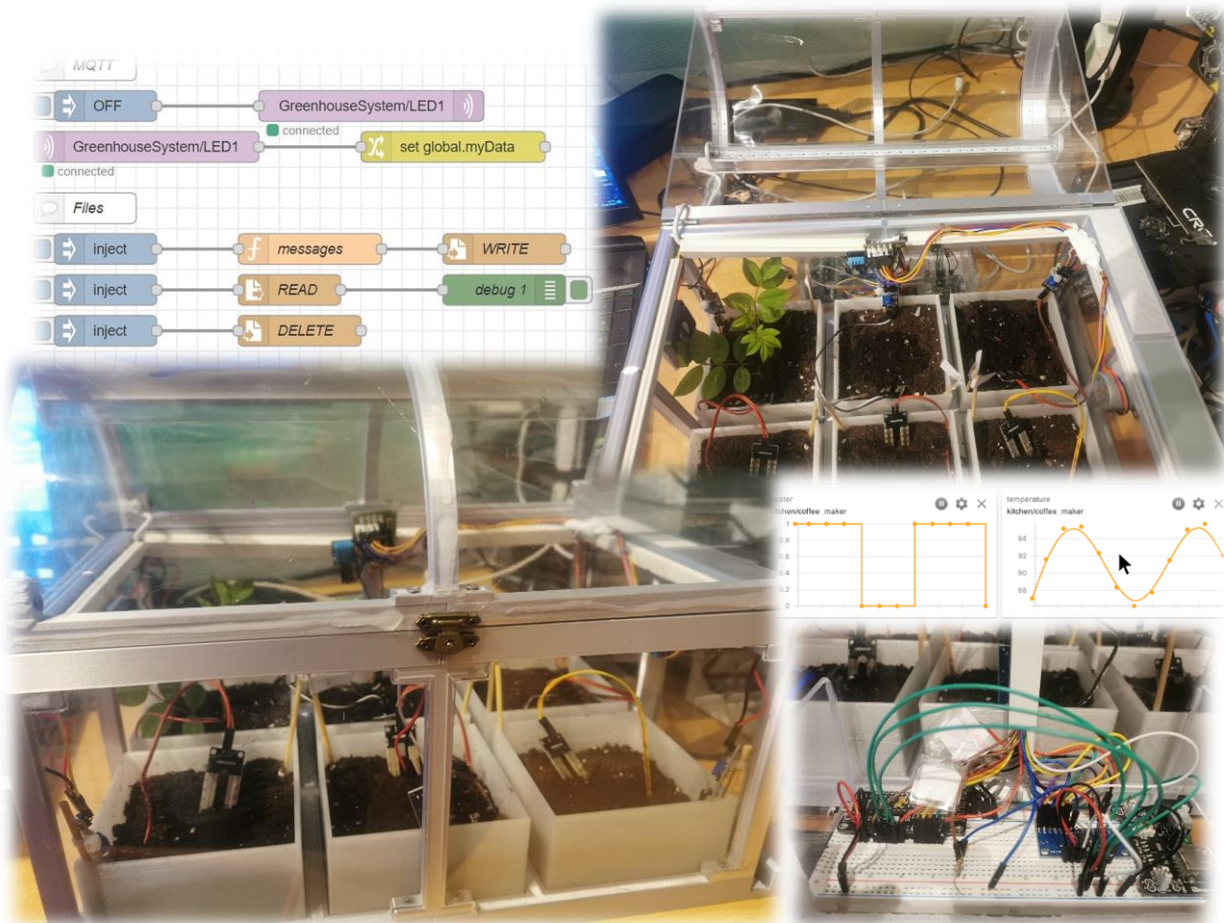


Subject:

Application to robotics & IOT

Document title:

Automated Greenhouse System (Mockup)



Prepared by:

Name

M&M

Course

4ºA

08/05/2023

Date

Reviewed:

Name

Date

File name:

M&M.pdf

Date:

08/05/2023

Edition:

1

Nº pages:

38



Documentary control

Authors	Miguel Vilanova Fenollar Miguel Monge Rodríguez
Project Name	Automated Greenhouse System

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
0.6	3D model	Design all pieces and printed	15/03/2023
0.7	Prototype Testing	Tested the initial prototype and identified issues (need to buy more hardware)	30/03/2023
0.8	Refinement	Refined the design based on testing results	06/04/2023
0.9	Final Prototype	Produced the final prototype	18/04/2023
1.0	Documentation Revision	Documented the final design and testing results	07/05/2023

* N/A = Not applicable

Index

Introduction	5
Planification.....	6
Determine the requirements	7
URD.....	7
Risk planning	8
Hardware.....	9
Select sensors and actuators.....	9
Design physical model.....	10
Software	11
Design NodeMCU interface.....	12
Develop the software	14
Develop control algorithm	22
Building and Testing.....	23
Buy & Check	23
Building the 3D mockup	23
Programming.....	26
Arduino.....	26
NodeRED	26
Telegram.....	27
Testing.....	30
Sensor data validation.....	30
Stepper motor control	30
MQTT communication.....	30
Telegram.....	31
NodeRED	31
System stability and reliability	31
Integration testing.....	31
State of Art (Professional Approach)	32
SDGs	34
Summary	36
Pros.....	37
Cons.....	37
References.....	38

Figures

1: Gantt Project, file attached.	6
2: GreenHouseSystem Process flow - own elaboration	11
3: NodeMCU e-diagram [1]	12
4: Node Red interface example [2]	13
5: MQTT Explorer (MQTT broker portable) [3]	13
6: Docker Image configuration - own elaboration	14
7: MQTT container running on port 1883 - own elaboration	14
8: Telegram @BotFather	15
9: Telegram Bot setup: - own elaboration	15
10: Telegram Bot on the M&M Group - own elaboration	16
11: NodeRED container	18
12: Manage Palette NodeRed - own elaboration	18
13: NodeRed Flow - own elaboration	20
14: Amazon S3 bucket [9]	20
15: Mockup Chassis - own elaboration	23
16: Temperature & Humidity sensors - own elaboration	23
17: Soil moisture sensors - own elaboration	24
18: Door rotor - own elaboration	24
19: Fan integrated - own elaboration	25
20: Finaliced Mockup - own elaboration	25
21: JavaScript node OpenDoor - own elaboration	27
22: JavaScript node Feedback - own elaboration	27
23: JavaScript node Control - own elaboration	28
24: JavaScript node UmbralTemp - own elaboration	28
25: Files to send to AWS Node configuration	29
26: Growlink Website [4]	32
27: Autogrow by Bluelab Website [5]	32
28: Heliospectra Website [6]	33
29: SDGs UN [7]	34

Tables

Table i: URD	7
Table ii: Risks	8
Table iii: Main Hardware	9
Table iv: Sensors & Actuators	9
Table v: Filament	10

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.

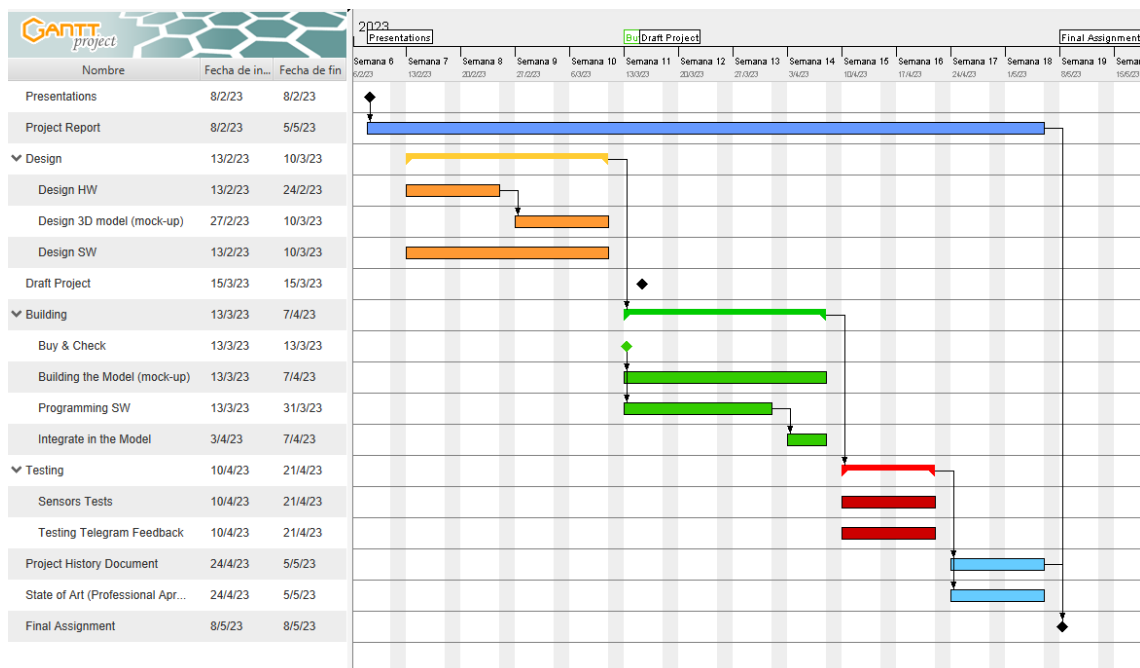
1. **Sensors collect data:** The sensors placed inside the greenhouse collect data on the environmental conditions such as temperature, humidity, light, soil moisture, and CO2 levels.
2. **NodeMCU collects and sends data:** The NodeMCU board, which is connected to the sensors, collects the data and sends 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.



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

Table i: 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

Risk planning

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



Table ii: Risks

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

Design

Hardware

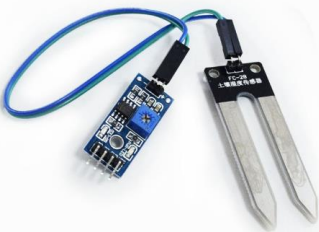
Table iii: Main Hardware


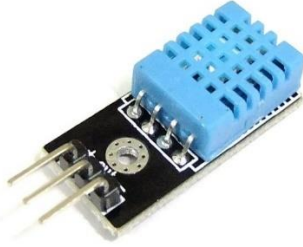

Motherboard	Description	Price	Photo
NodeMCU ESP8266 X1	We use this board to be able to control the sensors and actuators and communicate with the node-red through the MQTT protocol	4,00€/u	
CD74HC4067 Multiplexor X1	We use this multiplexer to make use of all the sensors that require analogic signal, since ESP8266 has only one A0.	1,00€/u	

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.

Table iv: Sensors & Actuators

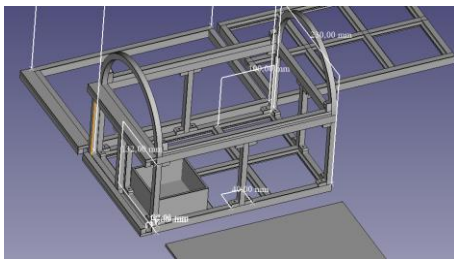
Sensor / Actuators	Description	Price	Photo
soil moisture X6	It measures soil moisture and is more accurate than others that we have used before	0,52€/u	

Servo motor X1	We use one servo motor to open de door and renew the air.	2,30€/u	
Temperature X1	we use this sensor to measure the temperature inside the greenhouse.	0,70€/u	
UV led strip X1	we use ultraviolet led to increase growth in low light hours.	3,68€/u	

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.

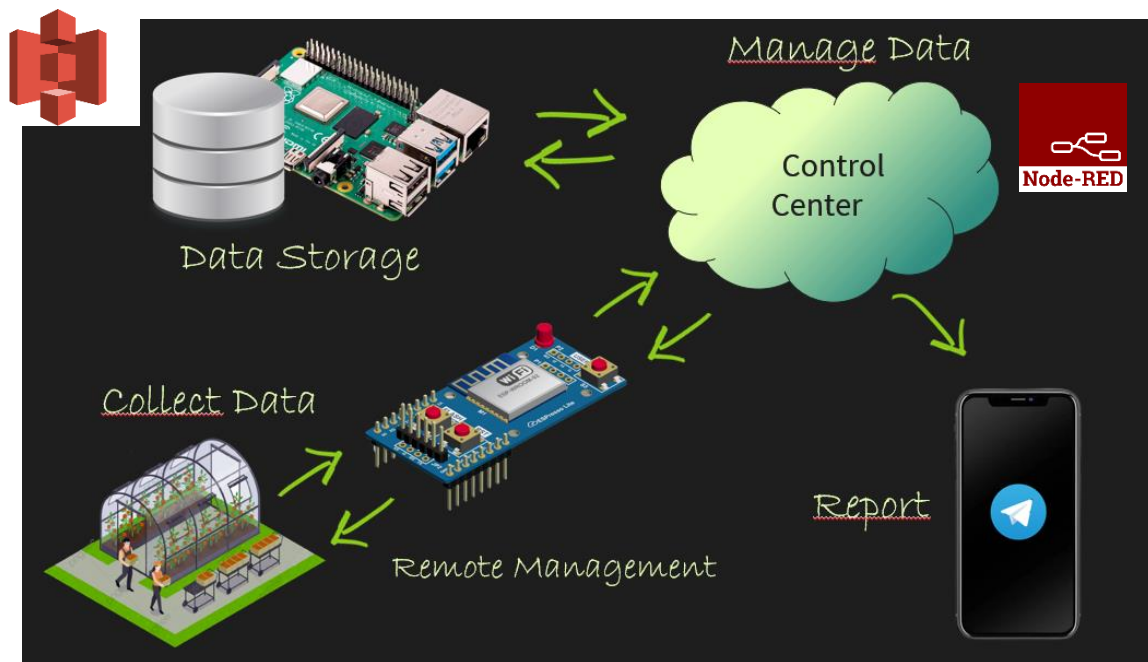
Table v: Filament

Piece	Time	Description	Price	Photo
3D filament X6	The totally time to print all the pieces was around 118 hours	we use 3d filament to print all the parts of the model.	27,50 €/u	

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.



2: GreenHouseSystem Process flow - own elaboration

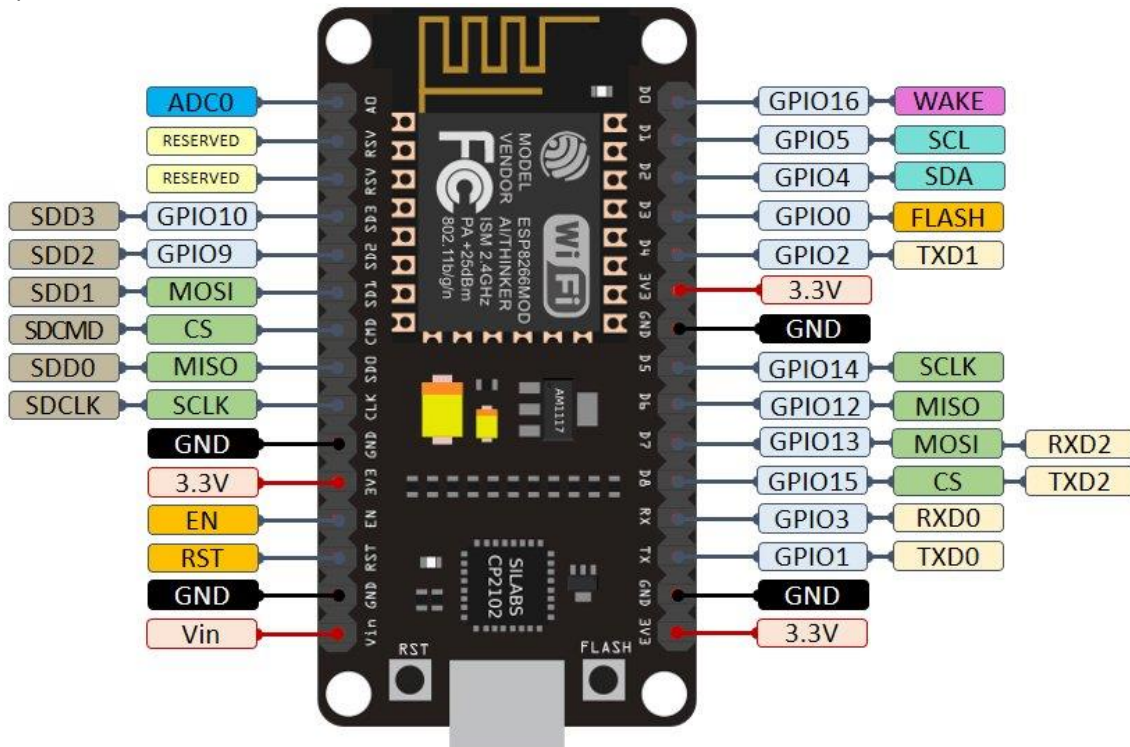
By following these steps, we can ensure that the design phase of the automated greenhouse system is thorough and well-planned, and that it meets the objectives and plant requirements. This will set us up for success during the building and testing phase.

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.

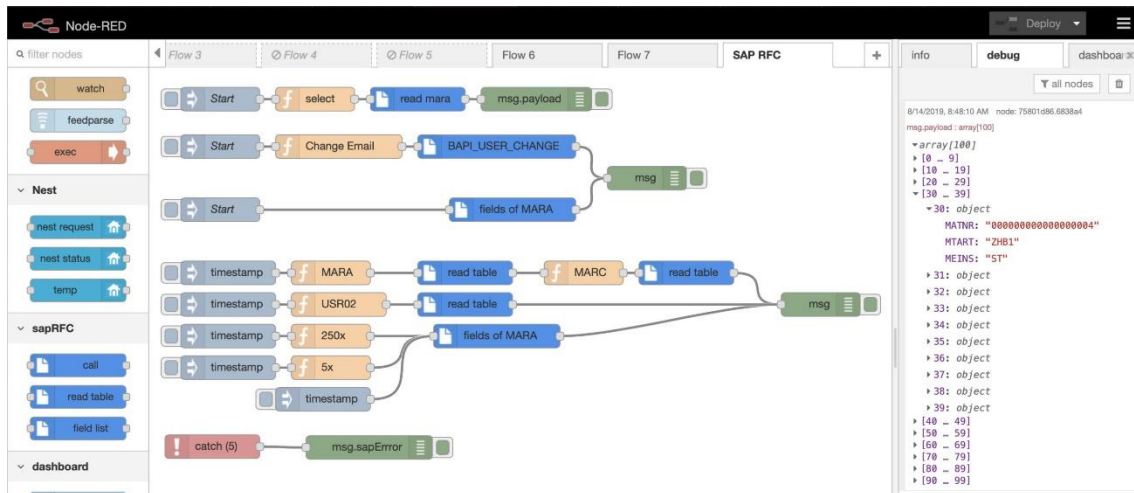


3: NodeMCU e-diagram [1]

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.



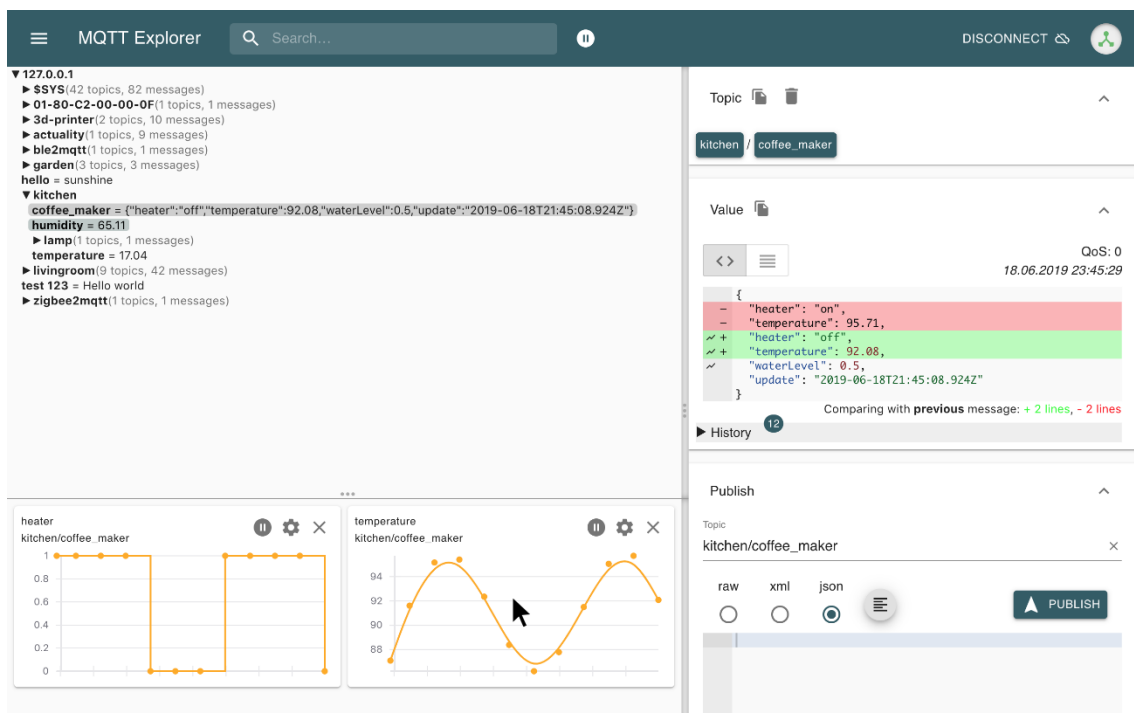
4: Node Red interface example [2]

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.



5: MQTT Explorer (MQTT broker portable) [3]

Develop the software

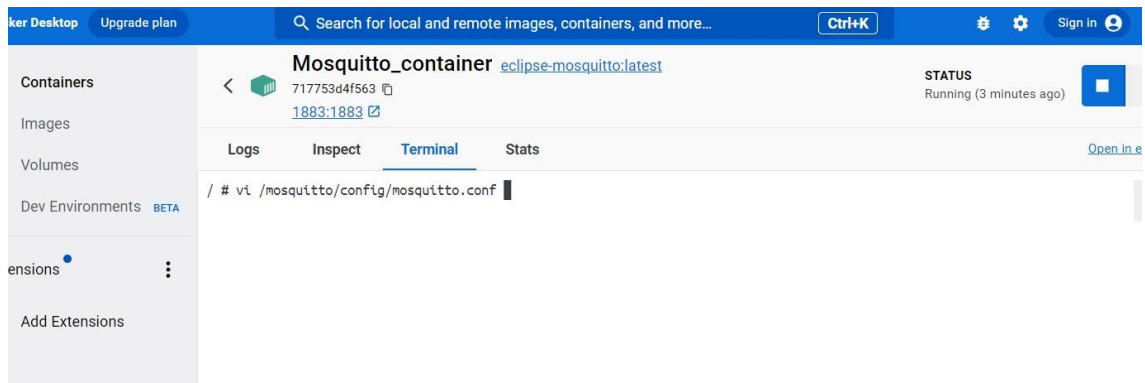
Finally, we will develop the software components, including the Node-RED platform and the data storage and analysis tools. This will allow us to process and analyze the data collected from the sensors and provide insights into the growth and efficiency of the plants.

Setup MQTT server (Docker)

We need to download the latest [eclipse-mosquitto](#) image:

CMD

```
docker pull eclipse-mosquitto
```



6: Docker Image configuration - own elaboration

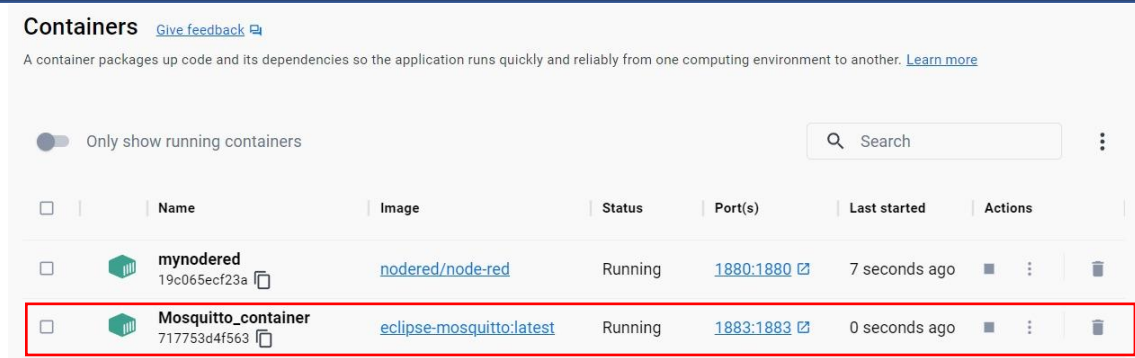
We need to edit the [configuration](#) file to allow anonymous connections and listen on port 1883, also we can create a config file and run the docker with that configuration.

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

```
listener 1883
allow_anonymous true
```

CMD

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



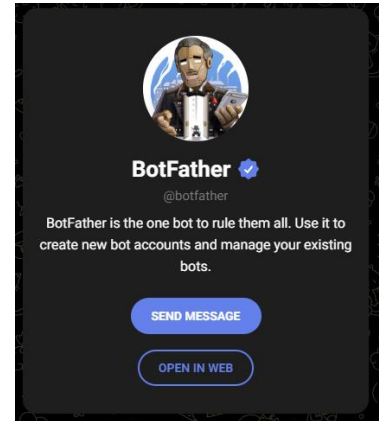
7: MQTT container running on port 1883 - own elaboration

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

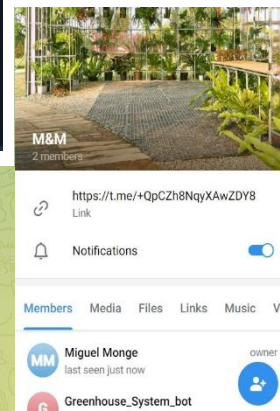
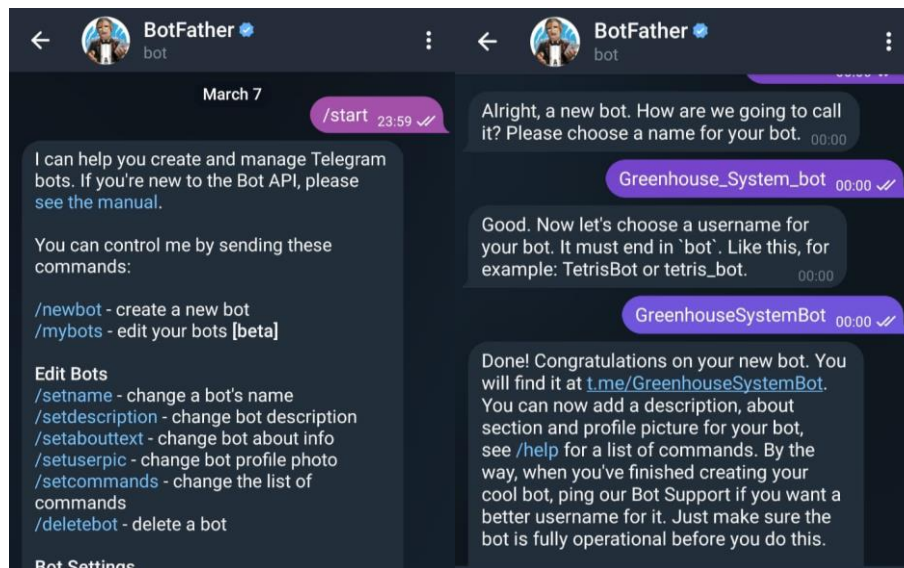
Setup Telegram Bots

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 or by typing '/start'.
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" (*you need to turn Group privacy off in the bot settings to allow the bot to read messages from the group).



8: Telegram @BotFather



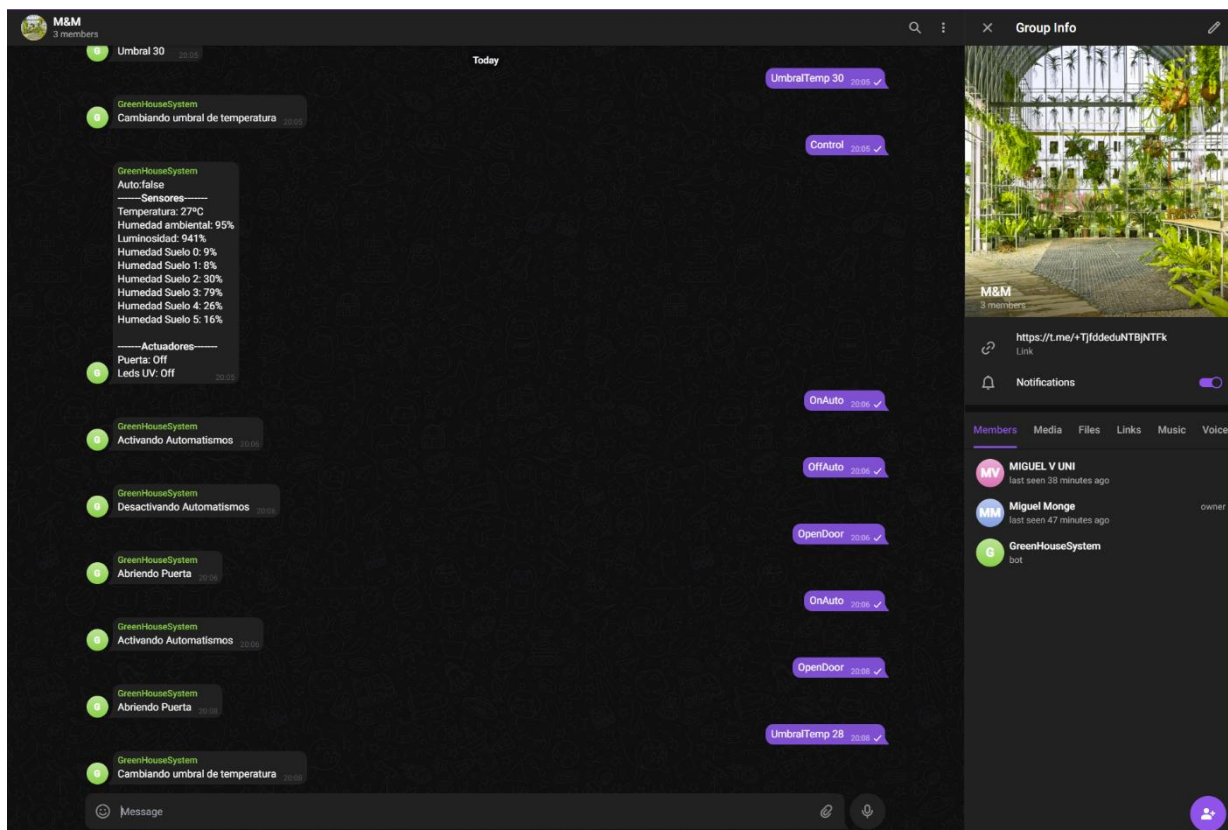
9: Telegram Bot setup: - own elaboration

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

Telegram Usage

We have a few commands to control the different variables and behaviors of the GreenHouseSystem:

- 'OpenDoor' makes the door open, and the fans turn on to reduce the temperature, if the automations are activated it will open when the established temperature threshold is passed.
- 'CloseDoor' to close it and turn the fan off.
- 'OnUV' turns on the ultraviolet leds.
- 'OffUV' turns off the ultraviolet leds.
- 'OnAuto' activates the automations.
- 'OffAuto' turns them off.
- 'UmbraTemp' + 'number' establishes the number that we have passed as the temperature threshold, so if the Automatismos are activated the door will open only when it passes this threshold.



10: Telegram Bot on the M&M Group - own elaboration

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.

Amazon S3 > Buckets > Crear bucket

Crear bucket [Info](#)

Los buckets son contenedores de datos almacenados en S3. [Más información](#)

Configuración general

Nombre del bucket

El nombre del bucket debe ser único en todo el mundo y no debe contener espacios ni letras mayúsculas. [Consulte las reglas para la denominación de los buckets](#)

Región de AWS

Copiar la configuración del bucket existente: *opcional*
Solo se copia la configuración del bucket en los siguientes ajustes.

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

	Nombre	Región de AWS	Acceso	Fecha de creación
<input type="radio"/>	mygreenhousebucket	UE (Londres) eu-west-2	Bucket y objetos que no son públicos	7 Mar 2023 11:32:13 PM CET

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.

CMD

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

Containers [Give feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

☐ Only show running containers

Search

<input type="checkbox"/>	Name	Image	Status	Port(s)	Last started	Actions
<input type="checkbox"/>	mynodered 19c065ecf23a	nodered/node-red	Running	1880:1880	7 seconds ago	
<input type="checkbox"/>	Mosquito_container 717753d4f563	eclipse-mosquitto:latest	Running	1883:1883	0 seconds ago	

11: NodeRED container

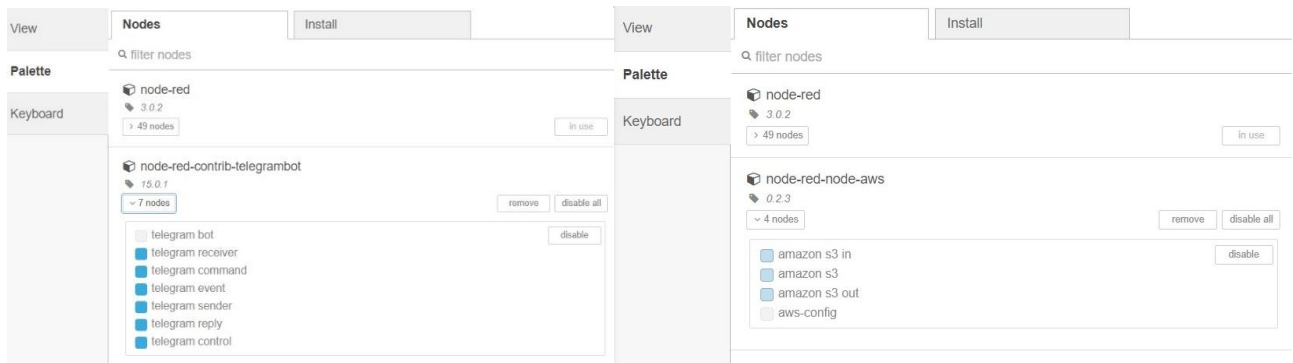
Now you can easily create files in the data volume inside Node Red container, we will need this to upload logs to AWS S3 bucket (and then maybe process it with EC2).

Click on manage palette and search for:

node-red-contrib-telegrambot

&

node-red-node-aws:



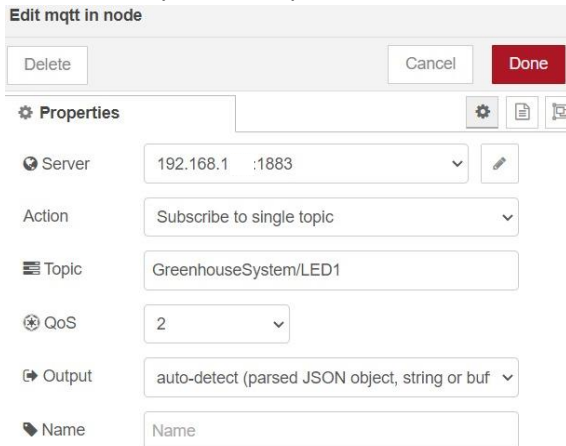
12: Manage Palette NodeRed - own elaboration

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.
 - a. Use your local Ip, localhost:1883 does not work since is a container.



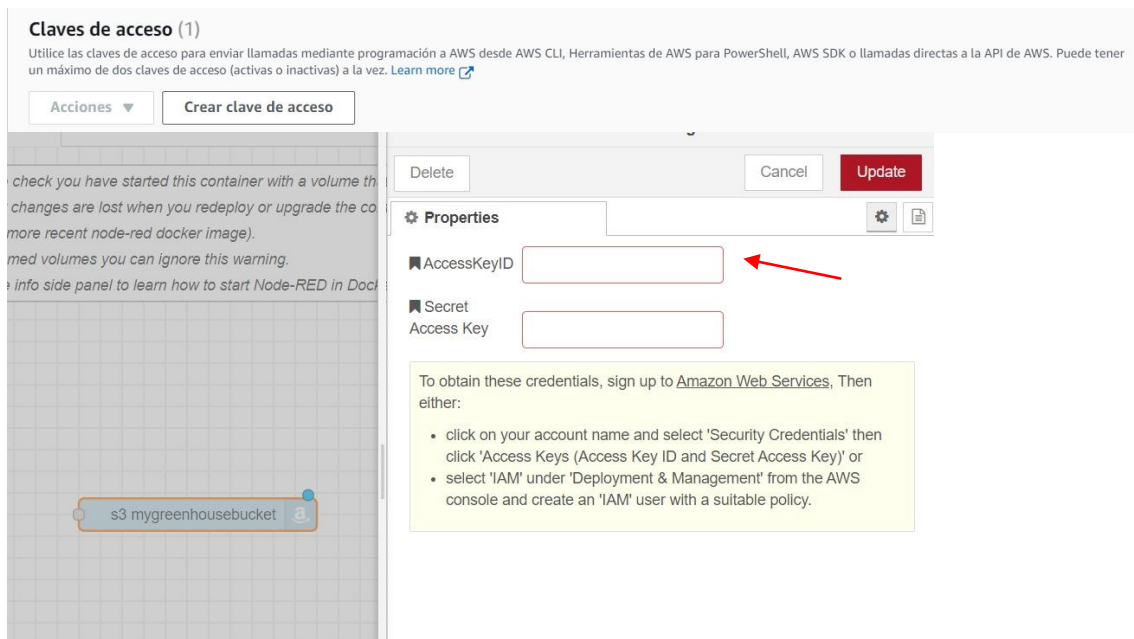
2. Make sure the telegram bot token and chat IDs are correct.
 - a. Parse msg.payload as follows to tell the bot which chat to type in:

```

1  var message = "LED 1 = " + global.get("myData");
2  // Configuramos el payload
3  msg.payload = {
4    chatId: -912627664,
5    type: 'message',
6    content: message
7  };

```

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



Claves de acceso (1)

Utilice las claves de acceso para enviar llamadas mediante programación a AWS desde AWS CLI, Herramientas de AWS para PowerShell, AWS SDK o llamadas directas a la API de AWS. Puede tener un máximo de dos claves de acceso (activas o inactivas) a la vez. [Learn more](#)

Acciones ▼ Crear clave de acceso

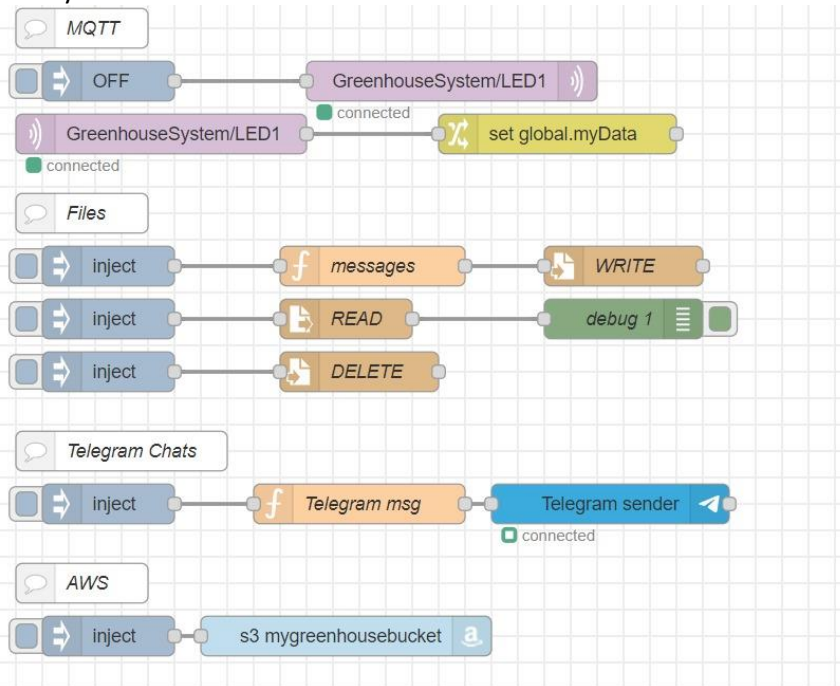
check you have started this container with a volume th
changes are lost when you redeploy or upgrade the co
more recent node-red docker image).
med volumes you can ignore this warning.
Info side panel to learn how to start Node-RED in Doc

s3 mygreenhousebucket

To obtain these credentials, sign up to [Amazon Web Services](#). Then either:

- click on your account name and select 'Security Credentials' then click 'Access Keys (Access Key ID and Secret Access Key)' or
- select 'IAM' under 'Deployment & Management' from the AWS console and create an 'IAM' user with a suitable policy.

Now you can create a test flow as follows:



13: NodeRed Flow - own elaboration

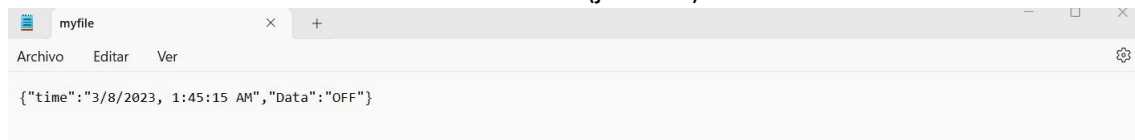
As we can see the test file is created in local storage, the data volume we created previously, and now we send that file to the AWS bucket.



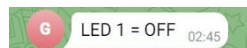
14: Amazon S3 bucket [9]

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:



Another function appends the msg to a file.txt and store in data volume.

Edit amazon s3 out node

Delete Cancel Done

Properties

AWS AWS

Bucket mygreenhousebucket

Filename myfile.txt

Local Filename /data/myfile.txt

Region eu-west-2

Name Name

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.

As our system is very small we do not have enough values to make an interesting datamining model.

In general, greenhouse systems require climate and humidity control, which means that internal conditions must be monitored and adjusted to ensure they are optimal for plant growth. If the greenhouse system is small and only basic monitoring is needed, machine learning or data mining may not be necessary. In that case, a simple rule-based control strategy can be used to automatically adjust humidity and temperature levels based on ambient conditions.

On the other hand, if the greenhouse is large and complex, with many variables to control and adjust, a machine learning system or data mining model can be improved. These systems can use data from sensors and other factors to predict how internal greenhouse conditions will change over time and make decisions about how best to adjust humidity and temperature levels to optimize plant growth.

What we have done is simply define the thresholds of temperature, humidity, and other relevant parameters for the growth of your plants and establish a control system that automatically adjusts the internal conditions of the greenhouse according to those thresholds.

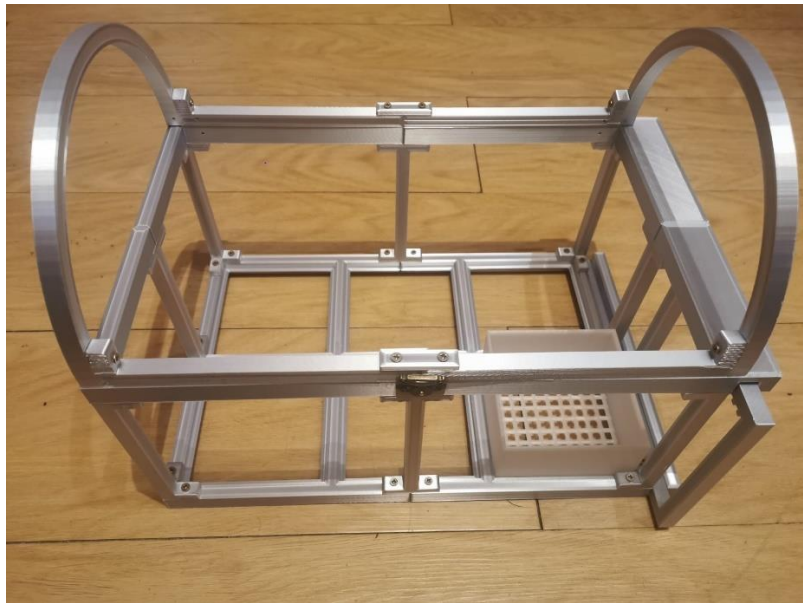
*For example, you can program the system so that if the temperature exceeds a certain value, the ventilation system is activated to cool the environment, if the greenhouse is not receiving enough light, then the UV LEDs are turned on, or if the humidity is too low, irrigation is activated to increase it (**WIP**).*

Building and Testing

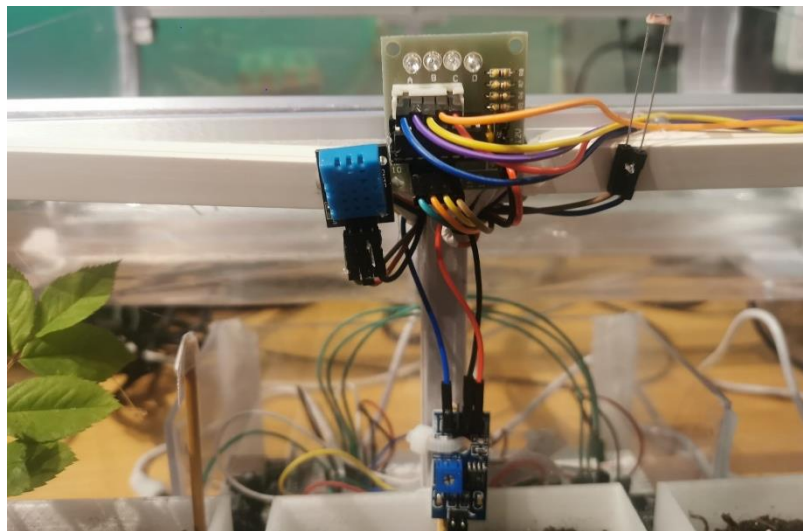
Buy & Check

Building the 3D mockup

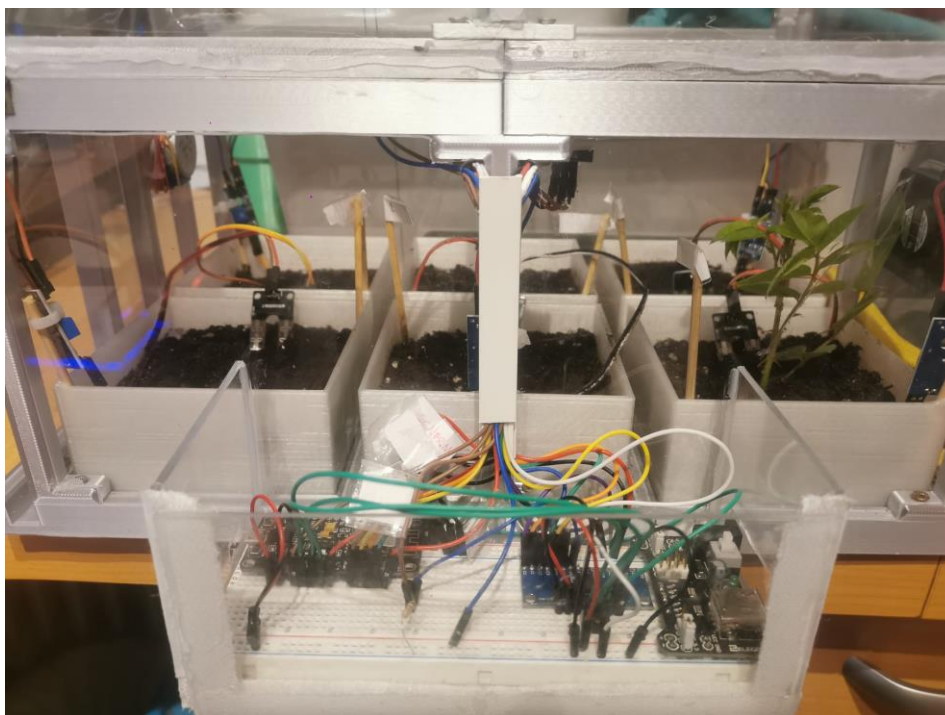
This is how the complete impression of all the pieces has been and assembled. The next step is to include the sensors and actuators and finalize the model.



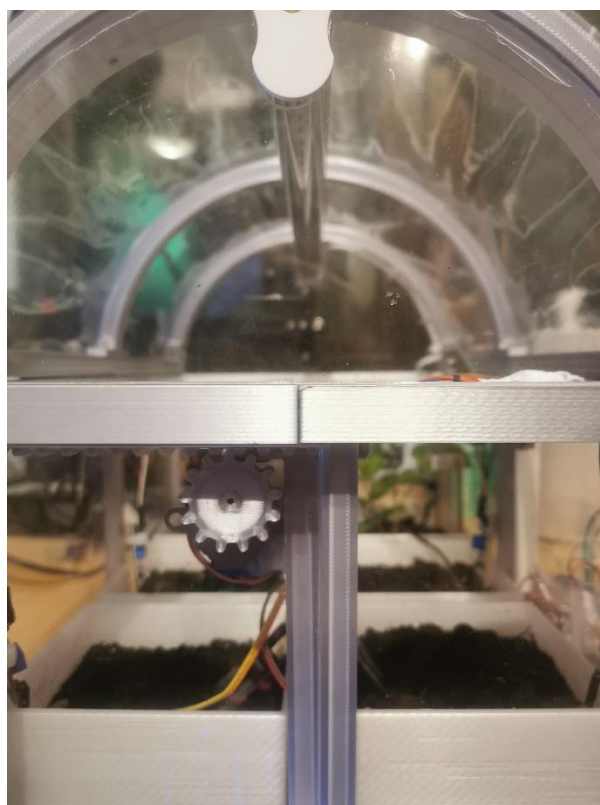
15:Mockup Chassis - own elaboration



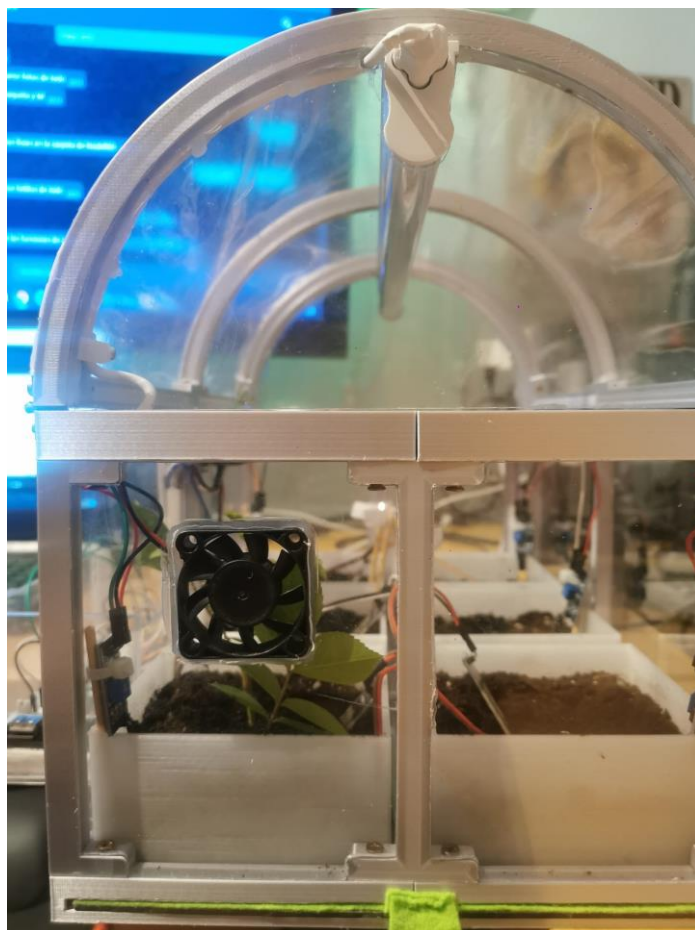
16: Temperature & Humidity sensors - own elaboration



17: Soil moisture sensors - own elaboration



18: Door rotor - own elaboration



19: Fan integrated - own elaboration



20: Finaliced Mockup - own elaboration

Programming

Arduino

This code is a complete implementation of an automated greenhouse system using an ESP8266 NodeMCU board to control various sensors and actuators. The system connects to an MQTT server for communication and control.

The following libraries are included:

- 1) DHT.h: Library for reading DHT humidity and temperature sensors.
- 2) AccelStepper.h: Library for controlling stepper motors using acceleration profiles.
- 3) ESP8266WiFi.h: Library for connecting and managing the ESP8266's Wi-Fi capabilities.
- 4) PubSubClient.h: Library for MQTT communication.

The code starts by defining MQTT, Wi-Fi, and sensor configurations. The motor configuration, including stepper motor pins and steps per revolution, are also defined.

In the `setup_wifi()` function, the ESP8266 connects to the Wi-Fi network using the provided credentials, same example function.

The `callback()` function handles incoming messages from the MQTT server. The code checks for messages related to the greenhouse's door motor and controls the stepper motor accordingly.

The `reconnect()` function attempts to reconnect the ESP8266 to the MQTT server if it's not already connected.

In the `setup()` function, the following tasks are performed:

- 1) Initializing the serial port.
- 2) Connecting to the Wi-Fi network.
- 3) Configuring the MQTT server and callback function.
- 4) Initializing the DHT sensor.
- 5) Configuring the sensor and motor pins as inputs or outputs.
- 6) Setting the stepper motor's speed and acceleration.

The `loop()` function maintains the MQTT connection and calls the `sensores()` function every second to read sensor data and then send it to the MQTT server.

The `sensores()` function reads data from multiple sensors connected to a multiplexer. It iterates through each channel and reads the appropriate sensor values for soil moisture, ambient light, temperature, and humidity. The readings are then printed to the serial console.

NodeRED

RED Node is the intermediary between Telegram, the ESP8266 .ino code, the MQTT server and the cloud database. First we need to receive the data from the MQTT that the ESP8266 .ino code writes, the Sensors topic, then we need to receive messages from Telegram and give feedback with the data recovered from the MQTT, later we can make it so that through what the telegram receives, data is modified in the MQTT server, in this case the Actuators topic.

Telegram

We managed to receive telegram messages and comparing each command to redirect to a specific action node of the flow:

For example, for the subject of ventilation and the mechanical door, if OpenDoor is sent, the value of the topic is modified in the MQTT server:

```
var messageContent = msg.payload.content;
var newMsg;

if (messageContent.includes("OpenDoor")) {
    newMsg = {
        payload: "On"
    };
} else if (messageContent.includes("CloseDoor")) {
    newMsg = {
        payload: "Off"
    };
} else {
    newMsg = null;
}

return newMsg;
```

21: JavaScript node OpenDoor - own elaboration

Subsequently send feedback to the group to confirm that the action has been processed:

```
// Obtener el contenido del mensaje
var messageContent = msg.payload.content;

if (messageContent.includes("OnUV")){
    msg.payload = {
        chatId: -945976936,
        type: 'message',
        content: "Encendiendo Leds UV",
        options: {
            parse_mode: "HTML"
        }
    }
}
else if (messageContent.includes("OffUV")){
    msg.payload = {
        chatId: -945976936,
        type: 'message',
        content: "Apagando Leds UV",
        options: {
            parse_mode: "HTML"
        }
    }
}

// Devolver el objeto `msg`
return msg;
```

22: JavaScript node Feedback - own elaboration

Control data from all sensors is also sent in this way on the same node but changing the JavaScript function that prepares the message:

```
msg.text = " Auto:" + flow.get("Auto") + "\n" +
  "<b>-----Sensores-----</b>" +
  "\nTemperatura: " + flow.get("Temp") + "°C" +
  "\nHumedad ambiental: " + flow.get("Humi") + "%" +
  "\nLuminosidad: " + flow.get("LDR") + "%" +
  "\nHumedad Suelo 0: " + flow.get("SM0") + "%" +
  "\nHumedad Suelo 1: " + flow.get("SM1") + "%" +
  "\nHumedad Suelo 2: " + flow.get("SM2") + "%" +
  "\nHumedad Suelo 3: " + flow.get("SM3") + "%" +
  "\nHumedad Suelo 4: " + flow.get("SM4") + "%" +
  "\nHumedad Suelo 5: " + flow.get("SM5") + "%\n\n" +
  "<b>-----Actuadores-----</b>" +
  "\nPuerta: " + flow.get("Puerta") +
  "\nLeds UV: " + flow.get("LedsUV") + "\n";
var messageContent = msg.payload.content;
if (messageContent.includes("Control")) {
  msg.payload = {
    chatId: -945976936,
    type: 'message',
    content: msg.text,
    options: {
      parse_mode: "HTML"
    }
  }
}
```

23: JavaScript node Control - own elaboration

For the threshold we use a regex to extract the numbers that follow the command:

```
var messageContent = msg.payload.content;
var newMsg;

if (messageContent.includes("UmbralTemp")) {
  let umbral = messageContent.match(/\d+/)[0]; // extrae el primer número de la cadena
  let umbralString = umbral.toString(); // convierte el número en un string
  newMsg = {
    payload: umbralString
  };
} else {
  newMsg = null;
}

return newMsg;
```

24: JavaScript node UmbralTemp - own elaboration

AWS S3

We set the local file logs to be generated every day from 9:00 to 21:00 and send it to the cloud service so we can check anytime if there are errors or so we can provide more insight.

☐ Inject once after seconds, then

▾

every minutes

between and

on ☒ Monday ☒ Tuesday ☒ Wednesday
☒ Thursday ☒ Friday ☒ Saturday
☒ Sunday

25: Files to send to AWS Node configuration

Testing

The code provided doesn't include any explicit testing methods or unit tests. However, based on the code structure, we can perform some manual tests iteratively during development and deployment, with any issues being addressed and resolved to ensure the automated greenhouse system functions correctly and reliably.

Sensor data validation

As the code reads values from various sensors (soil moisture, ambient light, temperature, and humidity), it's essential to verify that the sensor readings are accurate and consistent. This can be done by comparing the readings with known values or manually checking the sensor output with reference equipment.

During our testing phase, we encountered a few issues and learned valuable lessons while working on the project:

- ✓ Soil sensor calibration: We found that some of the soil sensors were not properly adjusted, so we had to map their voltage values. Calibrating these sensors correctly is essential to obtain accurate readings. As a team, we compared their output with known moisture levels in the soil and adjusted the mapping accordingly.
- LDR sensor issues: We noticed that the LDR sensor wasn't working properly. This required further investigation and possible adjustments to the circuit or replacement of the sensor to ensure accurate light readings. (**WIP**)

Stepper motor control

The code controls a stepper motor for opening and closing the greenhouse door based on received MQTT messages. Testing the motor control involves verifying that the motor moves the correct number of steps in the desired direction when receiving "On" or "Off" commands.

- ✓ Stepper motor selection: Initially, we used a 90-degree servo motor for opening the door. However, it didn't provide the required range of motion, so we had to switch to a regular motor. Unfortunately, the first motor we tried caused issues in the overall circuit, leading us to test another motor. This trial-and-error process helped us understand the importance of selecting the right components for the system.

MQTT communication

The code uses MQTT for sending and receiving messages to control the system. Testing the MQTT communication involves verifying that the ESP8266 can connect to the MQTT server, subscribe to the correct topics, and receive messages as expected. It's also crucial to ensure that the callback function processes the incoming messages correctly and performs the appropriate actions, such as controlling the stepper motor for opening or closing the door.

- ✓ Works just as expected.

Telegram

We integrated our automated greenhouse system with Telegram, a popular messaging platform. This allowed us to send real-time notifications, alerts, and updates about the greenhouse's environmental conditions to a designated group chat. Moreover, we were able to send commands and control the greenhouse remotely through the Telegram bot. During the testing phase.

- ✓ We verified that the Telegram bot was functioning correctly, responding to commands, and providing accurate updates.

NodeRED

Node-RED, a flow-based programming tool, played a crucial role in processing the data collected by the sensors in real-time and triggering actions based on pre-defined conditions. We tested the Node-RED flows to ensure that they were correctly processing the sensor data, and the actuators were responding to the commands. Additionally, we verified that the data storage and analysis were functioning properly, providing valuable insights for optimizing resource usage and predicting future growth.

- ✓ By thoroughly testing the integration of Telegram and Node-RED, we ensured that our automated greenhouse system was able to operate efficiently, while providing remote monitoring and control capabilities.

These tests helped us identify any issues or improvements needed for better performance and user experience.

System stability and reliability

As the system will run continuously, it's essential to test the system's stability and reliability over extended periods. This can be done by monitoring the system's performance and checking for any issues or failures in the sensor readings, MQTT communication, or motor control.

- ✓ Maintenance and sensor lifetime: We learned that our system requires periodic maintenance, mainly due to the limited lifetime of some sensors. Being aware of this aspect will help us plan and schedule maintenance tasks to keep the system running optimally.

Integration testing

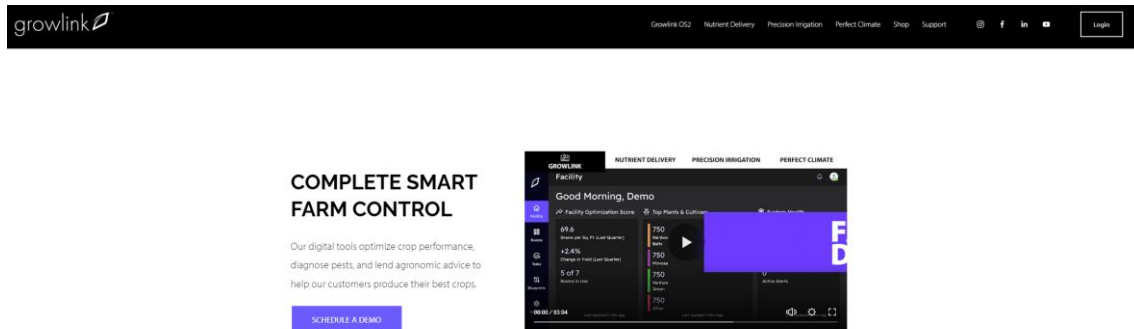
Finally, after testing individual components, it's crucial to perform integration testing to ensure that the entire system works as intended. This involves running the complete system and verifying that sensor readings are accurate, MQTT communication is functioning correctly, and actuators are responding as expected.

- ✓ System integration: We successfully integrated our system with Telegram and AWS, which allowed us to monitor and control the greenhouse remotely. This integration showcases the potential of our automated greenhouse system in a real-world context.

State of Art (Professional Approach)

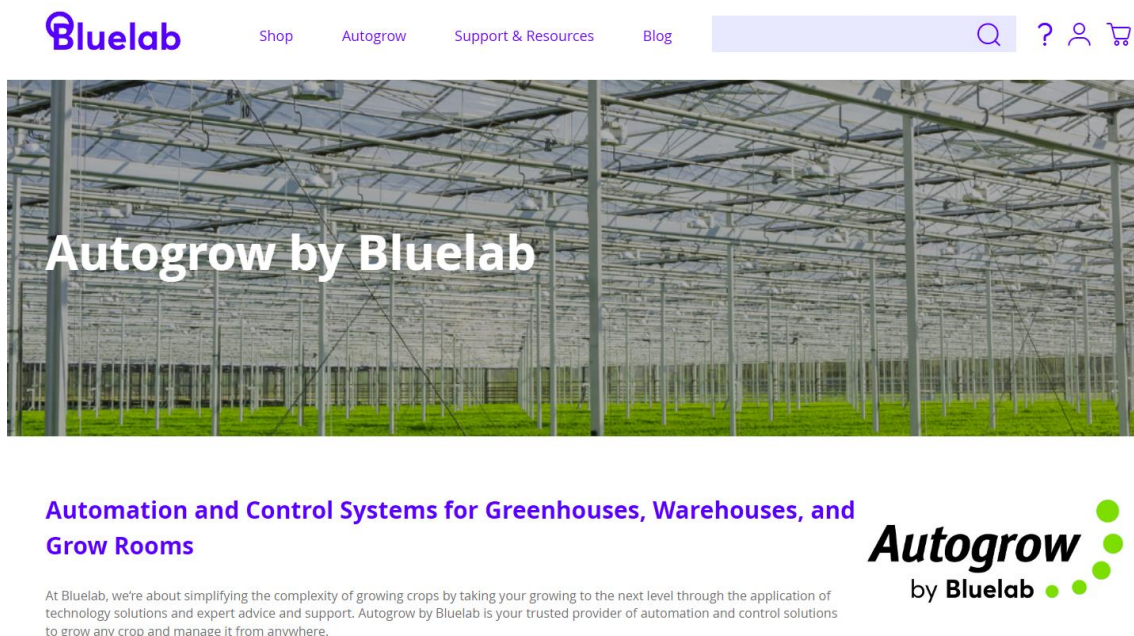
There are existing solutions in the market that offer automated greenhouse systems. Some well-known companies and products in this space include:

1. Growlink offers a smart greenhouse controller system that enables users to monitor and control their greenhouse environment remotely. Their platform integrates various sensors, actuators, and controls for temperature, humidity, CO₂, irrigation, and more.



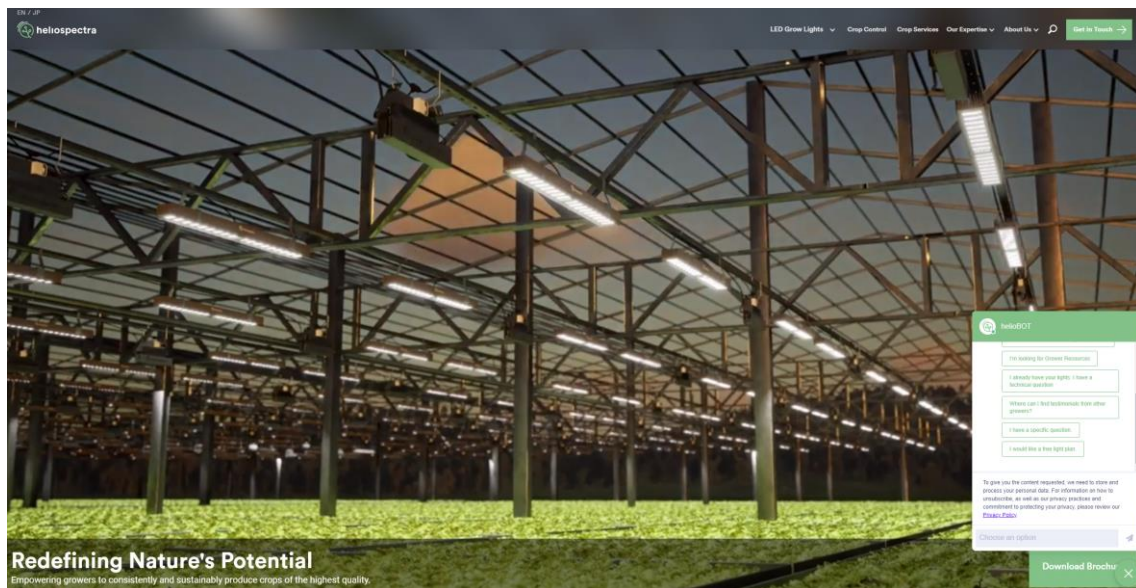
26: Growlink Website [4]

2. Autogrow, by Bluelab, provides a range of automated greenhouse solutions, including their IntelliGrow and IntelliDose systems. These systems offer remote monitoring and control of environmental factors, irrigation, and fertigation.



27: Autogrow by Bluelab Website [5]

- Heliospectra specializes in intelligent lighting technology for controlled environment agriculture. They offer a range of LED grow lights, along with their helioCORE™ control system, which enables growers to automate and optimize their lighting strategies.



28: Heliospectra Website [6]

By collecting and analyzing data from various sensors in real-time, these projects are focused on making data-driven decisions for optimizing plant growth, resource usage, and environmental conditions. This data-centric approach enhances the system's effectiveness and sets it apart from traditional greenhouse management methods.

The key is the modular design and cloud-based platform allow for customization and scalability, making it suitable for a wide range of greenhouse sizes and agricultural applications. This flexibility ensures that the system can adapt to the specific needs of different users, adding value to the project.

SDGs



29: SDGs UN [7]

These projects emphasize on resource efficiency and sustainability, addressing the growing global concerns regarding environmental impact and resource depletion. This focus on sustainability appeals to environmentally conscious customers and aligns with the global trend towards eco-friendly practices in agriculture.



SDG 2: Zero Hunger

By optimizing plant growth and maximizing resource efficiency, your automated greenhouse system can contribute to increased agricultural productivity, which can help to address global food security issues and work towards the goal of zero hunger.



SDG 6: Clean Water and Sanitation

The project's focus on efficient irrigation systems and monitoring soil moisture helps to reduce water waste and ensure that water resources are used more sustainably in agricultural practices.



SDG 7: Affordable and Clean Energy

The automated greenhouse system promotes the efficient use of energy through smart lighting controls and ventilation systems. By minimizing energy waste, your project contributes to the development of more sustainable and efficient energy consumption practices in agriculture.



SDG 9: Industry, Innovation, and Infrastructure

Your project integrates cutting-edge technologies and innovative solutions for agricultural practices. By fostering innovation in greenhouse management, the project contributes to the development of sustainable and resilient infrastructure for agriculture.

**SDG 11: Sustainable Cities and Communities**

The automated greenhouse system can be utilized in urban farming initiatives, helping to create sustainable food production methods within cities and communities, reducing the carbon footprint associated with transporting food over long distances.

**SDG 12: Responsible Consumption and Production**

The project's emphasis on resource efficiency and data-driven decision-making helps to minimize waste and promote more responsible consumption and production patterns in the agricultural sector.

**SDG 13: Climate Action**

By optimizing resource usage and promoting sustainable agricultural practices, your automated greenhouse system can contribute to the reduction of greenhouse gas emissions, ultimately aiding in the global fight against climate change.

Development Goals by promoting sustainable agricultural practices, optimizing resource usage, and integrating innovative technologies. These efforts contribute to a more sustainable and resilient future for both agriculture and the environment.

Conclusions

Summary

The automated greenhouse system aims to optimize plant growth, maximize resource efficiency, and promote sustainability by integrating sensors, actuators, and control algorithms. The system utilizes various sensors to collect data on environmental conditions within the greenhouse, such as temperature, humidity, light, soil moisture, and CO2 levels. This data is sent to the cloud-based platform, Node-RED, via a NodeMCU board with Wi-Fi connectivity. Node-RED processes the data in real-time and triggers actions based on pre-defined conditions, helping to maintain optimal environmental conditions for the plants. Processed data is stored and analyzed on a Raspberry Pi or cloud-based database, providing valuable insights for optimizing resource usage and predicting future growth.

The project plan consists of five phases: Design, Building and Testing, Sensor Testing and Feedback, Project Documentation, and Final Assignment. The system requires various sensors and actuators, such as temperature, humidity, soil moisture, and CO2 sensors, as well as lighting controls and an irrigation system. Risk planning and contingency plans have been developed to address potential challenges, such as sensor malfunctions, system failures, power outages, and data loss.

The design phase involves creating a 3D model of the automated greenhouse system, selecting appropriate sensors and actuators, and designing the NodeMCU interface and software components. By implementing this automated greenhouse system, the goal is to create a sustainable and efficient model for future agricultural practices, ultimately benefiting both plants and humans.

Pros

An automated greenhouse system offers numerous benefits, such as optimized plant growth by providing ideal environmental conditions tailored to each plant's specific needs. This can lead to increased crop yield and overall productivity.

Resource efficiency is also enhanced, as the system can intelligently manage water, energy, and nutrient consumption, reducing waste and operational costs. The real-time monitoring and data collection capabilities enable informed decision-making, allowing for quick adjustments to the environmental conditions or maintenance needs.

Moreover, the system promotes sustainability by reducing the greenhouse's ecological footprint and fostering responsible agricultural practices. Ultimately, an automated greenhouse system can serve as a model for future agricultural innovations, paving the way for a more sustainable and efficient food production system.

Cons

Despite the many advantages of an automated greenhouse system, there are some drawbacks to consider, including the initial cost of investing in sensors, actuators, and other technology components. Additionally, the technical complexity of installation, configuration, and maintenance can be challenging, and reliance on internet-connected systems exposes the greenhouse to cybersecurity risks.

The rapid evolution of technology may also lead to obsolescence of certain system components, requiring periodic updates. Furthermore, the manufacturing and disposal of electronic components can have an environmental impact, making it important to consider sustainable solutions throughout the entire lifecycle of these components.

References

- [1] Robots Argentina, "Robots didácticos, Automatización, control industrial, microcontroladores, electrónica digital," [Online]. Available: <https://robots-argentina.com.ar/referencia-de-distribucion-de-pines-ESP8266-que-pines-GPIO-usar.htm>. [Accessed 15 Febrero 2023].
- [2] NodeRED, "NodeRED," [Online]. Available: <https://nodered.org/>. [Accessed 2023 Febrero 15].
- [3] Eclipse Foundation, "Eclipse mosquitto - An open source MQTT broker," [Online]. Available: <https://mosquitto.org/>. [Accessed 15 Febrero 2023].
- [4] growlink, "Growlink," [Online]. Available: <https://www.growlink.com/>. [Accessed 20 Abril 2023].
- [5] Bluelab, "New Zealand Autogrow," [Online]. Available: https://bluelab.com/new_zealand/autogrow. [Accessed 20 Abril 2023].
- [6] Heliospectra, "Heliospectra," [Online]. Available: <https://www.heliospectra.com/>. [Accessed 20 Abril 2023].
- [7] UN, [Online]. Available: <https://www.un.org/sustainabledevelopment/es/development-agenda/>. [Accessed 20 Abril 2023].
- [8] Arduino, "Arduino.cc ESP8266," [Online]. Available: <https://search.arduino.cc/search?tab=reference&q=Esp8266>.
- [9] Amazon, "Amazon S3," [Online]. Available: <https://aws.amazon.com/es/s3>. [Accessed 15 Febrero 2023].