Greenhouse Digital Twin Project | Version 0.1

Technical Specifications Document

Guiziou Maleaume | Juan Francisco Martinez

|  |  |  |
| --- | --- | --- |
|  |  | Text  Description automatically generated |

# Document Version Control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version No.** | **Implemented By** | **Revision Date** | **Approved By** | **Approval Date** | **Description** |
| 0.1 | Andrew de Juan | 27/07/2023 | <Name> | <dd/mm/yyyy> | Initial draft |
| 0.2 | Andrew de Juan | 03/08/2023 |  |  | WBS headings and work packages |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Table of Contents

[Document Version Control 1](#_Toc544835204)

[Introduction 3](#_Toc172659111)

[Background 4](#_Toc1174195560)

[Document Purpose 4](#_Toc649210217)

[Intended Audience 4](#_Toc1123285967)

[Overview 5](#_Toc1602281101)

[Project Scope 5](#_Toc1041511273)

[Terminology 6](#_Toc1805043845)

[Reference Documents 6](#_Toc1188742678)

[Unity 3D (2021.3.2f1 LTS) 6](#_Toc697843018)

[Application Interface 6](#_Toc1981606213)

[1 Project Management 6](#_Toc1944531686)

[1.1 Define Scope and Objectives – This is a Work Package 7](#_Toc580327344)

[1.1.1 Task 01 7](#_Toc1818229894)

[1.2 Identify stakeholders and their roles 7](#_Toc1191802959)

[1.2.1 Task 01 7](#_Toc1048507244)

[1.2.2 Task 02 7](#_Toc1420226940)

[1.2.3 Task 03 7](#_Toc386169214)

[2 Requirements Gathering & Analysis 7](#_Toc1408300589)

[2.1 Identify sensor data sources 8](#_Toc15422433)

[2.1.1 Sensor Research and Selection 8](#_Toc575912001)

[2.1.2 Vendor Evaluation and Procurement 8](#_Toc364807037)

[2.1.3 Integration Planning 8](#_Toc1868188743)

[2.1.4 Data Interfacing and Compatibility 9](#_Toc548419108)

[2.2 Gather functional requirements 9](#_Toc1997785042)

[2.2.1 User Interviews and Surveys 9](#_Toc583591132)

[2.2.2 Use Case Definition 9](#_Toc739290904)

[2.2.3 Requirements Workshops 10](#_Toc1575152590)

[2.2.4 Prototyping and Feedback 10](#_Toc201067004)

[2.2.5 Requirement Documentation 10](#_Toc1224138664)

[2.3 Gather non-functional requirements 10](#_Toc1127889771)

[2.3.1 Task 11](#_Toc1147931946)

[2.4 Analyse and prioritize requirements 11](#_Toc1914275950)

[3. Design & Architecture 11](#_Toc43014368)

[3.1 Define data schema and storage requirements 12](#_Toc1970369437)

[3.2 Determine the technologies and frameworks for the frontend 12](#_Toc531816728)

[3.3.1 Unity 3D Game Engine 12](#_Toc422736974)

[3.3 Design the front-end user interface 12](#_Toc562716313)

[3.2.1 Camera System 12](#_Toc201712462)

[3.4 Design the back-end architecture for data processing 12](#_Toc1675183128)

[3.5 Plan data integration and APIs between sensors and the front-end 13](#_Toc1410089517)

[4 Sensor Data Acquisition 13](#_Toc1580110404)

[4.1 Procure necessary sensors and hardware 14](#_Toc1284862047)

[4.2 Develop or configure data collection software 14](#_Toc1229647995)

[4.3 Test data acquisition system and validate sensor data 14](#_Toc707639075)

[5 Data Processing and Storage 14](#_Toc1440992242)

# Introduction

## Background

The Internet-of-Things (IoT) technologies have enabled the mass generation of sensor data that is ready to be analyzed and visualized for the consumer to derive useful information to enable better decision-making. The idea for this is to harness and monitor data from greenhouses and present this in a very user-friendly and intuitive interface.

## Document Purpose

The purpose of this document is to provide sufficient detail on the overall process in achieving this end-to-end system architecture from sensor to front-end application design and development. This project is broken into multiple work packages which also function as headings in this document structure. The intended audience includes designers, developers, system integrators, and target end-users of the platform. This document is envisaged to serve as a blueprint with sufficient detail to replicate the workflow as well as commentaries on the developers’ experience working on platforms and systems.

[add more text]

## Intended Audience

People with different skillsets are likely to see this document for their own purposes, as follows:

**Developers:** This group will use the document to understand the technical requirements specifications and how they will be transformed into a system design when the final software application is written.

**Testers:** This group will use the document during unit testing and system integration testing to understand all system features and how the system responds to external stimuli.

**End users:** System end users will be more interested in the overall description of the Greenhouse dashboard, which will help them understand the final product that the developers and testers will deliver and be able to propose other requirements that the development team might miss.

This document is primarily for developers looking into expanding the project and for technical users to understand the various functions and features of the GUI.

## Overview

[add text]

## Project Scope

The project scope involves the development and implementation of a digital twin system for a greenhouse, encompassing an end-to-end sensor to front-end solution. The digital twin will be a virtual representation of the physical greenhouse, allowing real-time monitoring and analysis of environmental conditions and plant health. The system's primary goal is to visualize sensor data from various greenhouse components, such as temperature, humidity, soil moisture, light intensity, and CO2 levels, on an interactive and user-friendly front-end interface.

To achieve this, the project will include the acquisition of suitable sensors and hardware for data collection within the greenhouse environment. Data processing algorithms and machine learning techniques will be integrated into the back-end system to analyze the collected sensor data. These techniques will enable the system to identify patterns, trends, and anomalies in the greenhouse environment, as well as predict future conditions based on historical data. This data-driven approach will facilitate the optimization of greenhouse operations and support decision-making processes for crop management.

The back-end architecture will be designed to process and store the sensor data securely in a structured database, facilitating efficient data retrieval and analysis. The front-end application will provide an intuitive and responsive user interface that visualizes the sensor data in real-time, enabling users to interact with the digital twin and monitor greenhouse conditions at any time from remote locations. The application will offer customizable dashboards and data visualizations, allowing users to tailor the display to their specific needs and preferences.

Furthermore, the digital twin system will undergo rigorous testing and validation to ensure data accuracy and reliability. User training sessions will be conducted to familiarize stakeholders with the system's features and functionalities. Continuous monitoring and maintenance will be put in place to ensure the system's performance and to address any potential issues promptly.

By developing this digital twin of the greenhouse and incorporating data processing and machine learning analysis, the project will empower greenhouse operators and stakeholders to gain deeper insights into environmental conditions and plant health. It will facilitate data-driven decision-making, resource optimization, and sustainable agricultural practices, ultimately leading to increased productivity and efficiency in greenhouse operations.

## Terminology

**Graphical User Interface (GUI):** a type of interface that allows users to interact with a computer or device using graphical elements such as icons, buttons, windows, menus, and other visual elements, rather than using text commands. In Unity, GUI can be created using the built-in UI system, which allows developers to create and customize GUI elements easily using a visual editor or through scripting with the C# programming language.

## Reference Documents

*List any referenced document names or links.*

Please read this in conjunction with the following reference documents:

1. DENIM HVAC FDD GUI Requirements specification\_06102022.docx
2. AHU\_FDD\_Fault\_Descriptor.xlsx
3. AHU\_FDD\_Mapping.xlsx

## Unity 3D (2021.3.2f1 LTS)

Unity 3D is the selected development platform to achieve this due to its compatibility in publishing to other platform such as WebGL, Mixed Reality (HoloLens2) and Virtual Reality (Oculus Quest 2). It’s object-oriented programming (OOP) is ideal to enable any game objects or components to present consumable data.

For the initial version, WebGL is the format of the build which can be hosted in a server to run the application.

## Application Interface

Isometric view > zooming into various spots > Turn into First Person – walking around WASD

# 1 Project Management

## 1.1 Define Scope and Objectives – This is a Work Package

[Work Package Description]

The following are the list of tasks that need to be completed

* + Task 01
  + Task 02
  + Task 03

### 1.1.1 Task 01

[add text]

## 1.2 Identify stakeholders and their roles

### 1.2.1 Task 01

[add text]

### 1.2.2 Task 02

[add text]

### 1.2.3 Task 03

[add text]

# 2 Requirements Gathering & Analysis

This section delineates the requirements that were identified during stakeholder engagement, particularly within the technical team that consists of experts in data modelling, back-end development, and software platform integration. The primary goal of this phase is to establish the content as well the scalability and extensibility of the graphical user interface (GUI) that will be presented to the end-users.

## 2.1 Identify sensor data sources

To successfully implement the sensor-to-front-end project, the first crucial step is to identify sensor data sources. This involves thoroughly assessing the various data points required to create a comprehensive digital twin of the greenhouse. Key sensor sources include temperature sensors to monitor ambient and soil temperatures, humidity sensors to track moisture levels, light intensity sensors to measure illumination, soil moisture sensors to assess the water content in the soil, and CO2 sensors to monitor carbon dioxide levels. Additionally, other relevant data sources, such as weather APIs for external weather data or connected devices for irrigation and ventilation systems, might be considered for integration. The identification of these sensor data sources lays the foundation for data collection, processing, and analysis, enabling the system to provide real-time and accurate insights into the greenhouse environment, empowering users to make informed decisions and optimize greenhouse operations effectively.

### 2.1.1 Sensor Research and Selection

[add text]

### 2.1.2 Vendor Evaluation and Procurement

After evaluating various sensor vendors and analyzing their offerings, including specifications, pricing, and reliability, informed decisions were made regarding sensor procurement. The selection process considered factors such as accuracy, compatibility with the greenhouse environment, and communication protocols, resulting in the acquisition of sensors that align with the project's requirements and objectives. Please the specification sheets in Appendix [#]

### 2.1.3 Integration Planning

**<Define the integration strategy for the selected sensors. Determine how the sensors will be installed, calibrated, and connected to the data processing infrastructure. Consider factors like wiring, wireless connectivity, and power sources.>**

[add text]

<https://www.seeedstudio.com/Wireless-Smart-Agriculture-Kit-Greenhouses-p-4951.html>

### 2.1.4 Data Interfacing and Compatibility

**<Specify how the sensor data will be collected, transmitted, and received. Determine compatibility with data processing systems, ensuring seamless integration into the digital twin environment.>**

## 2.2 Gather functional requirements

Gathering functional requirements is a vital phase in our project management plan to develop a sensor-to-front-end interface for visualizing greenhouse sensor data. During this stage, we will engage with stakeholders, including greenhouse operators, agronomists, and end-users, to elicit and document the specific functionalities and features they expect from the system. This process involves identifying the key user interactions, data visualization requirements, and performance expectations. Additionally, we will define the desired data processing capabilities, such as real-time data updates, historical data analysis, and predictive insights through machine learning algorithms. By meticulously gathering these functional requirements, we will ensure that the resulting front-end interface meets the needs of our users and aligns with the project's objectives, fostering an intuitive and user-friendly experience to effectively leverage the potential of greenhouse sensor data for informed decision-making and enhanced agricultural practices.

For a detailed list of the functional requirements implemented in the software, please see Appendix #

### 2.2.1 User Interviews and Surveys

**<Conduct interviews and surveys with potential users of the desktop software to understand their needs and expectations. Gather insights into the specific features and functionalities they require.>**

### 2.2.2 Use Case Definition

**<Define various use cases that the software will address. Identify scenarios where users will interact with the application, specifying the inputs, outputs, and interactions involved.>**

### 2.2.3 Requirements Workshops

**<Organize workshops involving software developers, stakeholders, and end-users. Collaboratively define and prioritize functional requirements through brainstorming and discussions.>**

### 2.2.4 Prototyping and Feedback

**<Develop a functional prototype of the software with basic features. Present the prototype to stakeholders and gather their feedback. This iterative process helps refine the requirements.>**

### 2.2.5 Requirement Documentation

**<Document each requirement in detail, including user stories, acceptance criteria, and any relevant constraints. This documentation serves as a reference for the development team.>**

**\*\*\***Specific ways to make the graphical user interface (GUI) more intuitive for the end-users:

* Ensure a consistent design pattern throughout the application, including color schemes, typography, and layout.
* Simplify navigation by using clear and descriptive labels for menu items and buttons.
* Provide tooltips and help sections to assist users in understanding features and functionalities of the GUI.
* Use visual aids such as icons and images to convey information quickly and effectively.
* Conduct user testing and gather feedback throughout the development process to identify areas for improvement and enhance the overall user experience.
* Prioritize the most important information and actions to be displayed prominently, while minimizing clutter and unnecessary details.
* Incorporate user-friendly error messages that clearly explain the problem and provide guidance on how to resolve it.
* Consider the context and goals of the end-users to ensure that the GUI is optimized for their specific needs.
* Continuously evaluate the effectiveness of the GUI and adjust as needed to improve the user experience.

## 2.3 Gather non-functional requirements

### 2.3.1 Task

## 2.4 Analyse and prioritize requirements

# 3. Design & Architecture

## 3.1 Define data schema and storage requirements

## 3.2 Determine the technologies and frameworks for the frontend

### 3.3.1 Unity 3D Game Engine

Unity 3D is the most popular platform for developing games due to its flexibility and interoperability with multiple platforms. It also is ideal for rapid prototyping

### 3.3.2 Install Newtonsoft's Json.NET Package

On Unity go to Windows->Package Manager, once the Package Manager window opens, go to Add package from git URL, type **com.unity.nuget.newtonsoft-json** press Add and done

## 3.3 Design the front-end user interface

This work package focuses on the design and development of the envisioned software application. There are multiple elements that considered important for the user to have a seamless experience with the

### 3.2.1 Camera System

Using the package

1. **Installing Cinemachine -** Use the Unity Package Manager (in the top menu: Window > Package Manager) to select Cinemachine for installation. If you don't see the package, look for a dropdown menu above the list and select “All packages”. Cinemachine is free and available for any project. If you already have it installed, you may update to the latest version. See link - <https://unity.com/unity/features/editor/art-and-design/cinemachine>
2. Once Cinemachine is imported into the scene, you can create a Virtual Camera game object.
3. The Virtual Camera can be set to ‘Look at’ and ‘Follow’ an object
4. Create a script that manages the switching between the views/Virtual Cameras

### 3.2.2 Tooltip System

* Designing a tooltip system to draw and display tooltips.
* Tooltip trigger component to handle interaction
* Dynamically size tooltip based on contents

1. Setup a separate canvas (Tooltip Canvas) in order for the panels to be on top of anything in the UI.
2. Set the sort order to the highest value to ensure it’ll render on top.
3. Add Canvas Scaler component and set the UI Scale Mode to ‘Scale With Screen Size’ with the reference resolution as 1920 x 1080.
4. Add an Image child game object which will contain the two TextMeshPro components. Adding a Vertical Layout Group with Control Child Size enabled.
5. Add Content Size Fitter component with Horizontal and Vertical Fit as ‘Preferred Size’
6. Add a Layout Element component with a Preferred Width of 500 to control the size of the text length.

## 3.4 Design the back-end architecture for data processing

The back-end is built using the Python Flask framework to create the APIs and MySQL for the database.

Python Flask is a micro web framework for building web applications in Python. It is designed to be lightweight and simple, providing the essentials needed to create web applications while allowing developers the flexibility to choose their tools and libraries for other components. <https://flask.palletsprojects.com/en/2.3.x/>

The Flask program is built as follows:

- “app.py” which is used to run the application and import the other files

- “auth.py” is the Python script that contains the ‘User’ class and the endpoint, “/api/users” which is used to create an account by posting a JSON with a ‘username’ and a ‘password’. This file also has the endpoint “/api/login” which is used to log in to an account by sending the same JSON than the account creation endpoint.

- There is also the “sensors” folder in the project tree. It contains all the sensors files.

- Each of the files inside the sensors folder are used to create several endpoints to access the sensor values that are stored in the database.

The following endpoints are contained in “sensorLightExt.py” which are as follows:

* “/api/sensors/sensor\_light\_ext/latest” – this is the endpoint to retrieve the latest value.
* “/api/sensors/sensor\_light\_ext/day/pic-average” – this endpoint returns the maximum light intensity during the day, the maximum light intensity during the night, and the average of these two values for the current day.
* ”/api/sensors/sensor\_light\_int/day/average” – this is the endpoint to retrieve the data of the last 24 hours, each hourly value is the average of the ten-minute interval.
* “/api/sensors/sensor\_light\_int/day” - this is the endpoint to retrieve all the data of the last 24 hours.
* “/api/sensors/sensor\_light\_ext/week” – this is the endpoint to retrieve the data of the last seven days. A list of the average of each hour over the last seven days is returned.
* “/api/sensors/sensor\_light\_ext/month” – this is the endpoint to retrieve the data of the last 30 days. Each day is divided into six 4-hour periods which are averaged.

This program uses an ORM (SQLAlchemy), which allows database tables to be created directly from classes created in Python.

The databases tables are simple. For the User table, there is an ID which is autoincrementing and which is the primary key, there is also a username and a password\_hash column. The password in encrypted with “werkzeug.security”.

The sensors tables are simple too. They have got an ID which is also autoincrementing and which is also the primary key, there is one or more value column depending of the sensor and a datetime column.

## 3.5 Plan data integration and APIs between sensors and the front-end

Sensors use LoRa to communicate. LoRa (Long Range) is a wireless technology for the Internet of Things (IoT), offering long-range coverage and low power consumption.

Sensors therefore transmit data frames containing information such as values, time, etc.

This data is received by a gateway, which then sends it to the Internet, in our case to TTN (The Things Network).

The Things Network registers the gateway as well as the various sensors.

The data is then sent via a webhook to the Flask backend application, where it is processed.

## 3.6 Server deployment

The backend (Flask and MariaDB) is hosted on a NUC Dell EMC Edge Gateway 3200.

Docker is used to deploy these services.

Docker is a lightweight virtualization technology for running portable, isolated applications in container form.

Since the NUC isn't just used to host this project, Docker makes it possible to keep each project running on the NUC separate.

**How to use the NUC :**

An ethernet cable must run from the NUC's first ethernet port to an internet router and a second cable from the NUC's 2nd ethernet port to the NAS's first ethernet port (LAN1).

Login : server

Password : 1MTUr3s34rch4DM1N ( same password for admin)

All the projects are in home/server/docker-containers/

Files shared with the NAS are in home/server/nas and home/server/bathorse

Docker commands :

* docker ps -> used to see what container is running
* docker-compose build -> build the project
* docker-compose up -d -> run the containers
* docker-compose down -> turn off the containers

docker ps can be used anywhere

All the docker-compose commands must be used in a directory which contains a docker-compose.yml file

**Ports used :**

|  |  |
| --- | --- |
| **App** | **Port** |
| Greenhouse frontend | 8079/TCP |
| Greenhouse backend | 8080/TCP |
| Projects page | 8085/TCP |
| VIB backend | 8084/TCP |
| Bathorse | 3005/TCP |
| BathorseUDP | 5550/UDP & 5555/UDP |

# 4 Sensor Data Acquisition

## 4.1 Procure necessary sensors and hardware

This project uses 8 LoRa sensors from Seeed :

* 2 Light intensity sensors
* 2 CO2, Temperature, Humidity sensors
* 4 Soil Moisture, Temperature, electro conductivity sensors

## 4.2 Develop or configure data collection software

The gateway sends sensor data to the TTN.

**Sensors Configuration**

To receive data from LoRaWAN sensors, each device must first be registered on **The Things Network (TTN)**.

**Steps to Register a Sensor on TTN:**

1. **Log in** to The Things Stack Console.
2. Navigate to your **application** or create a new one if needed.
3. In the application dashboard, click **"Add end device"**.
4. Choose the **manual option** to register the device manually.
5. Enter the following information:
   * **End device ID**: a unique name you choose for the sensor (e.g., light-sensor-ext1)
   * **Join EUI** (also known as App EUI): provided by the manufacturer (Seeed Studio)
   * **Dev EUI**: unique identifier of the sensor, printed on the device or provided by Seeed Studio
   * **App Key**: also provided by the manufacturer, used for OTAA activation
6. Select the **LoRaWAN version** and **regional parameters** (e.g., LoRaWAN 1.0.3, EU868 for Europe).
7. Finalize the registration and **save** the device.

Once registered, the sensor will be able to join the network via **Over-The-Air Activation (OTAA)**, and its messages will appear in the **Live Data** tab under the application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **ID** | **AppEUI** | **DevEUI** | **AppKey** |
| Light intensity out | eui-2cf7f1c04430094f | 11E62CA65AAD4B92 | 2CF7F1C04430094F | C48EDA4F132DC08FB6C739955B058E41 |
| Light intensity in | eui-2cf7f1c044300975 | 263F0C0CC52FAB2B | 2CF7F1C044300975 | 1B7C61DCE711FDD645D2AC9C3E444A59 |
| CO2, Temp, Hum out | eui-2cf7f1c044300436 | D87673FE226C96F3 | 2CF7F1C044300436 | B82D2A732905A8C7A75B7F3C2B635C78 |
| CO2, Temp, Hum in | eui-2cf7f1c0443004b1 | 8ABB1B9C9273EBF1 | 2CF7F1C0443004B1 | BE41B363444195613265D046DEF22A6C |
| Soil Moisture, Temp, EC 1 | eui-2cf7f1c0435006c8 | 8BAC255955815EB5 | 2CF7F1C0435006C8 | EC575A8F8893807DE07CD3CAE968CB7B |
| Soil Moisture, Temp, EC 2 | eui-2cf7f1c043500707 | 5E9C5CECCBDB86C4 | 2CF7F1C043500707 | E3EAA1A54F2A33E28618A49B814D3DD0 |
| Soil Moisture, Temp, EC 3 | eui-2cf7f1c043500681 | FAC375674CD1FD1C | 2CF7F1C043500681 | 2279F5260EA7B55D55F2C46599DAEBA9 |
| Soil Moisture, Temp, EC 4 | eui-2cf7f1c0435005e6 | C645419B93B20198 | 2CF7F1C0435005E6 | 34D1BF4CAF20FC39FF0ABD6254E5D0F5 |

**Webhook configuration :**

One of the key steps in ensuring that sensor data is transmitted to the backend is the configuration of a webhook on The Things Network (TTN). This webhook automatically forwards decoded LoRaWAN messages from TTN to the server responsible for processing and storing them (our Flask backend).

**Webhook Objective**

The goal is to send the sensor data received by TTN to a publicly accessible URL that the Flask backend listens to. In a production environment, this URL typically points to a public IP address or domain name associated with a server.

**Issue Encountered**

In my case, I did not have access to the router’s administration interface, which prevented me from configuring port forwarding to my local machine. As a result, TTN could not reach my backend directly.

**Solution: Using Ngrok**

To bypass this network limitation, I used Ngrok, a tool that creates a secure tunnel between a local port and the internet. Ngrok generates a temporary public URL (e.g., https://xxxx.ngrok.io) that forwards requests to a local server (e.g., http://localhost:8080).

**Configuration Steps Using Ngrok:**

1. Start the Flask backend locally (port 8080):
2. python app.py
3. Launch a tunnel using Ngrok:
4. ./ngrok http 8080
5. Ngrok will generate a public URL, such as:
6. https://xxxx.ngrok.io
7. In the TTN console, go to Webhooks > Add Custom Webhook.
8. Enter the Ngrok public URL followed by /api/ttn as the base URL:
9. https://xxxx.ngrok.io/api/ttn
10. Ensure that the Flask backend has a corresponding route, for example:
11. @app.route('/api/ttn', methods=['POST'])
12. Send test data from TTN and verify that the backend receives and processes the messages correctly.

**Important Notes:**

* Ngrok URLs expire each time it restarts unless you use a paid plan with a reserved domain.
* You’ll need to update the webhook URL in TTN every time Ngrok is restarted during development.

**Une image contenant texte, capture d’écran, logiciel, nombre

Le contenu généré par l’IA peut être incorrect.**

**Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement**

## 4.3 Test data acquisition system and validate sensor data

**Step 1 – Data Reception and Decoding**

The gateway forwarded the sensor payloads to TTN, which then relayed them via a webhook to my Flask backend. Using the ttn.py script, I decoded the incoming payloads based on the device ID and ensured proper parsing of each data field. I also checked the timestamp formatting and the insertion process into the appropriate database tables.

**Step 2 – Real-Time Visualization**

With the full pipeline in place — from sensors to TTN, to the backend, to the frontend — I was able to visualize real-time data on the dashboard. Values were displayed instantly under each section (Luminosity, Environment, Soil), and the data graphs updated correctly according to the selected period filters (Day, Week, Month).

**Validation Result**

The successful real-time visualization of sensor data validated the entire architecture. I checked:

* The accuracy and completeness of received payloads.
* The stability of the database insertions.
* The responsiveness and clarity of the web interface.
* The correct operation of the filtering system per sensor and time period.

This validation confirmed that the system is ready for deployment and future extension, once all sensors are installed around the greenhouse and fully operational.

# 5 Data Processing and Storage