Journal of Ambient Intelligence and Smart Environments 16 (2024) 389–405
DOI 10.3233/AIS-230359
IOS Press

GreenhouseGuard: Enabling real-time warning prediction for smart greenhouse management

Juan Morales-García [a,*], Diego Padilla-Quimbiulco [a], Magdalena Cantabella [a], Belén Ayuso [a], Andrés Muñoz [b] and José M. Cecilia [c]
[a] *Computer Science Department, Catholic University of Murcia (UCAM), ES, Spain*
E-mails: jmorales8@ucam.edu, depadilla@alu.ucam.edu, mmcantabella@ucam.edu, bayuso@ucam.edu
[b] *Departament of Computer Engineering, University of Cádiz (UCA), ES, Spain*
E-mail: andres.munoz@uca.es
[c] *Computer and Systems Informatics Department, Universitat Politècnica de València (UPV), ES, Spain*
E-mail: jmcecilia@disca.upv.es

Received 6 September 2023
Revised 19 February 2024

Abstract. Greenhouses constitute intricate systems where numerous variables play a pivotal role in enhancing crop yields within the framework of intensive agriculture. Consequently, real-time monitoring and visualization of these variables are imperative to strike a balance between resource efficiency and production maximization. Furthermore, the ability to make predictive assessments regarding these variables is essential to avert potential greenhouse disasters. In this article, we introduce an intelligent alert system designed to efficiently oversee agricultural operations within a functioning greenhouse, ultimately bolstering productivity through the optimization of crop growth and energy consumption. This system comprises a web application, GreenhouseGuard, which improves the graphical and statistical representation of data collected by a network of sensors strategically positioned throughout the greenhouse, as well as the forecasts generated from this data. These sensors are strategically located to provide more precise real-time data readings, thereby minimizing error margins. Moreover, GreenhouseGuard offers diverse data visualization options and forecasts of greenhouse variables to enable in-depth analysis of the acquired information. Consequently, this alert system empowers greenhouse managers to proactively address abnormal situations that may jeopardize their crop yields.

Keywords: Artificial intelligence, machine learning, temperature forecasting, warning system, smart greenhouses

1. Introduction

Agriculture is an activity of great importance in society. This field has experienced a variety of developments, particularly in the techniques and instruments employed. These elements, integral parts of the wider technological spectrum, have undergone significant evolution, primarily driven by advancements in technology. In this area, one of the most relevant lines of research is focused on providing assistance in the management of greenhouses [19].

In recent years, greenhouse agriculture has been steadily increasing almost everywhere on the planet. It is a fact that this type of agriculture is an important pillar in today's society by providing food resources to millions of

[*] Corresponding author. E-mail: jmorales8@ucam.edu.

people around the world and allowing the development of countries [11]. The problems that greenhouse managers may face are multiple and varied, for example, the climatic conditions, the use of natural resources, pollution, energy consumption, fertigation, etc. However, one of the biggest issues faced by these greenhouse managers is to optimally control ventilation and air circulation, as both are of great importance for the growth and well-being of the plants. Greenhouses normally have ventilation systems that help control humidity and temperature, but these systems can occasionally fail, either because they need maintenance or simply because of a one-off failure. For this reason, great importance is taken in greenhouse maintenance where several variables come into play such as outside temperature, heating temperature, ventilation temperature, etc. [18] These variables are obtained by means of specific sensors. With the advent of IoT in the agri-food industry, the monitoring of greenhouse conditions and the possibility of creating monitoring systems for smart greenhouses is facilitated [6].

In this context, this work introduces an Intelligent Monitoring System (IMS) for greenhouses, which allows total control over the current greenhouse status. It enables the monitoring of all the variables collected from a set of IoT sensors deployed in the greenhouse, so as to establish an alert system when an anomaly arises. This work also lays the foundation for future forecast of alert situations thanks to the integration of Artificial Intelligence in the system. In this regard, the developed system can be helpful to Autonomous Management Systems (AMS), aiding in understanding and monitoring everything that occurs within the automated environment in a more straightforward manner. Additionally, it establishes a solid base for future migratation to an AMS for greenhouses, which will allow for the self-management of greenhouses through advanced Artificial Intelligence models.

In particular, this system provides a personalized interactive web application that allows visualizing the data collected from the sensors inside a greenhouse to be exposed in a simple manner and to alert on abnormal situations. This solution is framed in the perspective of "environmental intelligence" by providing an intelligent and proactive approach to improve the interaction between the greenhouse environment and the users. Our system aims to help greenhouse managers to get an overview of what is happening in the greenhouse with just one click. Also, with real-time notifications, the greenhouse manager can act quickly on unexpected changes. Currently, although there are tools that allow a comprehensive management of a greenhouse, there is none that fits our study scenario (described in Section 3.2) since it is a very specific and changing sector in terms of variables such as crop type, season, etc. Therefore, the need arises to create a smart system that allows a user to manage a greenhouse from a web page without the need to check each sensor individually. The alert system proposed in this paper provides the ability to set an optimal measurement range for each sensor. This allows the user to receive real-time notifications if any measurement is outside the previously established range, thus allowing the user to have a quick reaction time.

Real time notifications are based on the data received by the sensor at the moment. However, it would be better to get alerts based on things that may happen, for example, a sudden change in the weather where in one hour the temperature drops 10 degrees, etc. For this, a prediction model has been added that is responsible for reading the data already stored in the sensor to generate future data and thus be able to predict the value that can be had within an hour and warn the farmer if it is necessary to make changes to prevent damage to their crops.

The main contributions of the paper include the following:

- Development of an interactive web application that monitors in real time all the variables collected by IoT sensors in a smart greenhouse.
- Displaying both graphically and numerically each of the variables collected by the sensors. In addition, implement an alert module to warn of abnormal situations collected by IoT sensors.
- Implement a Machine Learning (ML) model, namely Autoregressive Integrated Moving Average (ARIMA), to make predictions of the temperature inside the greenhouse and alert of possible incidents before they occur, in order to avoid possible catastrophes in the crops inside the smart greenhouse.

The rest of the paper is structured as follows. Section 2 shows related work that focuses on showing other greenhouse monitoring systems already available and how they differ. Section 3 shows all the tools used for the development of this proposal as well as its architecture. Section 4 the section shows the results obtained and the discussion of these results. Finally, Section 5 presents the main conclusions and discusses future works.

2. Related works

Several examples of greenhouse monitoring software development can be found in the literature. Some of the proposals are based on wireless sensor networks (WSNs) connected to a local network for data collection, as in [3], where authors presents a WSN prototype consisting of MicaZ nodes which are used to measure greenhouses' temperature, light, pressure and humidity. WSNs also provide benefits such as battery-powered devices that can sense information, process it locally and transmit it to the destination using low-power ad hoc wireless technologies, as shown by the authors of [22] where an approach to the design and performance analysis of a flexible greenhouse monitoring wireless sensor module, based on general purpose microcontrollers and low power ZigBee communication modules is presented. These solutions need expert knowledge of programming and chip management, which do not make them friendly to users who have basic knowledge and want a quick solution. Moreover, they usually are implemented as desktop applications, therefore being limited to running on certain types of devices.

There are other approaches that consider the implementation of a complete IoT system based on Raspberry Pi boards and sensors able to connect directly to the board in order to have a global system that can monitor a specific plant. Generally, these solutions tend to be developed for a specific type of problem-related to the greenhouse. In addition, the web implementation of this solution is quite basic in terms of the way in which the data is displayed, as the main effort is carried in the technical part of hardware rather than in the development of software that can complement the data collected by the system [10].

Moreover, it must be taken into account that many of the systems currently developed are focused on a certain type of crop since in the world of agriculture each crop is special and must be treated in a different manner. Some solutions even tend to use cameras to be able to analyze crops in a more comprehensive way through the use of neural networks. This development is very specific to crop research and would be very difficult to replicate on a large scale and would require a great deal of development to handle all the data obtained. It is worth noting that not only tools have been developed to check the state of a greenhouse, but the tools, when based on a crop, focus on notifying the user if the crop is ready to be cultivated [15].

Other implementations like GRETAs [5] focus more on building fully intelligent greenhouses. This option is interesting as it provides the user with full control over the greenhouse. His vision is based on offering the user a full view of the crops and the greenhouse through an augmented reality application. This offers the analysis of the plants, their form of growth, and tips for pests or to help the growth of the plant. But being augmented reality, it takes a lot of resources to build the app and to turn the greenhouse into a smart home that communicates with the device and gives it the correct readings, such as, for example, an expansive sensor array. It also has a web interface that allows the user to get an overview of the greenhouse and different sections that show sensor data. This approach is interesting, but it must be taken into account that it is a tailor-made development, so the addition of new greenhouses or new clients that would like to use the application would entail a large cost of management and resources.

These applications have a more exhaustive development to alert the user as they are ad-hoc software and can rarely use add-ons made by the community to achieve these objectives. In terms of performance, it may vary a bit because these solutions tend to implement their own servers which means that they manage the load and handle it as they wish while the application proposed in this paper uses third-party services to alert the user and depends entirely on the availability that the third party has for the project.

In terms of predictions, several solutions have been implemented such as GCP_ LSTM models for greenhouse climate prediction [16]. This solution contemplates that the climate change inside a greenhouse is not linear and therefore uses a short-term model to capture the dependence between historical climate data. Furthermore, this solution considers that the short-term climate has a greater impact on the future trend of climate change within the greenhouse. On the other hand, the Ref solution considers that the usual prediction methods are inefficient and proposes a solution using recurrent neural networks (RNN) with short-term memory. This model evaluates different environmental parameters of the greenhouse to predict them for a whole year [4]. There are also more complex solutions that are not only based on the data obtained by the sensors but also based on their location, distance between them, greenhouse measurements, etc. This solution uses a regression model based on a dense neural network (DNN) which is a more complex model that is based in a regression model and uses different dense channels to predict the values [2].

On the other hand, there are some solutions that have chosen to focus on individual plants and create an intelligent system around them. These systems, such as Plants for Life "P4L" [12], automate the care of potted plants to improve air quality and make a building's indoor environment healthier. This is a valid approach to an intelligent system but leaves out the plants that are used for cultivation as it is intended for building interiors and can be very complicated to adapt to the agriculture industry, especially because of the ad hoc development of IoT system components.

Likewise, there are solutions that also use the ARIMA model for carbon emission predictions, albeit with some modification and assistance from R3det, which is responsible for determining objects by rotary remote sensing and Beidou satellite navigation [17]. This helps to locate vehicles and buildings in order to predict the carbon emissions they generate. This experiment was carried out in Tangshan Industrial Park, Hebei-China where a successful prediction of carbon emissions was made. This project is interesting because although it is aimed at cities, it could also be directed to the field of agriculture due to the carbon released by plants and especially if they are inside a closed greenhouse, this carbon is affected in the long term, which would help to predict the level of carbon and to control it with the help of sensors and alerts as is done in the implementation of the intelligent system of the project mentioned in this article.

All these solutions offer diverse approaches to collecting and handling data. Most involve creating custom sensor systems with programming for data acquisition. Additionally, a network setup is necessary to link the central board with all sensors. In line with this, our focus is on enabling users to access specific sensor data that's pre-built, requiring only configuration for data transmission. We utilize modern frameworks for swift web app development, ensuring efficient site processing and user-friendly interfaces. Unlike many existing solutions, which lack user-friendliness for non-tech-savvy users, our system prioritizes presenting data understandably for decision-making, as well as a simple graphical interface that allows access to any component of the system (data, sensor, forecast, alert, configuration, etc.) following Universal Design principles for the web [9]. It's adaptable for future tech integration and accessible across Internet-connected devices. Notably, our system stands out for providing real-time notifications of changes in measurements, an advanced feature that is often complex due to the nuances of server-client communication. Our setup uses HTTP for communication between the front-end and back-end, requiring constant listening for messages sent by the server to update specific sections of the application – a form of reactive programming. This is achieved by implementing a real-time protocol that sends data from IoT devices to the back-end and, in turn, implementing sockets in the back-end, facilitating bidirectional communication between the front-end and back-end.

3. System architecture

This section describes the background tools used to develop the web application, the dataset used and the artificial intelligence model implemented for real-time predictions in the smart greenhouse.

Figure 1 shows the architecture of the proposed alert system for monitoring greenhouses. It results in an application that can be used in any device regardless of its operating system that connects to a database and sensors in order to provide real-time data awareness of the actual state of a greenhouse.

Everything is connected through the Internet. The main actor is the server since it is in charge of receiving the data from the sensors, processing it and sending it to the database. Likewise, the server is in charge of managing the HTTP requests from the client to obtain the latest measurements and the history of each sensor. Finally, there is a direct connection between the client and the server because the client must be actively listening if the server issues a message about an alarm, when the client receives the alarm, it notifies the server that it has been received and avoids a double propagation of alerts.

The system has a landing dashboard that allows a quick overview of all the sensors data in a glance as shown in Fig. 2. On this home page the latest measurements of the different sensors that the greenhouse has are shown. Each measurement has its specific unit, for example: Outdoor humidity ("HumedadExterior") is shown as a percentage since this is how this measurement is measured. Likewise, sensors such as the rain alarm ("AlarmaLluvia") are shown as on or off since they only collect data on the possibility of rain. A message is also displayed showing how long ago that measurement was obtained ("Last Updated") in order to notify the user about the time that has passed since the last time a measurement was received in any sensor.

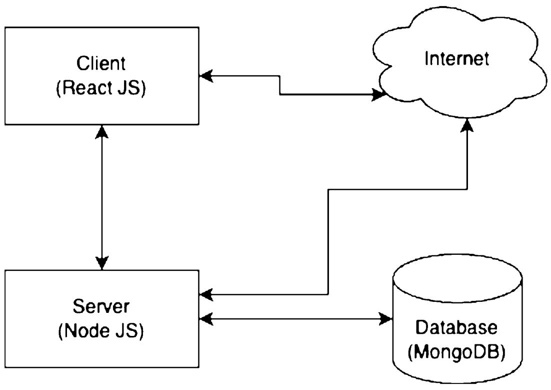
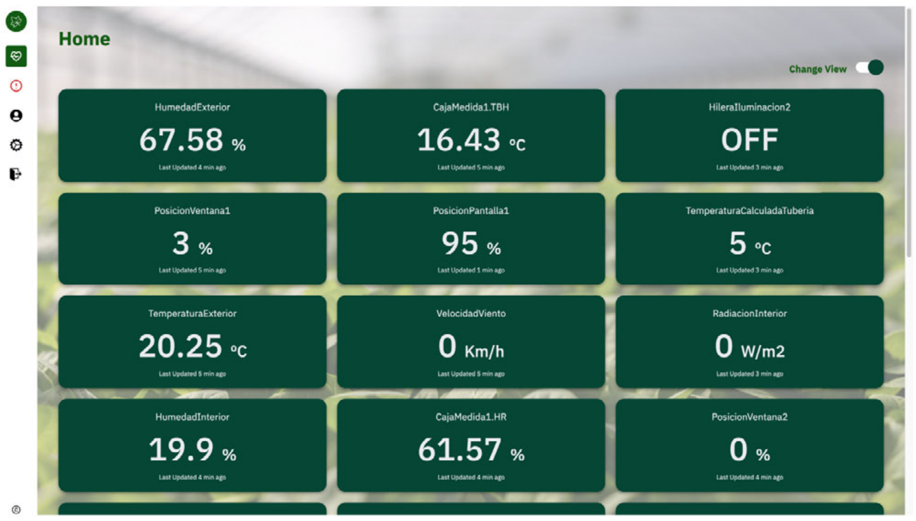


Fig. 1. Web flow schema.



HumedadExterior
67.58 %
Last Updated 4 min ago

CajaMedida1.TBH
16.43 °C
Last Updated 5 min ago

HileraIluminacion2
OFF
Last Updated 3 min ago

PosicionVentana1
3 %
Last Updated 5 min ago

PosicionPantalla1
95 %
Last Updated 1 min ago

TemperaturaCalculadaTuberia
5 °C
Last Updated 3 min ago

TemperaturaExterior
20.25 °C
Last Updated 5 min ago

VelocidadViento
0 Km/h
Last Updated 5 min ago

RadiacionInterior
0 W/m2
Last Updated 3 min ago

HumedadInterior
19.9 %
Last Updated 4 min ago

CajaMedida1.HR
61.57 %
Last Updated 4 min ago

PosicionVentana2
0 %
Last Updated 4 min ago

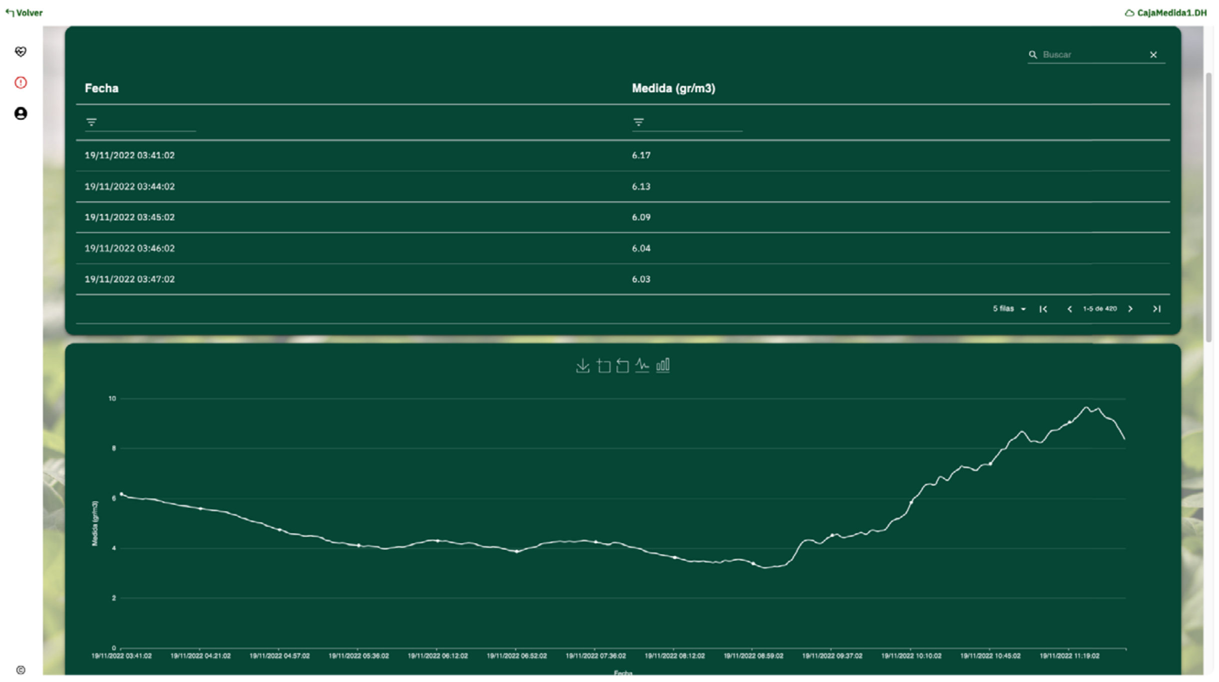
Fig. 2. Home page.

Figure 3 shows data representation is an important part of the application as it allows visualization in different ways to help the user understand more about the data that has been collected over a period of time. Within the table the user can find two columns date and the measurement accompanied by the respective unit of that sensor. This table allows filtering of the data and adds a listing perspective that is more familiar to the user. Likewise, it offers the functionality to search both by measurement or by date.

As a second data representation, there is a linear graph that shows the evolution of the measurements over time. This graph shows the date as the horizontal axis and the measure as the vertical axis. When the user has only selected one day, the history is shown for each record on that day, which are usually records obtained every five minutes by the sensor. If the user chooses several days, the form of representation will be changed and the data grouped by day will be displayed. Likewise, if the user decides to filter between two months, it is grouped by each month.

3.1. Backend tools

The backend tools used are described in Table 1.



Fecha	Medida (gr/m3)
19/11/2022 03:41:02	6.17
19/11/2022 03:44:02	6.13
19/11/2022 03:45:02	6.09
19/11/2022 03:46:02	6.04
19/11/2022 03:47:02	6.03

Fig. 3. Data page.

3.2. Study scenario

The operational greenhouse targeted in this study, referred to as "ETIFA", is shown in Fig. 4. ETIFA is a functioning greenhouse owned by NUTRICONTROL, a Spanish company focusing on the development of climate control and automatic fertigation technology. Covering a surface area of 50 m^2, it is located in Murcia, a semi-arid region in the southeast of Spain, with an average yearly temperature of approximately 25 ºC. Inside the ETIFA greenhouse, a modular IoT infrastructure for climate control and fertigation system is deployed. This IoT infrastructure is coordinated by the OPTIMUM$^{®}$ integrated control system.

3.3. Sensor configuration

The architecture of IoT sensors installed in the greenhouse, as well as their characteristics, is shown below.

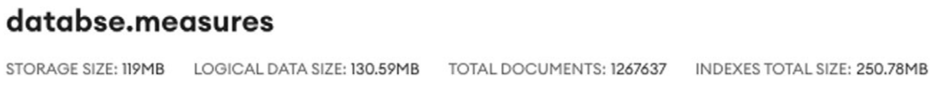
- **CajaMedida1.TBS:** The temperature inside the greenhouse is measured by a dedicated thermometer. It is an internal variable of the greenhouse with a range of values: [9.573–33.724] °C.
- **CajaMedida1.TBH:** The humid temperature inside the greenhouse is measured by a dedicated thermometer. It is an internal variable of the greenhouse with a range of values: [8.323–29.678] °C.
- **CajaMedida1.HR:** The percentage of humidity inside the greenhouse is an internal variable with a range of values: [53.257–100.000] %.
- **CajaMedida1.DH:** This represents the lack of irrigation of crops in the greenhouse and is an internal variable with a range of values: [0.000–10.611] gr/m^3.
- **TemperaturaExterior:** The temperature outside the greenhouse is an external variable with a range of values: [2.465–32.689] °C.
- **HumedadExterior:** The humidity outside the greenhouse is an external variable with a range of values: [11.398–100.000] %.
- **DireccionViento:** The wind direction outside the greenhouse is an external variable with a range of values: [46.357–296.571] °.

Table 1
Backend tools

Tool	Description
ReactJS	It is a front-end JavaScript library that employs a development logic of reusable UI components. It allows visual development for large and complex web applications that need to display data changes without the need to refresh the page. In addition, it generates good performance compared to other libraries that consume a lot of computer and server resources [1]. This library will be the core of the application since all the data collected from the sensors will be shown here and, in addition, it will be in charge of connecting with the server to obtain notifications.
NodeJS	It is a JavaScript execution environment that allows handling asynchronous events within a server in real time. In addition, it allows the creation of fast web applications, since it allows the handling of many simultaneous connections without losing performance [23]. It will be in charge of executing the server that will collect the data from the sensors and will also perform tasks for data formatting and internal comparison of values to generate alerts.
ExpressJS	It is a Node JS framework that allows better manage of requests within a server. What Express allows is to handle HTTP requests from a client and return a response which means it can communicate with any device that speaks the language of the internet [7]. Thanks to its ease of implementation, Express will allow the creation of an API so that the client is able to obtain the necessary data.
MongoDB	It is a NoSQL database system that is oriented to document management that stores data in a JSON-like format called documents. This type of database is really useful in applications where large amounts of data are expected to be received, as they are easier to process compared to a relational database [13]. It will be the main database where a history of the sensor readings will be kept and where all the sensor and application configurations will be stored.
Socket.IO	It is a library that allows low latency bidirectional communication between client and server. Socket.IO has more functionality than an easy to use Web-Sockets API as it provides the ability to use other real-time protocols if sockets are not available as some browsers may not support Web-Sockets [8]. It is the most important technology since it is what allows a real-time connection between the server and client. Therefore, it allows the notification of alerts to the user.
MQTT	It is a M2M (machine-to-machine) communication protocol based on TCP/IP as the basis for communication. MQTT is a standardized publish/subscribe protocol that was developed with the idea of sending accurate data over a slow connection network. The protocol is based on users subscribing to topics to receive messages published by a client [21].This technology will be used to send data from the sensors to the server. The connection of the sensors will provide a subscription to read data periodically.
Mailjet	It is a mailing platform that allows a fast integration in different web projects to send web mails to users in a fast and secure way. It is used to send notifications to the user through the mail. This platform will allow the sending of emails to the user within the platform and the default user.

- **VelocidadViento:** The wind speed outside the greenhouse is an external variable with a range of values: [0.000–15.231] Km/h.
- **Radiacion:** Solar radiation received by the greenhouse is an external variable with a range of values: [0.000–884.231] W/m^2.
- **CO2:** The CO_2 reading inside the greenhouse is an internal variable with a range of values: [363.500–408.933] Ppm.
- **PosicionPantalla1:** The first screen inside the greenhouse is an internal variable with a range of values: [0–100.000] %.
- **PosicionPantalla2:** The second screen inside the greenhouse is an internal variable with a range of values: [0–100.000] %.
- **HileraIluminacion1:** The first illumination level inside the greenhouse is an internal variable with a range of values: [On, Off].
- **HileraIluminacion2:** The second illumination level inside the greenhouse is an internal variable with a range of values: [On, Off].
- **Humidificacion:** The level of humidification inside the greenhouse is an internal variable with a range of values: [On, Off].
- **Destratificacion:** The level of destratification inside the greenhouse is an internal variable with a range of values: [On, Off].
- **AlarmaLluvia:** The raining alarm outside the greenhouse is an external variable with a range of values: [On, Off].

Fig. 4. ETIFA: NUTRICONTROL's operational greenhouse located at Murcia (Spain).



databse.measures

STORAGE SIZE: 119MB LOGICAL DATA SIZE: 130.59MB TOTAL DOCUMENTS: 1267637 INDEXES TOTAL SIZE: 250.78MB

Fig. 5. Dataset.

– **AlarmaTormenta:** The storm alarm outside the greenhouse is an external variable with a range of values: [On, Off].

3.4. *Dataset*

The dataset used for the measures prediction contains all the readings from all the sensors such as humidity, temperature, rain, hydric deficit, storm, destratification, lighting, wind, humidification, radiation and carbon dioxide. This dataset is filtered based on sensor type as each sensor contains very different values and ranges that can affect the predictions. This dataset is not filtered between dates as it is considered that the model should use all the data available to make an accurate prediction with a small margin of error. In addition, with the climate change in recent years, these differences must be taken into account in order to train the model correctly. The datasets of each sensor can be grouped in different ranges, either by hour, day, week or month, depending on the value to be predicted.

This dataset consists of more than one million records that increase every five minutes that data is received from the sensors, as shown in Fig. 5.

The short time difference in the records is of great help because when using the data for prediction, values closer to the real ones can be obtained. These are some examples:

```
{
        "value": {
                "$numberDouble": "22.34"
        },
        "dateTime": {
                "$date": {
                        "$numberLong": "1692369421000" // 2023-08-18T14:37:01.000+00:00
                }
        },
        "type": "CajaMedida1.TBH",
        "id": "temperature"
}
```

```
{
        "value": {
                "$numberDouble": "22.35"
        },
        "dateTime": {
                "$date": {
                        "$numberLong": "1692369481000" // 2023-08-18T14:38:01.000+00:00
                }
        },
        "type": "CajaMedida1.TBH",
        "id": "temperature"
}
```

```
{
        "value": {
                "$numberDouble": "22.58"
        },
        "dateTime": {
                "$date": {
                        "$numberLong": "1692369601000" // 2023-08-18T14:40:01.000+00:00
                }
        },
        "type": "CajaMedida1.TBH",
        "id": "temperature"
}
```

The short difference in the timestamp allows the ARIMA model to make a precise prediction of the future sensor values.

3.5. Prediction model

The ARIMA model [20] is a statistical technique used in time series analysis that allows a value to be represented as a linear combination of prior data and random errors. The term ARIMA is an abbreviation for AutoRegressive Integrated Moving Average, which refers to a statistical model that uses variations and regressions of statistical data to identify patterns and make future predictions. These predictions are based on past data rather than independent variables. It is recommended that this model have at least 50 observations or time series data to obtain accurate predictions. The model is defined by the formula (1), where p represents the parameters corresponding to the autoregressive (AR) part, d indicates the differences needed to make the original series stationary, and q represents the parameters corresponding to the moving average (MA) part of the model [14].

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \epsilon_t - \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \tag{1}$$

Where:

- Y_t: Observation at time t.
- c: Constant or intercept term, representing the mean level of the time series.
- ϕ_i: Autoregressive coefficients that weight the previous Y_{t-i} observations. These capture the linear relationship between the current observation and its past values.
- Y_{t-i}: Observations in the previous p periods.
- ϵ_t: Error term at time t, representing the difference between the model prediction and the actual observation.
- θ_j: Moving average coefficients that weight the above error term ϵ_{t-j}. These capture the linear relationship between the current error term and its past values.
- ϵ_{t-j}: Error terms in the previous q periods.

The ARIMA model is created with the order (5, 1, 3). This means that the model uses 5 past values (autoregressive term), differences the series once to make it stationary (differencing term), and uses an error term of 3 past forecast errors (moving average term). The model is then fitted to the data. This is where the model learns the underlying patterns in the data and once the model is fitted, it's used to make predictions.

4. Evaluation and discussion

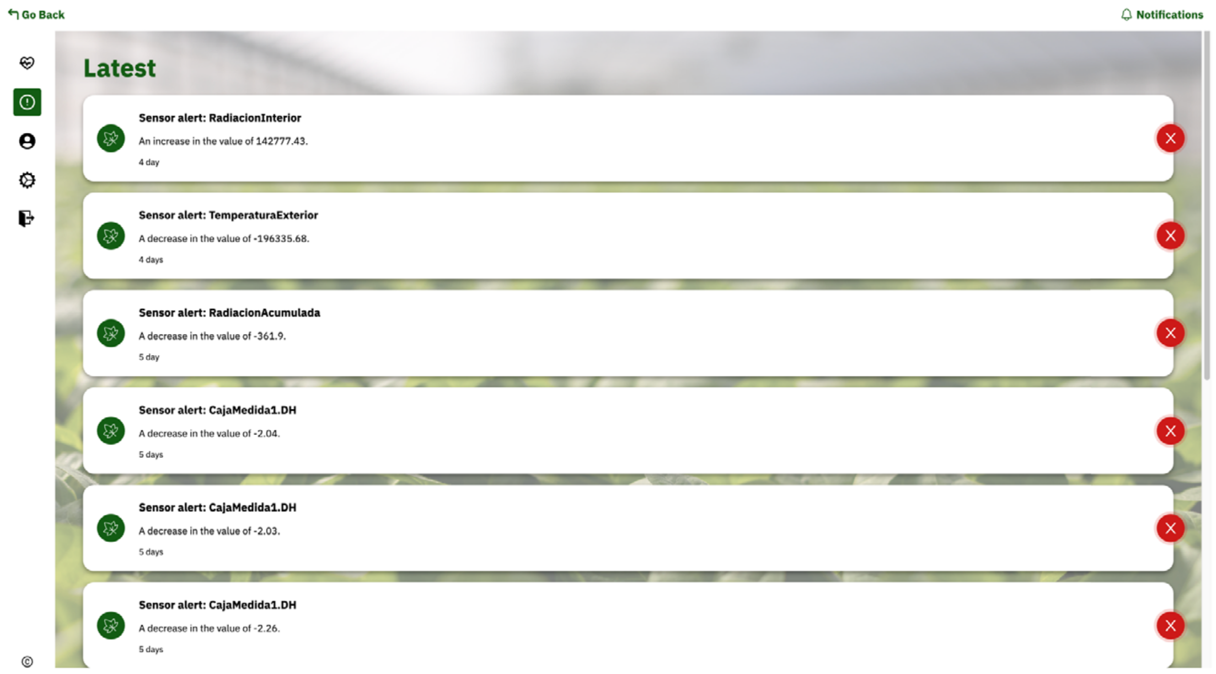
In this section, a comprehensive evaluation and discussion of the results are shown.

4.1. *Running example*

A use case for the application could be that the greenhouse is located in an arid zone, where the summer is very strong and the temperatures are too high. The greenhouse must have the air conditioning system working correctly to keep the crops at a perfect temperature and humidity to avoid damage. Within the application, it has been established that the value of the internal temperature of the greenhouse must be between twenty (20) and twenty-five (25) degrees Celsius. The greenhouse is stable but suddenly the air conditioning system begins to fail, the farmers are doing other tasks and are not in the greenhouse to perceive that the temperature is increasing slightly inside the greenhouse. The sensor that is responsible for obtaining the temperature reading collects a measurement of thirty (30) degrees Celsius that is later sent via the Internet and MQTT to the application server. The server is in charge of receiving this data, saving it, and later comparing the measurement with the range established for that sensor, as it detects that the measurement is greater than the maximum value, it proceeds to send a notification to all connected clients and generates an email that it will reach the default user who will be the greenhouse manager. Upon receiving the mail, this manager will go to the greenhouse and begin to check if the machines are working well. In this way, you can observe the operation of the application to prevent damage to crops due to non-human failures.

4.2. *Real-time alerts*

Notifications are one of the most important features of the system as the user needs to know the status of the greenhouse in real time. This means that any changes that happen must be informed to the user. The notifications are shown as a list in a page as a historic of all the alerts that have happened, as shown in Fig. 6. In this section called recent, a list with the history of all the notifications that have been generated by the application is shown. Here all the alarms are shown, both for increase and decrease of measurements such as alert in the CajaMedida1.DH sensor



Latest

Sensor alert: RadiacionInterior
An increase in the value of 142777.43.
4 day

Sensor alert: TemperaturaExterior
A decrease in the value of -196335.68.
4 days

Sensor alert: RadiacionAcumulada
A decrease in the value of -361.9.
5 day

Sensor alert: CajaMedida1.DH
A decrease in the value of -2.04.
5 days

Sensor alert: CajaMedida1.DH
A decrease in the value of -2.03.
5 days

Sensor alert: CajaMedida1.DH
A decrease in the value of -2.26.
5 days

Fig. 6. Notification list.

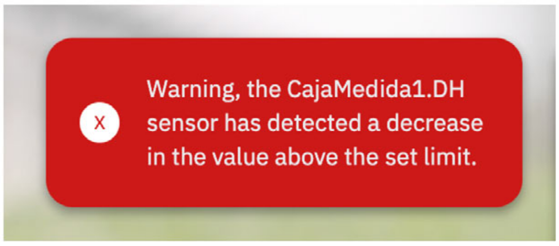
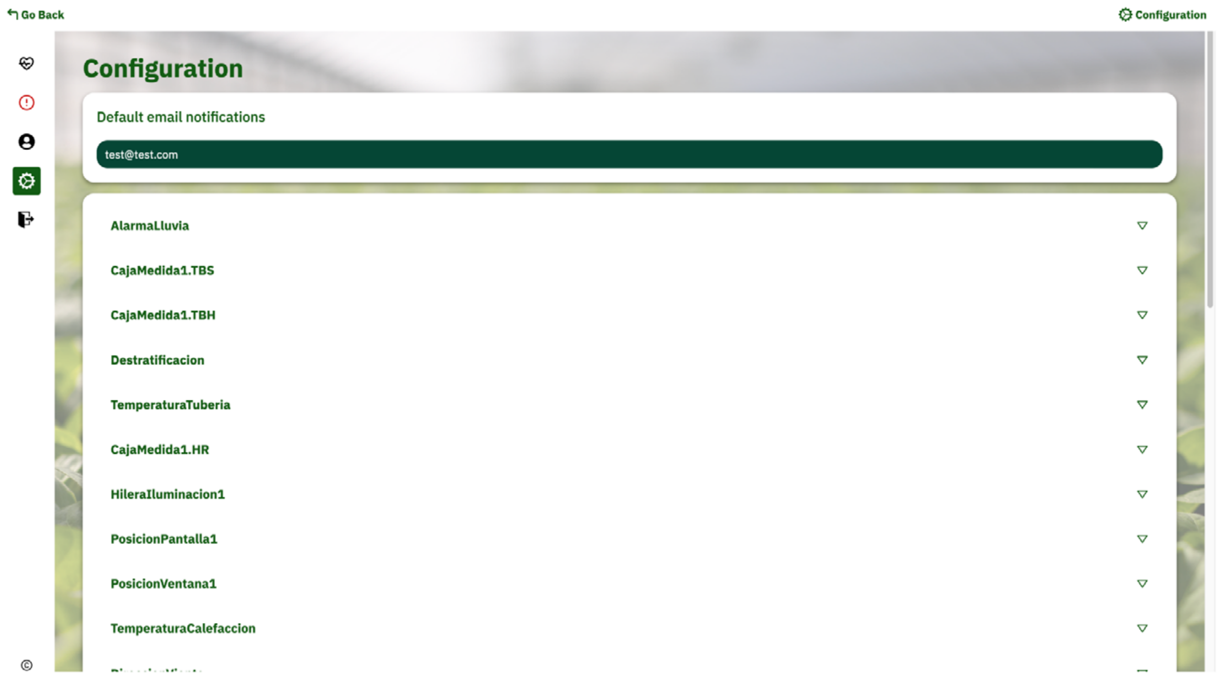

Warning, the CajaMedida1.DH sensor has detected a decrease in the value above the set limit.

Fig. 7. Overheating notification.

where a decrease in the measurement was detected. The date on which the alarm occurred is also displayed. On the other hand, the user is offered the ability to delete the notification from the history in case he considers the alert resolved.

Notification alerts are displayed as a floating modal dialog box at the top of the page and their color depends on whether the measurement that generated the alert is higher or lower than the range set in the settings, as seen in Fig. 7 which displays the text "Warning, the CajaMedida1.DH sensor has detected a value increase over the established limit" which informs the user that the sensor has detected that a measure has increased in value and the greenhouse manager must address this alert. Notifications are red when the value exceeds the upper limit and green when the value falls below the lower limit. Notifications are shown across all the web application no matter in which web page the user is working on. On the other hand, the application does not establish a maximum or minimum resolution time for the alerts. This time will be considered depending on the greenhouse, its location and the alert since not all of them will have the same rank of importance. We leave it in the hands of the farmers as they know best the consequences if any measure is out of range.

The system allows login through Google Firebase, which makes it possible to manage users with Google email or accounts. In addition, a basic password registration and recovery system is included.



Configuration

Default email notifications

test@test.com

AlarmaLluvia

CajaMedida1.TBS

CajaMedida1.TBH

Destratificacion

TemperaturaTuberia

CajaMedida1.HR

HileraIluminacion1

PosicionPantalla1

PosicionVentana1

TemperaturaCalefaccion

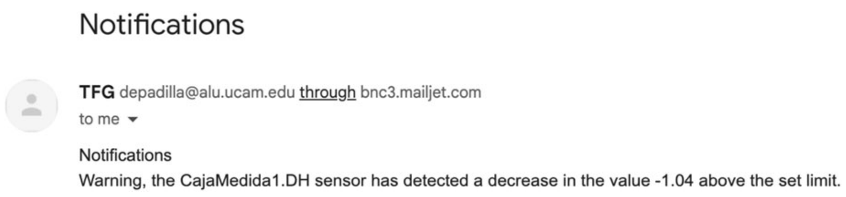
Fig. 8. Configuration page.

Configuration

Default email notifications

test@test.com

Fig. 9. Email configuration section.



Notifications

TFG depadilla@alu.ucam.edu through bnc3.mailjet.com
to me

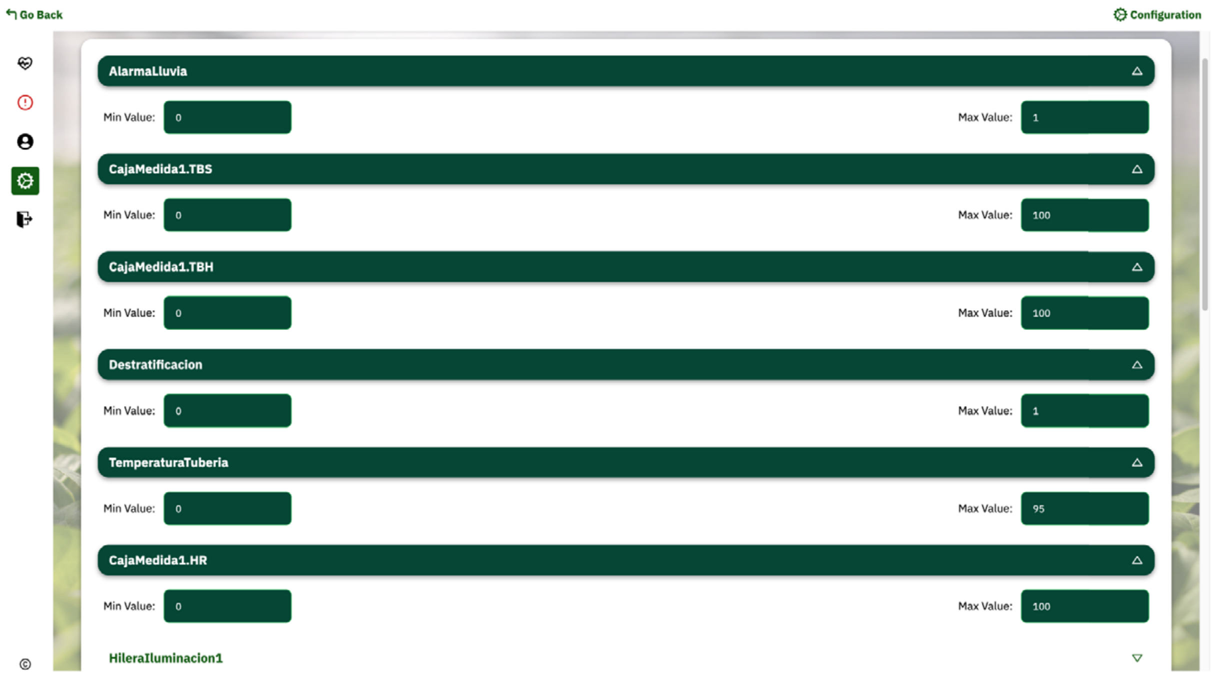
Notifications
Warning, the CajaMedida1.DH sensor has detected a decrease in the value -1.04 above the set limit.

Fig. 10. Email with the notification alert.

The configuration section shown in Fig. 8 is only shown to authenticated users in the web application. A list of all the sensors available to configure is displayed.

Within this section the user will have control over the sensor limits and can also configure the email to which the notifications are to be sent, as shown in the Fig. 9. It is important to note that the notifications will be both for the user who is logged in and for the user defined in the email field “Default email notifications”.

The user will receive within the email all the notifications that have been generated, thus avoiding an overload of emails for the users. The email message shown in Fig. 10, “Notifications: Warning, the CajaMedida1.DH sensor has detected a decrease of the value -1.04 over the established limit.” is a warning about the decrease of the value above the previously established limit. It has a brief summary of which sensor has received the alert, whether it has been an increase or decrease of the set value and the value itself that has caused the alert to be triggered.



AlarmaLluvia
Min Value: 0 Max Value: 1

CajaMedida1.TBS
Min Value: 0 Max Value: 100

CajaMedida1.TBH
Min Value: 0 Max Value: 100

Destratificacion
Min Value: 0 Max Value: 1

TemperaturaTuberia
Min Value: 0 Max Value: 95

CajaMedida1.HR
Min Value: 0 Max Value: 100

HileraIluminacion1

Fig. 11. Sensors configurations sections.

Another relevant feature of the system is the configuration for the generation of these notifications. The application allows the ability to edit an optimal range of measurements for each of the sensors. There are only two data, minimum value and maximum value for each sensor as shown in Fig. 11. This enables the generation of notifications because when new measurements are received, the server starts to check if the measurement is out of the established ranges and generates the notification for each sensor. Then, these notifications are grouped and sent by mail to the user.

This functionality requires direct communication between the client (web page) and the server (Node JS). The server is constantly reading the data from the servers and comparing if each measurement is within a range to know if the measurement is above or below a range. When a measurement presents an anomaly, the server sends a communication to all the clients connected to the server so a notification is shown inside the web application, and also an email is sent to a predefined user, allowing real-time communication of changes. All this is achieved thanks to the use of sockets that allows the server to send a communication to all clients that are connected at that moment as seen in Fig. 12. Even if there is no user connected at that moment, the server will continue to check the measurements and notify the user by email.

4.3. Forecast data alerts

To display the prediction data, we have decided to use a line graph. In the first instance we want to compare the percentage of correctness of the predictions. As mentioned before, the predictions can be grouped by hours, days, weeks and months. An example of this comparisson can be seen in Fig. 13.

To begin with, we have the data grouped by hours. Table 2 show the results obtained by the ARIMA model, where we can see that the difference is almost minimal and error rate doesn't exceed the 0.02%.

Then we have the data grouped by days where we can see that the percentage of error increases a little but the predictions are still almost entirely correct.

In Fig. 14 all real data and predictions can be seen in a line plot. On the other hand, all the results obtained are displayed in Table 3.

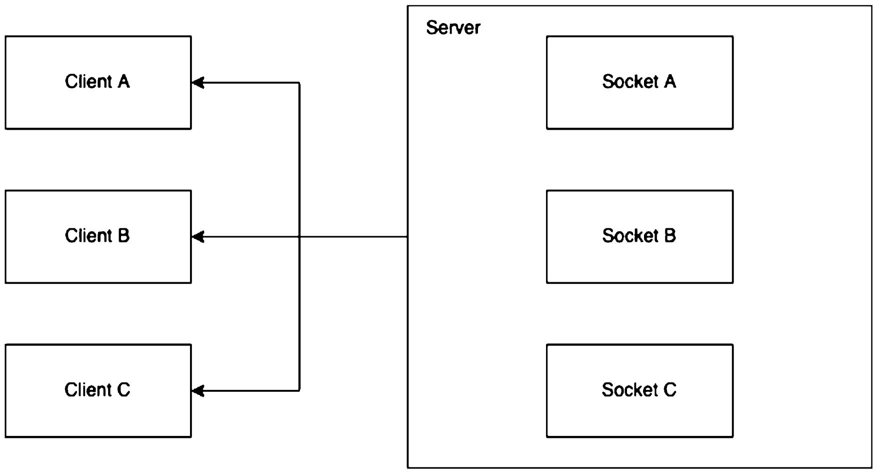


Fig. 12. TCP-based socket communication between the server and different clients.

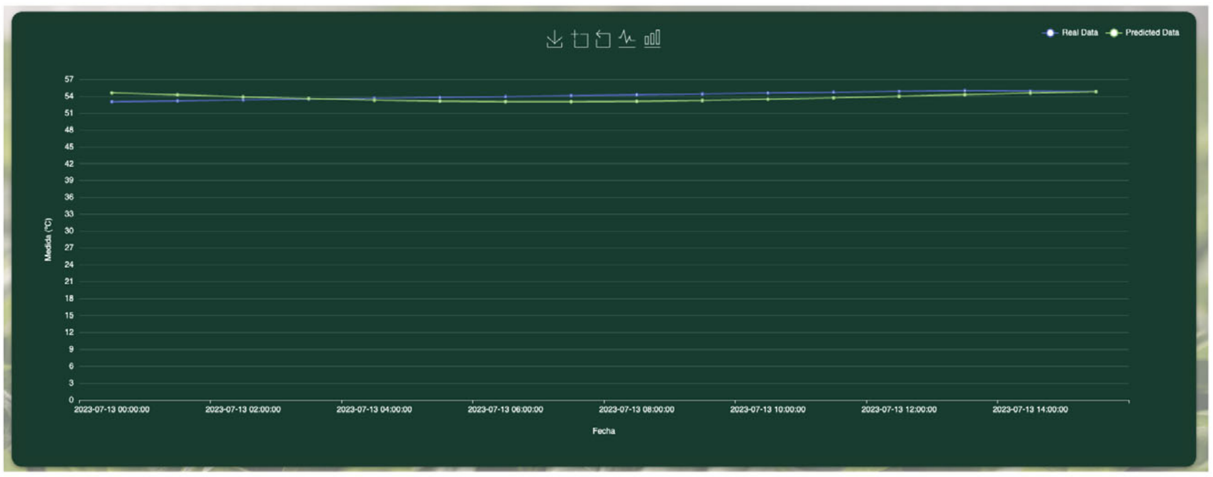


Fig. 13. Real data vs predicted data (hours).

Table 2
Error rate by hours

Date	Real data	Predicted data	Error rate
2023-07-13 00:00	53	54,57	0.02962
2023-07-13 01:00	53,03	54,23	0.02257
2023-07-13 02:00	53,31	53,87	0.01052
2023-07-13 03:00	53,46	53,55	0.00168
2023-07-13 04:00	53,61	52,28	0.02486
2023-07-13 05:00	53,65	53,08	0.01063
2023-07-13 06:00	53,79	52,95	0.01562
2023-07-13 07:00	53,95	52,92	0.01911
2023-07-13 08:00	54,10	52,96	0.02108
2023-07-13 09:00	54,25	53,09	0.02137

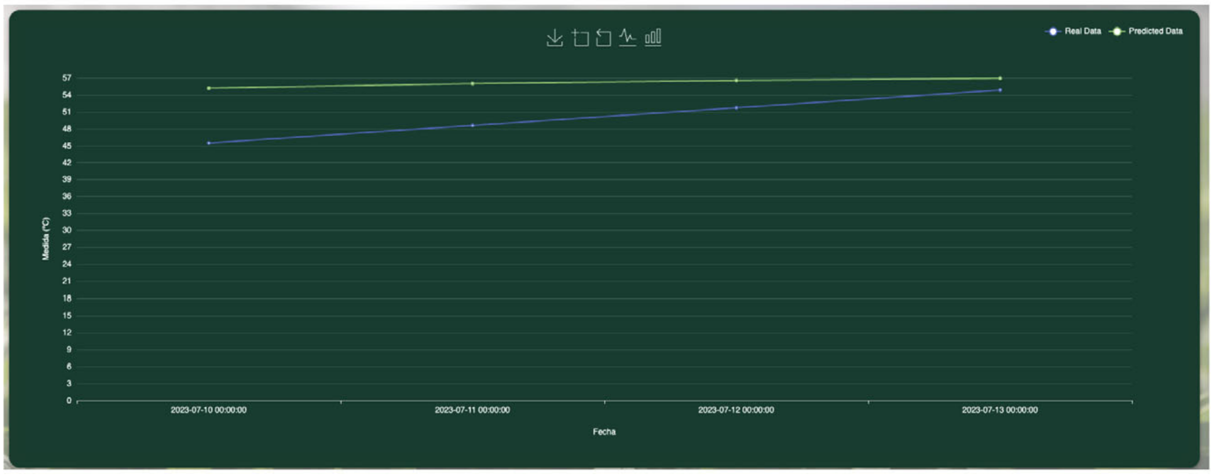


Fig. 14. Real data vs predicted data (days).

Table 3
Error rate by day

Date	Real data	Predicted data	Error rate
2023-07-10	45,43	55,31	0.21729
2023-07-11	48,51	56,11	0.15634
2023-07-12	51,58	56,89	0.10277
2023-07-13	54,66	57,7	0.05553

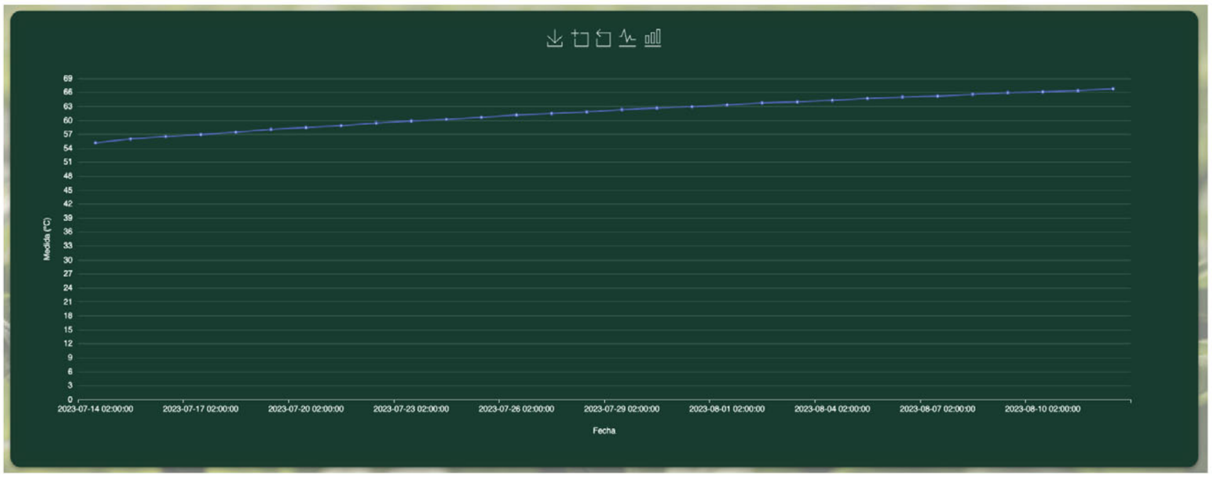


Fig. 15. Predicted data by days.

Finally, it has been decided to represent future forecasts separately. These predictions are based on the selected date filter, i.e. if the selected date is a single day, the predictions for that day are shown within the next 24 hours. If the filter is a range of dates within the same week, data for 31 days of predictions will be shown, if the range of dates is between two weeks, data for 4 weeks of predictions will be shown, and if the range of dates is between two months, predictions for the next 12 months will be shown, as displayed in Fig. 15.

5. Conclusion and future work

Given the growing significance of greenhouses in agriculture for managing variables such as humidity, temperature, and sunlight, this paper proposes developing a web-based smart system named GreenhouseGuard. This system is designed to display data collected from various sensors within the greenhouse and forecast data from those sensors. This feature could assist farmers in preparing for sudden environmental changes in their greenhouses.

GreenhouseGuard, based on a client/server architecture, offers a customizable dashboard accessible via a web browser. It employs well-known IoT technologies to connect sensors deployed in the greenhouse to the server, enabling the storage of collected data for continuous analysis and real-time monitoring of anomalies in the greenhouse's environmental variables. The analysis of this data is conducted using an ARIMA model, capable of making predictions ranging from the next 24 hours up to two months. However, it is important to consider the computational capacity of the web server when executing these predictions, as it may be a limiting factor in some scenarios

As for future work, alternative models could be sought but should not affect the application's performance since it should maintain its ease of use. Also, new tools or sensors capable of communicating with each other can be used to create a unified network so that one sensor can warn another if it detects something unusual. Other Artificial Intelligence models focused on the automation and management of the greenhouse could be included in the system. Finally, we will include rigorous end-user testing of the GreenhouseGuard UI to assess its usability and effectiveness. This will involve empirical studies with greenhouse operators, focusing on UI intuitiveness and responsiveness. Feedback will be systematically analyzed and integrated into subsequent UI refinements, ensuring an optimal balance between functional sophistication and user-centric design.

Declarations

Ethical approval

Not applicable

Conflict of interest

The authors declare they do not have any conflict of interest.

Authors' contributions

Conceptualization, J.M.G., M.C., B.A., A.M. and J.M.C.; methodology, J.M.G., D.P.Q. and M.C.; software, J.M.G. and D.P.Q.; validation, M.C., B.A., A.M. and J.M.C.; formal analysis, J.M.G., M.C., B.A., A.M. and J.M.C.; investigation, J.M.G. and D.P.Q.; writing – original draft preparation, J.M.G., D.P.Q., M.C. and B.A.; writing – review and editing, J.M.G., D.P.Q., A.M. and J.M.C.; visualization, J.M.G. and D.P.Q.; supervision, A.M. and J.M.C.; project administration, A.M. and J.M.C.; funding acquisition, A.M. and J.M.C.

All authors have read and agreed to the published version of the manuscript.

Funding

This work is derived from R&D projects RTC2019-007159-5, as well as the Ramon y Cajal Grant RYC2018-025580-I, funded by MCIN/AEI/10.13039/501100011033, "FSE invest in your future" and "ERDF A way of making Europe". Financial support for this research has been also provided under grant PID2020-112827GB-I00 funded by MCIN/AEI/10.13039/501100011033. It is also partially granted by the "EMERGIA" programme, funded by the Junta de Andalucia through the grant EMC21_004171.

References

[1] S. Aggarwal, Modern web-development using reactjs, *International Journal of Recent Research Aspects* **5**(1) (2018), 133–137.

[2] O. Ajani, M. Usigbe, E. Aboyeji, D. Uyeh, Y. Ha, T. Park and R. Mallipeddi, Greenhouse micro-climate prediction based on fixed sensor placements: A machine learning approach, *Mathematics* **11**(14) (2023), 3052. doi:10.3390/math11143052.

[3] M.A. Akkaş and R. Sokullu, An IoT-based greenhouse monitoring system with Micaz motes, *Procedia computer science* **113** (2017), 603–608. doi:10.1016/j.procs.2017.08.300.

[4] A. Ali and H.S. Hassanein, Wireless sensor network and deep learning for prediction greenhouse environments, in: *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*, 2019, pp. 1–5. doi:10.1109/SmartNets48225.2019.9069766.

[5] I. Bekiaris, A. Leonidis, M. Korozi, C. Stratakis, E. Zidianakis, M. Doxastaki and C. Stephanidis, GRETA: Pervasive and AR interfaces for controlling intelligent greenhouses, in: *2021 17th International Conference on Intelligent Environments (IE)*, 2021, pp. 1–8. doi:10.1109/IE51775.2021.9486584.

[6] C. Bersani, C. Ruggiero, R. Sacile, A. Soussi and E. Zero, Internet of things approaches for monitoring and control of smart greenhouses in industry 4.0, *Energies* **15**(10) (2022), 3834. doi:10.3390/en15103834.

[7] E. Brown, *Web Development with Node and express: Leveraging the JavaScript Stack*, O'Reilly Media, 2019.

[8] T. Cadenhead, *Socket. IO Cookbook*, Packt Publishing Ltd, 2015.

[9] W. Chisholm and M. May, *Universal Design for Web Applications: Web Applications That Reach Everyone*, O'Reilly Media, Inc., 2008.

[10] M. Danita, B. Mathew, N. Shereen, N. Sharon and J.J. Paul, IoT based automated greenhouse monitoring system, in: *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, 2018, pp. 1933–1937. doi:10.1109/ICCONS.2018.8662911.

[11] D. Gollin, S. Parente and R. Rogerson, The role of agriculture in development, *American economic review* **92**(2) (2002), 160–164. doi:10.1257/000282802320189177.

[12] G. Guerrero-Ulloa et al., Internet of Things (IoT)-based indoor plant care system, *Journal of Ambient Intelligence and Smart Environments* **15**(1) (2023), 47–62. doi:10.3233/AIS-220483.

[13] C. Győrödi, R. Győrödi, G. Pecherle and A. Olah, A comparative study: MongoDB vs. MySQL, in: *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, IEEE, 2015, pp. 1–6.

[14] J.C.S. Jose Manuel Ortuño Juan and A. Ramos, Análisis Estadístico de Series Económicas, Grado Estadística Empresarial (2017).

[15] N. Kitpo, Y. Kugai, M. Inoue, T. Yokemura and S. Satomura, Internet of things for greenhouse monitoring system using deep learning and bot notification services, in: *2019 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, 2019, pp. 1–4.

[16] Y. Liu, D. Li, S. Wan, F. Wang, W. Dou, X. Xu, S. Li, R. Ma and L. Qi, A long short-term memory-based model for greenhouse climate prediction, *International Journal of Intelligent Systems* **37**(1) (2022), 135–151. doi:10.1002/int.22620.

[17] Y. Mu, K. Gao and R. Du, Prediction of regional carbon emissions using deep learning and mathematical–statistical model, *Journal of Ambient Intelligence and Smart Environments Pre-press(Pre–press)* (2023), 1–17.

[18] P.V. Nelson et al., *Greenhouse Operation and Management*, 4th edn, Prentice Hall, 1991.

[19] R. Rayhana, G. Xiao and Z. Liu, Internet of things empowered smart greenhouse farming, *IEEE Journal of Radio Frequency Identification* **4**(3) (2020), 195–211. doi:10.1109/JRFID.2020.2984391.

[20] R.H. Shumway, D.S. Stoffer, R.H. Shumway and D.S. Stoffer, ARIMA models, in: *Time Series Analysis and Its Applications: With R Examples*, 2017, pp. 75–163. doi:10.1007/978-3-319-52452-8_3.

[21] D. Soni and A. Makwana, A survey on mqtt: A protocol of Internet of things (iot), in: *International Conference on Telecommunication, Power Analysis and Computing Techniques (ICTPACT-2017)*, Vol. 20, 2017, pp. 173–177.

[22] Z. Tafa, F. Ramadani and B. Cakolli, The design of a ZigBee-based greenhouse monitoring system, in: *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, 2018, pp. 1–4.

[23] S. Tilkov and S. Vinoski, Node.js: Using JavaScript to build high-performance network programs, *IEEE Internet Computing* **14**(6) (2010), 80–83. doi:10.1109/MIC.2010.145.