UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

Design and Implementation of a System of Automation, Monitoring and Control of Crops in Plastic Greenhouses using

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información.

Autor:

Barba Carvajal Marcelo Leonel

Tutor:

PhD. Pineda Arias Israel Gustavo

Urcuquí, diciembre 2020

Urcuquí, 29 de septiembre de 2020

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2020-00032-AD

A los 29 días del mes de septiembre de 2020, a las 14:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dra. GUACHI GUACHI, LORENA DE LOS ANGELES , Ph.D.
Miembro No Tutor	Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.
Tutor	Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.

El(la) señor(ita) estudiante **BARBA CARVAJAL, MARCELO LEONEL**, con cédula de identidad No. **1003954185**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **DESIGN AND IMPLEMENTATION OF A SYSTEM OF AUTOMATION, MONITORING AND CONTROL OF CROPS IN PLASTIC GREENHOUSES USING EMBEDDED DEVICES**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Presidente Tribunal De Defensa	Dra. GUACHI GUACHI, LORENA DE LOS ANGELES , Ph.D.	9,8
Tutor	Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.	10,0
Miembro Tribunal De Defensa	Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.	9,0

Lo que da un promedio de: **9.6 (Nueve punto Seis)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

Certifico que *en cumplimiento del Decreto Ejecutivo 1017 de 16 de marzo de 2020, la defensa de trabajo de titulación (o examen de grado modalidad teórico práctica) se realizó vía virtual, por lo que las firmas de los miembros del Tribunal de Defensa de Grado, constan en forma digital.*

[illegible]

BARBA CARVAJAL, MARCELO LEONEL
Estudiante

Firmado Digitalmente por: LORENA DE LOS ANGELES GUACHI GUACHI
Hora oficial Ecuador: 16/10/2020 14:01

Dra. GUACHI GUACHI, LORENA DE LOS ANGELES , Ph.D.
Presidente Tribunal de Defensa

Firmado electrónicamente por:
ISRAEL
GUSTAVO
PINEDA ARIAS

Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.
Tutor

ERICK EDUARDO
CUENCA PAUTA
Firmado digitalmente por ERICK EDUARDO CUENCA PAUTA
Fecha: 2020.10.19 23:25:45 -05'00'

Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.
Miembro No Tutor

TATIANA
BEATRIZ TORRES
MONTALVAN
Firmado digitalmente por TATIANA BEATRIZ TORRES MONTALVAN
Fecha: 2020.10.16 13:48:49 -05'00'

TORRES MONTALVÁN, TATIANA BEATRIZ
Secretario Ad-hoc

Autoría

Yo, **Marcelo Leonel Barba Carvajal**, con cédula de identidad **1003954185**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Marzo del 2020.

[illegible]

Marcelo Leonel Barba Carvajal
CI: 1003954185

Autorización de publicación

Dedication

"To my family, friends, and professors as their support and patience were very significant in reaching this achievement."

Acknowledgments

First of all, I would like to acknowledge Professor Israel Pineda, Ph. D., for his patience, dedication, and constant support in the development of this work and my training as an engineer and researcher. His writing and implementation recommendations allowed me to improve my skills and apply them to this and other projects I made. Without his help and perseverance, this project would hardly have developed as well as it did.

I also want to acknowledge Francisco Hidrobo, Ph.D., and Cristhian Iza, Ph, D., who helped shape this project in its early stages. Their recommendations and guidance helped me to define the basis of this project and develop it better. I greatly appreciate their invaluable collaboration in my work.

It is fair to mention the ideas, suggestions, and comments that helped to improve this project. I have to highlight the help of María Daniela Palma and Robinson Paspuel, which was meaningful for the organization and redaction of this degree project.

Finally, I want to thank everyone who allowed me to be part of the Yachay project, who made it possible for me to reach this point in my life and being able to access an excellent education at one of the best universities.

Abstract

The Internet of Things (IoT) has had a very significant development in the last years, mainly because of its wide variety of applications. From connecting a toaster to connecting a series of industrial machines to the Internet translates into greater efficiency and resource savings. The IoT impact has been so strong that even traditional interconnection models, such as OSI or TCP/IP, have been modified to adapt to this new technological revolution.

Combining web development techniques with IoT makes it possible to create tools to control devices instantly. One of the most potential fields for applying these integrated tools is precision agriculture. Thus, this work proposes a system that automates and evaluates the environmental variables of greenhouse crops. This IoT system can control and monitor crops through electronic devices and analyze the data acquired by digital and analogical sensors. This system uses a series of devices like sensors, actuators, and communication nodes to measure environmental variables inside a greenhouse, such as environmental temperature, environmental and soil humidity, irrigation, ventilation, and water level. The end-user can make decisions based on the acquired measurements displayed in a graphical interface as a time series.

The measurements and user interactions travel through an intermediary application (broker) using ESP8266 NodeMCU modules connected to the Internet by WiFi. The system employs an EMQX broker using the MQTT protocol for data transmission. Given the nature of MQTT implementation and performance, it consumes less bandwidth than the HTTP protocol, usually used for data transfer. The system also consists of a web application that reads, stores, and displays the data in real-time in a graphical user interface (GUI). One of the most critical functions of this application is the broker connection that manages the flow of both read and write signals, i.e., what the user sees and controls.

In its final stage, the project uses artificial intelligence tools to obtain relevant predictions of environmental variables based on data collected over time. A recurrent neural network known as Long Short-Term Memory (LSTM) is in charge of the forecast generation by analyzing the time series generated by the data readings.

Keywords: Internet of Things, Greenhouse, Precision Agriculture, IoT Platform, MQTT, Data Analysis.

Resumen

El Internet de las Cosas (IoT) ha tenido un desarrollo muy significativo en los últimos años, principalmente debido a su amplia variedad de aplicaciones. Desde conectar una tostadora hasta una serie de máquinas industriales a Internet se traduce en una mayor eficiencia y ahorro de recursos. El impacto del IoT ha sido tan fuerte que incluso los modelos de interconexión tradicionales, como OSI o TCP/IP, se han modificado para adaptarse a esta nueva revolución tecnológica.

Combinar técnicas de desarrollo web con IoT permite crear herramientas para controlar los dispositivos de forma instantánea. Uno de los campos con más potencial para la aplicación de estas herramientas combinadas es la agricultura de precisión. Por ello, este trabajo propone un sistema que automatiza y evalúa las variables ambientales de los cultivos en invernaderos. Este sistema de IoT puede controlar y monitorear los cultivos a través de dispositivos electrónicos y analizar los datos adquiridos por los sensores digitales y analógicos. El sistema utiliza una serie de dispositivos como sensores, actuadores y nodos de comunicación para medir las variables ambientales dentro de un invernadero, como la temperatura ambiental, la humedad ambiental y del suelo, la irrigación, la ventilación y el nivel de agua. El usuario final puede tomar decisiones basadas en las medidas adquiridas que se muestran en una interfaz gráfica a manera de serie temporal.

Las mediciones y las interacciones del usuario viajan a través de una aplicación intermediaria (broker) usando módulos ESP8266 NodeMCU conectados a Internet a través de WiFi. El sistema emplea un broker EMQX que utiliza el protocolo MQTT para la transmisión de datos. Debido a la naturaleza de la implementación y funcionamiento de MQTT, este protocolo consume menos ancho de banda que el protocolo HTTP, normalmente usado para la transferencia de datos. El sistema consiste en una aplicación web que lee, almacena y muestra los datos en tiempo real en una interfaz gráfica de usuario. Una de las funciones más críticas de esta aplicación es la conexión con el broker para gestionar el flujo de señales tanto de lectura como de escritura, es decir, lo que el usuario ve y controla. En su etapa final, el sistema utiliza herramientas de inteligencia artificial para obtener predicciones relevantes de las variables ambientales, basadas en datos recogidos a lo largo del tiempo. Una red neuronal recurrente conocida como memoria a corto plazo (LSTM, por sus siglas en inglés) es la encargada del análisis de series temporales generadas por las lecturas de datos y la generación de las predicciones.

Palabras Clave: Internet de las Cosas, Invernadero, Agricultura de Precisión, Plataforma de IoT, MQTT, Análisis de Datos.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Agriculture is one of the most critical activities of humanity. Farmers worldwide are forced to produce more and better quality food due to the increase in population. This is a highly complex problem for which we need modern and innovative solutions. For example, we can use current technologies such as electronics and computers to aid people working in this field. The Internet of Things (IoT), is a giant Internet-based network that connects physical and virtual *things* with standard and interoperable communication protocols [4]. Modern IoT-based systems [1][5] include not only the electronics and network components but also web applications and machine learning models for data analysis. This sophisticated combination of techniques provides users with easy-to-use tools that enable farmers to carry out cultivation activities more efficiently.

IoT technology allows us to control the environmental variables of the crops in a much more efficient way than a person could do. The sensors work uninterruptedly and give us real-time information about the changes that occur in the crops. Based on this information, we can take action through the use of other electronic devices known as actuators. This information can be stored and sent through the Internet to visualize it in a graphical user interface (GUI). It would not make sense for our data to be stored in a database without any meaningful benefit. Thus, giving adequate treatment to the data is very important to obtain new and useful insights. Modern machine learning techniques extract these insights [6][7]. Neural networks are potent machine learning algorithms capable of processing time series [8][7] such as those obtained from environmental sensors.

This project proposes a system of IoT consisting of several components organized in a four-layer architecture, as suggested by Varutti et al. [1]. First, the device layer describes the arrangement of sensors, actuators, and data transmission nodes (NODEMCU). The second level corresponds to the network layer, here it is defined how the data will be sent (protocols) and where they will be sent (routes) at their lowest levels. The application services platform layer is the third layer of the architecture. Here are the data transmission protocols, in our case MQTT [9] and the application in charge of handling it, in our case, the EMQX broker. Finally, the closest layer to the end-user is the application layer. Here we build the web application using several components [10] such as front-end, back-end, database, and a server that form a structure usually called the stack. This layer usually can be developed independently from the others.

This project uses a virtual private server (VPS) since it provides certain advantages over other techniques [11], such as storage capacity, processing capacity, statistical analysis, and security. The system we designed also uses long short-term memory (LSTM) [12] to analyze the data. This machine learning technique efficiently makes predictions based on time series with discrete values.

1.2 Problem Statement

Farmers face several problems when growing crops. The main problem is usually irrigation, and more specifically, the achievement of uniform irrigation according to the growing substrate needs and the state of the crop. The appearance of pests and poor nutrition are also decisive factors in obtaining a good harvest. A yield with pests implies an extra expense in pesticides and a direct affectation to the final product. Growing in greenhouses is one of the most effective techniques to solve problems, such as those mentioned above. It facilitates the control of natural environmental variables such as temperature and humidity, and the management of artificial variables such as fertilizers and pesticides within the structure.

One of the most reliable motivations for using greenhouses is to grow without depending on external climatic factors such as ambient light, seasons, excessive rainfall, or drought. The improvements that a greenhouse brings translates into the possibility of having several harvests in a year, healthy crops, and better quality products. Without a doubt, this reflects in more benefits [13].

1.3 Objectives

1.3.1 General Objective

Design and implement an IoT system employing modern web development techniques, Internet of things, and data analysis to facilitate the monitoring and control of greenhouse crops.

1.3.2 Specific Objectives

- Build a greenhouse model and equip it with various sensors, actuators, and communication nodes to measure environmental variables.
- Develop an IoT platform to receive data sent by the communication nodes, store and display them on an intuitive user interface.

- Make predictions of environmental variables, based on the data collected, using artificial intelligence techniques.

Chapter 2

Theoretical Framework

This chapter presents the necessary concepts to understand this work, giving a detailed explanation of each component of the Internet of Things stack and the web development tools for creating a web application that allows the interaction of IoT devices with the end-user. It also explains the data analysis techniques that enable us to obtain predictions of the environmental variables over time employing artificial intelligence tools.

2.1 IoT Protocol Stack

The following IoT protocol stack is based on the model proposed by Varutti et al. [1], aiming to simplify and facilitate the development of IoT systems. Figure 2.1 shows the four layers of the IoT stack and compares them with the ISO-OSI and TCP/IP models.

According to this structure, the application layer can be developed independently of the underlying layers. The Application Services platform layer works as a middleware to distribute the required applications and protocols throughout the system while ensuring access to the device layer. The network layer continues to function in the same way as in traditional models. In the device layer are all the electronic devices that we want to use as sensors and actuators. In this layer, we also find the electronic devices in charge of receiving the data, sending it to the server, and receiving the user control signals.

2.1.1 Application Layer

In this first layer, the web application is developed, for which we currently have many programming languages and development styles. To create a web application, we need a server, a program that runs in the browser (Front-end), a program that runs in the server (Back-end), and a database to store data. Over time specific arrangements of web development tools have gained popularity among programmers. These arrangments are given the name of *The Stack*. Among the most used stacks [14], we find LAMP (Linux, Apache, MySQL, and PHP) and MEAN (MongoDB, Express, Angular and Node.js) stacks.

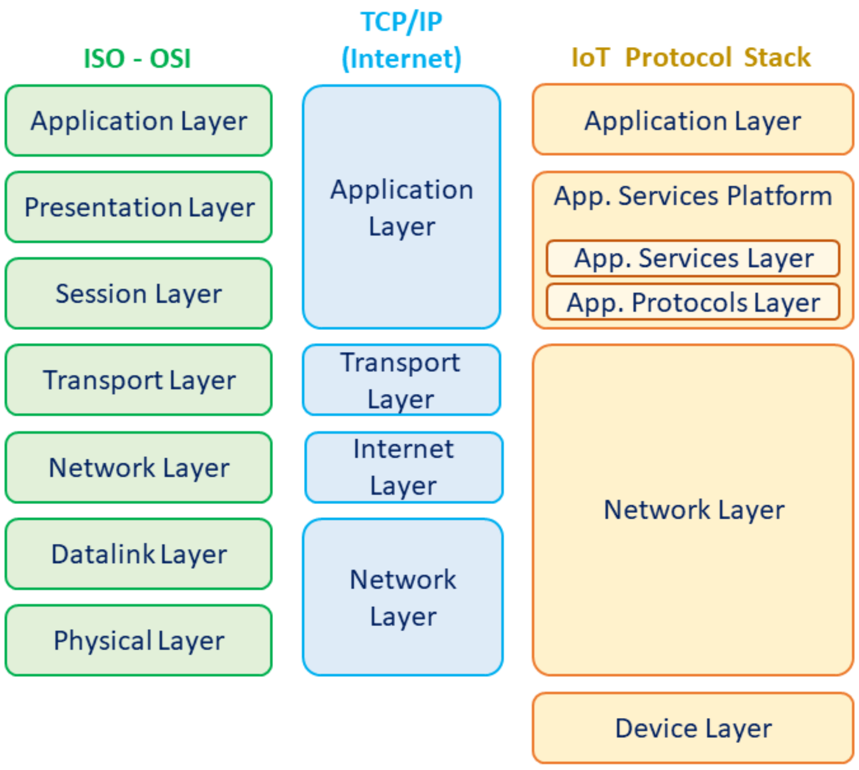


Figure 2.1: IoT Protocol Stack compared to the OSI and TCP/IP models.

MEAN Stack

The MEAN [15] stack is a set of modern web tools intended to be robust and flexible. It consists of four elements firmly focused on Javascript. The first element is MongoDB, a NoSQL database system, then Express, a web server framework for Node, AngularJS is a client-side script, and Node is a server platform. Each component of the MEAN stack is written in JavaScript, even the database stores data in JavaScript Object Notation (JSON) format. Using JavaScript as the primary programming language on both the client and server sides means that the MEAN stack is a robust set of tools. The following is a brief description of each of the components of this Stack.

- MongoDB

  MongoDB [16] is a NoSQL document storage model. It stores in the JSON format database, rather than in rows and columns as in the relational database. Given the scalability of MongoDB, it is used on high traffic websites such as Facebook and Twitter.

- Express

  Express [15] is a node-based framework. It organizes web applications on the server-side. Express uses the Node HTTP module and connects middleware components, allowing code reutilization within applications and even imparting it to others as modules.

- Angular

AngularJS [17] is a client-side JavaScript MVC system maintained by Google. It provides functionality to manage client information in the browser, to control information on the client-side, and to manage how components are displayed in the browser view.

- Node

  Node is a development framework built on Google's JavaScript V8 engine to build fast and scalable network applications. Node is a server-side scripting language. However, it can even be a web server. This framework even helps reduce development time and resources.

LAMP Stack

LAMP [18] is a web development stack with open-source software with Linux as the operating system, Apache as the webserver, MySQL as a database system, and PHP/Perl as the background scripting language. Each LAMP element allows smooth integration with each other, making LAMP one of the most widely used platforms for web application development. The following is a brief description of each of the components of this Stack.

- Linux Operating System

  Linux is the operating system that runs the computer's applications to perform the assigned task. It is known for its speed, low hardware requirements, security, and remote administration. It is the dominant operating system in the server world.

- Apache Webserver

  Apache is an open-source web server developed by the Apache Software Foundation (ASF). Apache works well with the Linux operating system and allows virtual hosting to run multiple websites on a single server.

- MySQL Database System

  MySQL is a powerful and robust relational database management system (RDBM) that allows data to be stored and retrieved using a scripting language such as PHP. Different types of data can be stored, such as the Boolean operator, text, integers, images, binary digits, ENUM or enumeration, large binary objects (BLOBs), and date.

- PHP Scripting Language

  PHP [19] is a server-side scripting language or programming language for building websites, which means it runs on a web server. PHP can be embedded in HTML and dynamically enhance an application that helps build very easy-to-use websites embedded with HTML and CSS.

MEAN vs. LAMP

The MEAN stack is becoming increasingly popular due to the characteristics of its components. However, the MEAN stack usually is not used for CPU-intensive applications.

In this case, the LAMP stack is the preferred choice. For simple single-page applications (SPA), which require a lot of user interaction, visualization and need much scalability, the MEAN stack is the most suitable.

2.1.2 Application Services Platform

Application Services Layer

The Application Service Layer contains the software in charge of handling the protocols for the traffic of the data coming from the sensors. These applications are the programs on which the protocols explained in the Application Protocol layer work. Among the main types of application services, we can find the following.

- Distribution Service (DDS)

  DDS is a decentralized publication and subscription communication protocol for Machine to Machine (M2M) applications. It is an application that uses multicast to distribute messages instead of a centralized intermediary. It also focuses on quality of service (QoS) and content reliability by introducing various parameters such as availability, delivery, timeliness, and data security. The protocol provides built-in security mechanisms such as data confidentiality and integrity, authentication, and authorization of publishers (i.e., authors of the data) and subscribers (i.e., readers of the data).Figure 2.2 shows different network protocols including DDS.

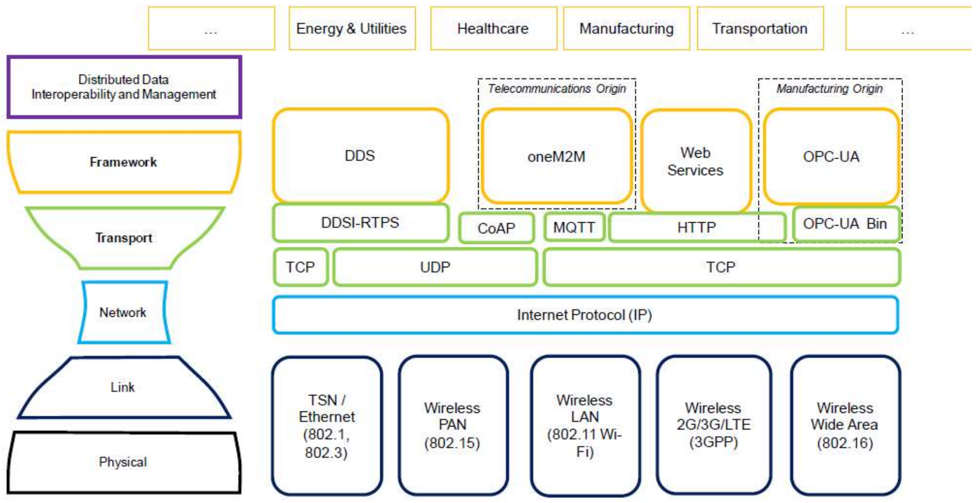


Figure 2.2: DDS and other Protocols in the Network Layer. Image from [2].

- OPC Unified Architecture (OPC-UA)

  OPC-UA is an M2M communication protocol developed by the Object Linking and Embedding for Process Control Foundation for industrial automation combining several protocols to create a universal data model and thus facilitate interoperability. It allows the exchange of information models of any complexity and supports a wide range of systems, from programmable logic controllers (PLCs) for production to enterprise servers.

- oneM2M for the Industry Domain

  oneM2M is a global standards project covering various requirements, architecture, API specifications, security solutions, and M2M interoperability. It was formed in 2012 with the support of eight international standards development organizations among which are TSDSI (India), ARIB (Japan), ATIS (USA), CCSA (China), TTA (Korea), ETSI (Europe), TIA (USA) and TTC (Japan). oneM2M seeks to develop technical specifications that address the need for a standard M2M service layer.

Application Protocols Layer

In this layer, we find the protocols in charge of handling the data coming from the devices. The IoT contains a wide variety of protocols with different approaches, including energy and bandwidth saving. The main protocols for message transport are the following.

- Message Queue Telemetry Transport (MQTT)

  MQTT [20][21] is an M2M protocol for message transport, frequently used in IoT. It is currently the most popular publish-subscribe solution. This protocol is a centralized implementation that uses a dedicated broker to communicate a publisher with a subscriber by filtering the topics, as shown in Figure 2.3.

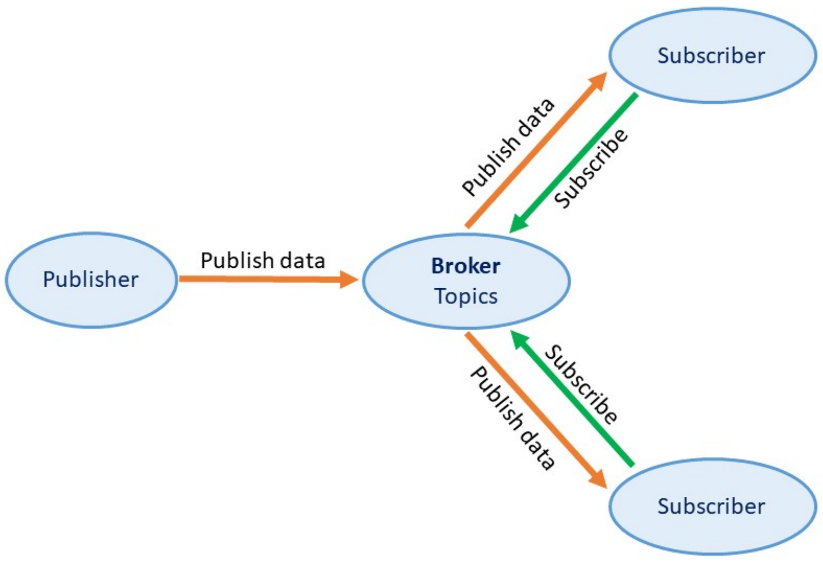


Figure 2.3: MQTT Communication Method

  This protocol also uses the TCP and TLS/SSL connection-oriented transport protocol for the reliability of its messages. MQTT has three Quality of Service (QoS) levels listed in the table 2.1 to ensure the successful transmission of messages.MQTT focuses on the complex intermediary logic, keeping publishers and subscribers as lightweight and straightforward applications that can be run on low-resource devices.

- Constrained Application Protocol (CoAP)

QoS Level	Description
0 (at most once)	With this type of delivery, only one delivery attempt is made. If the delivery was not successful, it is not reattempted and you do not have an ACK(acknowledgement) message to confirm receipt.
1 (at least once)	The message is delivered at least once to the receiver. This type of sending if you have an ACK verification message. Multiple message delivery is possible.
2 (exactly once)	The message is delivered just once using a four-part handshake. This type of delivery is the slowest, but also the safest. It is very useful if duplicate messages can interrupt the data flow.

Table 2.1: MQTT QoS Levels

CoAP is handled by the RESTful concept for a decentralized architecture. The REST protocols apply the client-server pattern where the communication path is given if the clients establish connections with the servers, the latter being unable to initiate the communication. To maintain a communication history, each client can store data in the cache.

- Extensible Messaging and Presence Protocol (XMPP)

  The XMPP [22] is a standard for instant messaging, audio, and video in real-time. It is a partially decentralized protocol that supports client-server and publish-subscribe communication models. This protocol uses the Extensible Markup Language (XML) as a plain text message format.

- Advanced Message Queuing Protocol (AMQP)

  AMQP is a centralized publication-subscription communication protocol with intermediary instances. It ensures the exchange of messages using a QoS system similar to MQTT. It uses TCP to perform these delivery guarantees. This protocol has an exchange module and message queues.

2.1.3 Network Layer

This part presents some of the network technologies currently available for communication, emphasizing the data link, routing, and encapsulation layers. Most of the protocols presented in this section are based on the IP, and UPD/TCP protocols, as figure 2.2 shows. Below are some standard and non-standard routing and packaging protocols for IoT applications. The routing protocols provide mechanisms for handling the transfer of packets from source to destination.

- IEEE 802.15.4

Application Layer		
App. Services Layer		OPC-UA, oneM2M, DDS
App. Protocols Layer		MQTT, SMQTT, DDS, CoAP, XMPP,
Network	**Transport**	TCP,UDP, DCCP
	Internet	IP, ICMP, ECN
	Encapsulation	6LowPAN, 6TiSCH, 6Lo
	Routing	RPL, CORPL, CARP
	Datalink and Physical	RPL, CORPL, CARP
Device Layer		

Table 2.2: IoT Protocol Stack and Commonly Available Protocols for Each Layer. Table from [1]

IEEE 802.15.4 is a low data rate wireless connectivity solution that focuses on very low complexity and long battery life in several months to several years. The potential applications of the solution range from interactive toys to home automation. This short-range radio technology targets very low-cost battery-powered devices that can intercom and send low-bandwidth data to a centralized tool.

- IEEE 802.11ah (Low Energy WiFi)

  IEEE 802.11ah is a lightweight version of the IEEE 802.11 access standard that defines a WLAN system operating in less than 1 GHz. Due to the low-frequency spectrum favorable propagation characteristics, this protocol can provide a better transmission range than conventional WiFi. Besides, to meet the requirements for IoT, it was designed with reduced overhead compared to the original WiFi. It also benefits from lower power consumption, which allows for large device networks, such as distributed sensor networks.

- Bluetooth Low Energy (BLE)

  Low Power Bluetooth (BLE) is a short-range wireless communication protocol widely used for WPANs. It aims to reduce energy consumption and costs while maintaining communication ranges similar to those of the original Bluetooth. Compared to its homonym, power consumption can be up to ten times lower in exchange for significantly increased latency, with up to 15 times more than in Bluetooth. The protocol follows a client/server architecture where the client is a device that requests pieces of information, and a server is a device that provides information. A device can be either a client or a server at certain times.

- Zigbee

  ZigBee is an IEEE 802.15.4 based specification for high-level communication protocols used to create WPANs with small, low power radios as automation for small environments. Its low power consumption limits the transmission distance to a line of sight of approximately 100 m, i.e., to increase distances, intermediary devices are necessary.

- 6LoWPAN

  6LoWPAN means IPv6 over low power WPAN. It is probably the most widely used standard for packet encapsulation, as it can efficiently enclose long IPv6 headers in small 128-byte packets. The specification supports different address lengths, low bandwidth, different network topologies, power consumption, low cost, scalable networks, mobility, unreliability, and long timeouts.

2.1.4 Device Layer

This part explains the components of the device layer, also called *smart things*. The device layer of an IoT architecture is the base on which all other layers are built. An IoT system is generally deployed when a network of interconnected and independent sensors, actuators, or subsystems is needed to automate an environment.

- Sensors

  Sensors are devices capable of detecting changes in the environment and measuring physical phenomena, such as temperature or humidity. They can also transmit the acquired data as an electrical signal to another device that can interpret it. The sensors can capture the phenomena through a series of reactions, such as the activation of conductors and chemical elements that react to heat or water. Among the most common sensors, we can find pressure sensors, light sensors, humidity sensors, accelerometers, gyroscopes, motion sensors, and more.

- Actuators

  An actuator is a device that takes an electrical input, which means off or on, and transforms it into a physical phenomenon. Sensors generate data from the interaction with the environment. These measurements allow the user to make decisions by activating the actuators. Some of the most known actuators are the servo motor, DC motor, stepper motor, relay, and solenoid.

- System on a Chip

  A System on a Chip (SoC) is a device that includes all the components necessary to be a complete system, one of the most popular SoCs is the Raspberry Pi. SoC components can consist of analog-to-digital converters, logic control circuits, memory modules, digital, analog or mixed-signal functions, and any other parts. Its operation does not depend on external components to operate. Due to their size and low power consumption, SoCs have become very popular in integrated systems. In the IoT ecosystem, the primary responsibility of an SoC is the communication between the sensors and the actuators. SoCs usually have different protocols by default to send and receive data within or between systems.

- Smart Things

  A smart thing is a real-world object with the ability to interact with its environment. Smart things must have connectivity to some network to send data or receive signals to communicate with other devices. Besides, they allow objects to have a virtual presence represented in digital systems. The data collected from intelligent things can be analyzed to help decision making, generate predictions, enable operational efficiency, and improve product performance. Among the essential smart devices of the last years are intelligent cars, intelligent meters, smartwatches, interconnected electric pumps and valves, public lighting, and smartphones.

2.2 Data Management

Data management in an IoT system is one of the most complicated tasks due to the fast sampling rates required for some activities. The diversity of the data and the volume needed to store it are also factors that can make the task difficult. If we add that this data is usually controlled or monitored in real-time, the complexity increases even more. These characteristics give rise to the non-trivial task of deciding which storage systems are the most suitable for data management. The following are some of the considerations to be taken into account to handle IoT platform data.

2.2.1 Sensor Data

The sensors generate data on their interaction with physical phenomena. These data have some characteristics that allow us to organize them and give them some treatment if necessary. We can distinguish two main types of data depending on their nature. In the first group, we can find the data generated by the sensors that perform periodic measurements. These data usually need a large space for storage, depending on the frequency of the measurements. In the second group, we find data generated by sensors that react to some external stimulus and take samples only in certain circumstances. These sensors do not usually require large amounts of storage, but they do require a reliable operation.

The solutions presented to manage the storage are focused on relational and non-relational systems. Relational storage systems rely on structured query language (SQL) to define storage structures. This system is characterized by being suitable for hosting large amounts of data and being robust but can be restrictive since they strictly adhere to the predefined scheme. On the other hand, non-relational storage systems have gained popularity in recent years because they can quickly adapt when the schema changes or even if a schema is not predefined. This is often the case with IoT systems where sensor networks can be deployed that may be growing or continuously changed.

2.2.2 Non-relational Data Storage

Non-relational data storage systems provide dynamic schemes for unstructured data, which has certain advantages for some data types. Because data is stored in many ways, it is easier to store unstructured data, and generally, the information is more scalable. Since IoT devices can generate large amounts of data, scalable schemes are more suitable for industrial environments. Non-relational data storage benefits for IoT are shown in Table 2.3.

Feature	Description
Flexible Data model	Flexible data models, such as documents, charts, key-value pairs, make it easy to combine data from any structure because they allow dynamic modification of the schema without affecting performance.
Scalability Performance	NoSQL allows you to partition your datasets, allowing quick and easy scalability on hardware deployed on servers or in the cloud.
Always-On Global Deployments	NoSQL storage systems are designed to work across many nodes. This means that they include replication to synchronize data across servers and data centers automatically.

Table 2.3: Non-relational Data Storage Benefits

2.2.3 Data Access

At present, we have different storage solutions that can be applied to non-relational databases that, unlike relational ones, do not have a defined structure. Table 2.4 shows different storage solutions.

Type	Description
Document Data Store	A data store manages the documents employing a field organization. The names of the objects and their values make up a structure called a document. The JavaScript Object Notation (JSON) format is a recurring format in a data store.
Column-Family	The data is stored as if it were a table or a matrix through rows and columns. This alternative is similar to using a relational database, but the columns are organized into groups called families.
Key/Value	A key is used to index the data using a hash function. This provides a uniform and independent storage of the data. As each element is atomic, repeated elements are overwritten.
Graph Data	In a graph, we have a structure to move around the nodes and edges. The edges represent the relationship between the objects, while the nodes represent the objects.
Time-Series	A time series allows the data storage based on the time intervals from where the data occurs. They can represent large volumes of data in real-time, making them ideal for representing continuous phenomena or comparisons between various phenomena.

Table 2.4: Data Storage Solutions

Chapter 3

State of the Art

This chapter provides the preliminary information required to design an IoT platform. Each subsection offers information about the IoT platforms from a broad and current perspective, and they are organized as follows. Subsection 3.1 presents the definitions of IoT and IoT platform. Then, subsection 3.2 explains the requirements for an IoT platform. Finally, subsection 3.3 presents the common architectures for an IoT platform.

3.1 Definitions

The Internet of Things was born in the RFID community in 1999. Since its appearance has attracted the interest of both academia and IT enthusiasts, although it represents a relatively simple idea, the definitions assigned to the IoT are diverse [23]. Below are some of the most accepted definitions.

- The IEEE Internet of Things Journal (IoT-J), at its launch in 2014, described the IoT as: "A network of objects, with built-in sensors, that are connected to the Internet."
- The International Telecommunication Union (ITU), the United Nations agency defines the IoT in 2005 as "A network available anywhere, anytime, for anyone."
- The World Wide Web Consortium (W3C), the international community that develops standards for the Internet, defines the IoT as follows: "The Internet of Things is essentially about the role of Web technologies in facilitating the development of applications and services for the Internet of Things, that is, the virtual representation of physical objects.

Gubbi et al. [24], defines the Internet of Things as the "Interconnection of sensing and actuating devices that provide the ability to share information in a unified framework, enabling the development of innovative applications."

As can be seen, despite a general understanding of IoT meaning, there is no standard definition. As there is no exact definition of what IoT is, neither is there a definition of what an IoT system or platform is, so many approaches have been created over time. Here are some definitions of IoT platforms.

- In [25], an IoT system is defined as ”A set of protocols and standards that allow the execution of applications of the Internet of Things. The frameworks can enhance the development of applications for IoT through rapid implementation, interoperability, maintainability, security, and flexibility of the technology.
- In [26], an IoT platform is described as ”cloud-based software and related service packages.” IoT platforms allow application users to automate features that would otherwise require considerable time, effort, and expense.
- In [27], an IoT platform is defined as follows: ”The Internet of Things platform is a software that allows you to connect machines and devices and then acquire, process, transform, organize and store data collected by sensors.”

3.2 Requirements of IoT Platforms

This section details the requirements of an IoT platform, taking into account different approaches and needs [28][29]. Among the main requirements are service requirements, non-functional requirements, and architectural requirements.

Middleware Service Requirements

1. Resource Discovery

   The resources of an IoT platform include hardware devices such as RFID tags, sensors, smart devices, the analog-to-digital (A/D) converter devices, the infrastructure or network level information and communication module, and the services provided by those devices.

2. Resource Management

   Resources must be monitored and fairly allocated or provisioned, and resource conflicts must be resolved.

3. Data Management

   Data refers primarily to sensor data or any information that is relevant to applications.

4. Event Management

   There is usually a considerable number of events generated in the IoT applications that must be managed in the middleware.

5. Code Management

   Deploying code on an IoT platform alters an entire software ecosystem and must be adaptable and compatible with the middleware.

3.2.1 Non-Functional Key Requirements

1. Real-Time

   Services must be provided in real-time because some processes may depend on the operation time.

2. Scalability

   An IoT platform must be scalable to match the growth of the network and services.

3. Reliability

   An IoT platform must remain operational even when there are failures.

4. Availability:

   The middleware that handles the platform applications, especially the critical ones, must be available at all times. The frequency of errors must be small enough to achieve availability most of the time.

5. Security and Privacy

   Security must be considered in all functional and non-functional blocks since it is critical for the exemplary operation and protection of the platform. Knowledge of the context in the middleware can reveal sensitive information about the users.

6. Ease of Deployment

   The handling of the platform should not require expert knowledge or support. Complicated installation and configuration procedures should be avoided.

3.2.2 Architectural Requirements

1. Interoperable

   A middleware must work with heterogeneous devices, technologies, or applications. No additional effort should be required from the software developers.

2. Programming Abstraction

   Providing an API for application developers can be an essential functional requirement for any software project. Abstraction must be present in every layer of development so that the software does not require additional changes.

3. Service-Based:

   A middleware architecture must be service-based to provide outstanding versatility. This makes it easy to add new features or modify existing ones.

4. Adaptable

   The software must be able to adapt to changes in your environment or circumstances. In case the sensor network or other devices may change periodically.

5. Context-Aware

   The IoT system must know the context of the users, the devices, and the environment. The services that the platform has must be subject to the needs of the user and his environment.

6. Autonomous

   The devices must be able to interact and communicate with each other without direct human intervention. The master program running in the background should be able to solve any problem that arises in a short time.

7. Distributed

   The devices of an IoT system in a large-scale system exchange information and collaborate.

3.3 Architectural Approaches to IoT Platforms

This section reviews the most common characteristics and architectures used in IoT systems. Although there are different architectural approaches to IoT platforms, we take Guth et al. [30] as an example to better understand and even compare IoT platforms. The architectural models shown below provide an abstract view of the components of the IoT platforms. Figure 3.1 represents the reference architecture.

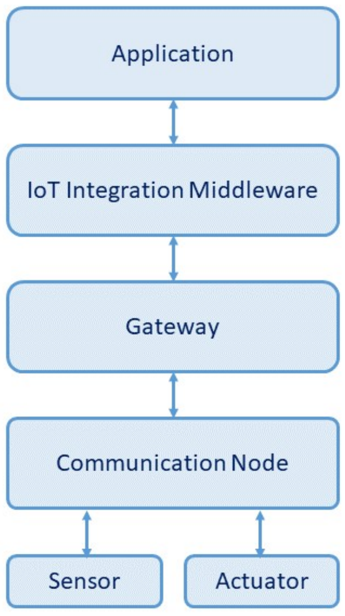


Figure 3.1: IoT Reference Architecture

1. Application

   This component represents the software used by the IoT integration middleware to obtain an overview of the physical environment by requesting data from the sensors or to control physical actions using the Actuators. Here is the software the end-user interacts with and the background software in charge of processing the orders.

2. IoT Integration Middleware

   The IoT integration software is in charge of receiving collected data from devices to process them using predefined rules and procedures. A Device can depend on the software developed in the middleware or load its program into an SoC. It can include all kinds of functionality required by a given computer system, such as the rule engine or graphical dashboards.

3. Gateway

   Communication nodes are often connected to a Gateway, a device that connects to the Internet or other systems. It is the component responsible for translating the different protocols and forwarding the communication between the Devices and other structures.

4. Communication Node

   A Communication Node is a hardware component connected to Sensors or Actuators through cables or wirelessly, even containing these components integrated. These devices are responsible for sending the data generated by the sensors and receiving the control signals from the application layer.

5. Sensors and Actuators

   This part forms the device layer, here are the devices capable of reading data and receiving signals. In chapter 2, we find a more detailed definition of what is a sensor and an actuator.

One of the essential parts of an IoT system is the middleware, which can be so complicated that we can find architectures only for this component, Nge et al. [3] defines three types: service-based, cloud-based, and actor-based.

Service-based middleware allows services to be added to IoT devices. In this architecture, we have a physical layer, a virtual layer where the server or cloud is located, and the application layer. Figure 3.2 shows these layers. The processing is available in the last two layers, which provides excellent performance, but higher bandwidth expenditure, so many procedures are parallelized.

Cloud-based middleware limits the number of devices that can be used, but users can easily collect the data. Functionality is limited to what is available in the cloud and can be accessed by a direct application or via an API. Figure 3.3 illustrates this method. With this architecture, information is exposed to the cloud provider's quality of service.

The architecture based on actors focuses on the plug-and-play model. As shown in Figure 3.4. This architecture is designed to be lightweight and adaptable to the development layers and has the benefit of connecting as many devices as needed without further user intervention.

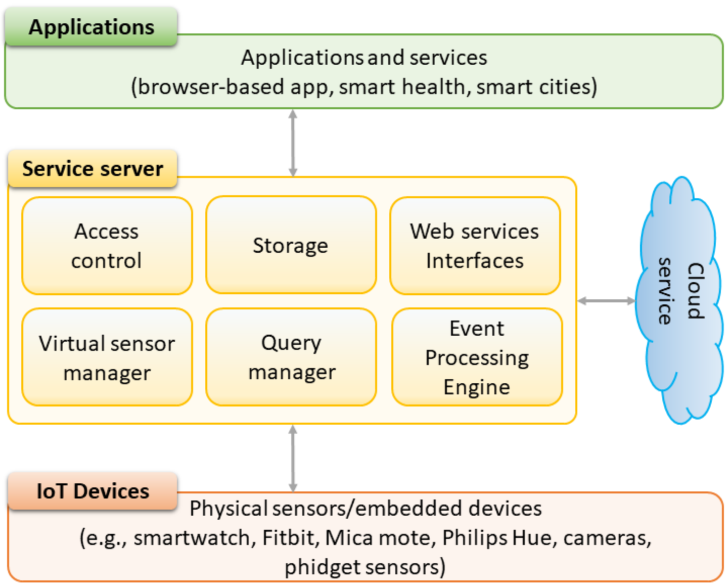

Applications
Applications and services
(browser-based app, smart health, smart cities)
Service server
Access control
Storage
Web services Interfaces
Virtual sensor manager
Query manager
Event Processing Engine
Cloud service
IoT Devices
Physical sensors/embedded devices
(e.g., smartwatch, Fitbit, Mica mote, Philips Hue, cameras, phidget sensors)

Figure 3.2: Service-Based IoT Architecture. Image from [3]

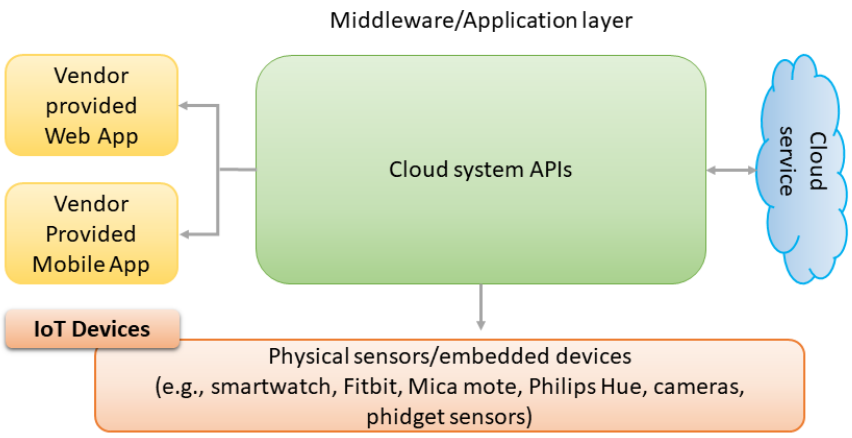

Middleware/Application layer
Vendor provided Web App
Vendor Provided Mobile App
Cloud system APIs
Cloud service
IoT Devices
Physical sensors/embedded devices
(e.g., smartwatch, Fitbit, Mica mote, Philips Hue, cameras, phidget sensors)

Figure 3.3: Cloud-Based IoT Architecture. Image from [3]

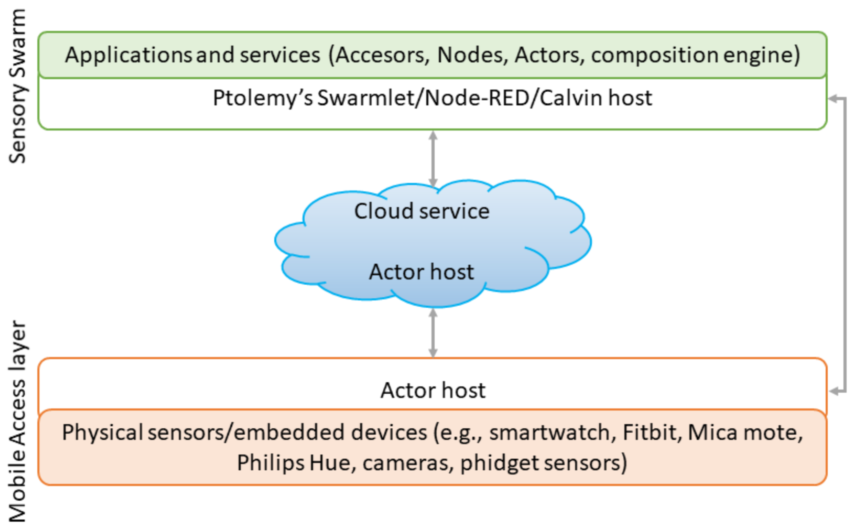

Sensory Swarm
Applications and services (Accesors, Nodes, Actors, composition engine)
Ptolemy's Swarmlet/Node-RED/Calvin host
Cloud service
Actor host
Mobile Access layer
Actor host
Physical sensors/embedded devices (e.g., smartwatch, Fitbit, Mica mote, Philips Hue, cameras, phidget sensors)

Figure 3.4: Actor-Based IoT Architecture. Image from [3]

Chapter 4

Methodology

This chapter presents the procedures applied to achieve the objectives, together with the justification for their use. First of all, it starts with an explanation of the steps for problem-solving in chronological order. Then, the model proposed is introduced, which includes the main components for the implementation of the IoT platform. Finally, we describe the data set and methods for analyzing the results.

4.1 Phases of Problem Solving

The following diagram shows the workflow that was taken to carry out this project.

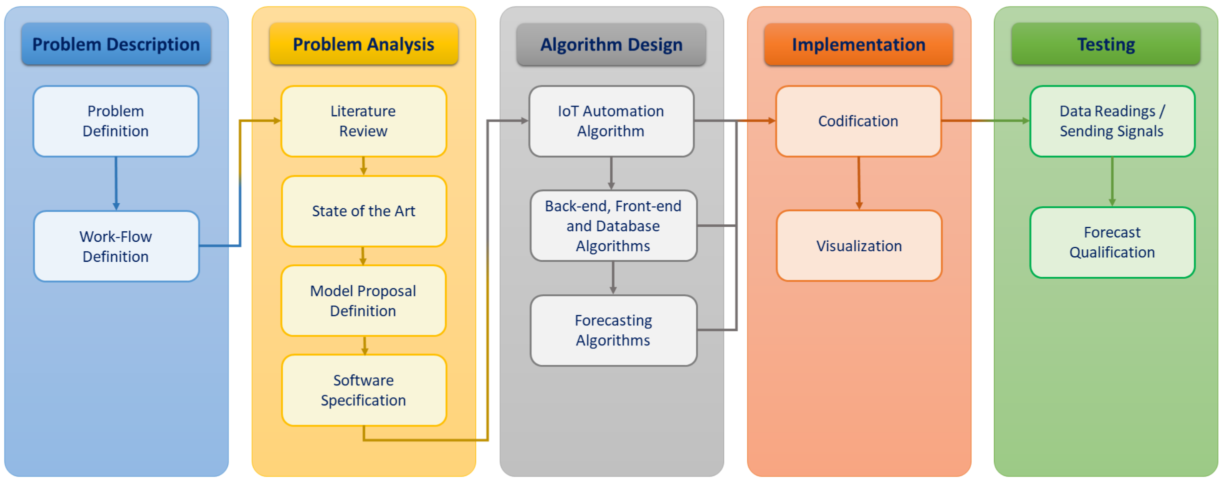


Figure 4.1: Phases of Problem Solving.

4.1.1 Description of the Problem

This phase lays the foundation for the development of the project as it presents the central theme, its potential problems, and how to address them. First, we propose the concept of greenhouse cultivation, the automation of this process, and its general characteristics.

Later, we address the sending, reception, and storage of the collected data. Later on, we present the model for both reading data and sending signals to control the system actuators. The workflow to design and develop the model proposal is also presented, as can be seen in Figure 4.1. In this case, it was decided to implement a platform from scratch, which allows total control over the processes that are executed and helps detect errors quickly. Then a stack is established to approach the implementation of the platform. Finally, artificial intelligence tools are evaluated to obtain significant predictions from the data obtained. Chapters 1 and 4 consolidate the steps mentioned in this phase.

4.1.2 Analysis of the Problem

This phase consists of the background and fundamental knowledge to a better understanding of this project. Chapter 2 presents this information in a systematic and orderly way, where we first analyze the internet of the things context used. Then, we deal with the web development tools used to develop the platform, and finally, we deal with data processing and cloud computing tools. In this sense, we focus on the problem of data flow. All the previously mentioned information was useful to describe a brief state of the art. According to this background, we defined a model proposal with contemporary software development tools, taking into account a modular programming design that allows a comfortable treatment and maintenance of the code, and data processing.

4.1.3 Algorithm Design

In this phase, the algorithms proposed in the model proposal were designed. These algorithms start with the one in charge of the intelligence of the IoT system. It is in charge of the automation of the greenhouse, the activation of the actuators, the reading of data employing sensors, and its later sending to the server. A master algorithm was implemented to solve the different types of requests in the Amazon Web Services (AWS) dependency, with the help of a complementary algorithm to deal with the database. Additionally, a couple of algorithms were developed to analyze the data and show predictions as a final result.

4.1.4 Implementation

In this phase, the software specifications used for each of the three stages of the system were determined. For the implementation of the IoT algorithms, the communication nodes use a variant of C++ (Sketch). In the implementation of the web application, a LAMP stack (Linux, Apache, MySQL, and PHP) was used with Node.js to retrieve MySQL data. In contrast, for the graphical interface, HTML, CSS, and JavaScript were used. In the final stage, Python was used to make the predictions of the environmental variables. The files that allow the correct platform performing are hosted in a dependency on Amazon Web Services. Chapters 2 and 4 consolidate the information mentioned in this phase.

4.1.5 Testing

This phase focuses on measuring and analyzing the performance of the IoT platform. For that propose, we perform several consecutive tests of data readings and sending of signals from the GUI. The thresholds established for each environmental variable, and the performance of the predictions are also evaluated. These results appear on the dashboard for better appreciation. Before presenting the predictions as a final result, we have to configure some parameters in the algorithm, being the most important the amount of data taken according to the date (daily, monthly or annual). Additionally, a prediction algorithms was evaluated to measure their performance by comparing the accuracy of their results. These steps are detailed in section 4.3.

4.2 Model Proposal

This work proposes an architecture consisting of four layers: Device layer, Network layer, Application Services Platform, and Application layer. This section provides details of each of these. In a summary of this architecture, the device layer contains digital sensors, analog sensors, and actuators, that interact with the greenhouse environment. The network layer has a twofold operation that uses WiFi and all of the well-defined routing protocols and processes that it Includes. Application Services Platform consists of the data sending en receiving data protocol, and it consists of two main phases. First, it sends all the data using the MQTT protocol; and second, it connects the back-end end with the broker EMQX using WebSockets. The application layer is implemented with a web application that retrieves, stores, analyzes, and generates visualizations of the data. Figure 4.2 presents a diagram of the proposed architecture with its different hardware and software components.

4.2.1 Device Layer

The device layer, the first of our architecture, interacts with the environmental variables inside the greenhouse. The variables to be measured are the humidity and temperature of the environment, the humidity of the soil, the intensity of the light, and the water level in the reservoir. Our systems use several sensors to record these variables.These sensors include the DHT11 sensors to measure humidity and temperature of the environment, FC-28 sensors to measure the humidity of the soil, a module with a photo-resistance to measure the intensity of light, and a water level sensor to measure the water level in the reservoir. Figure 4.3 shows the diagram of the electronic connections between the NodeMCU and the other devices.

Once the sensors perform the data readings, they send the data to the NodeMCU, a development board based on the ESP8266 chip. The collected data is sent through the node using WiFi technology to a broker using the MQTT protocol. It is important to emphasize that connectivity to the Internet is required. The user can control the activation of the artificial light, the intractor fan, the extractor fan, and the irrigation system by sending control signals. The NodeMCU board is in charge of receiving the user signals and carrying them out through relays connected to the node. The flow chart of the system can be seen

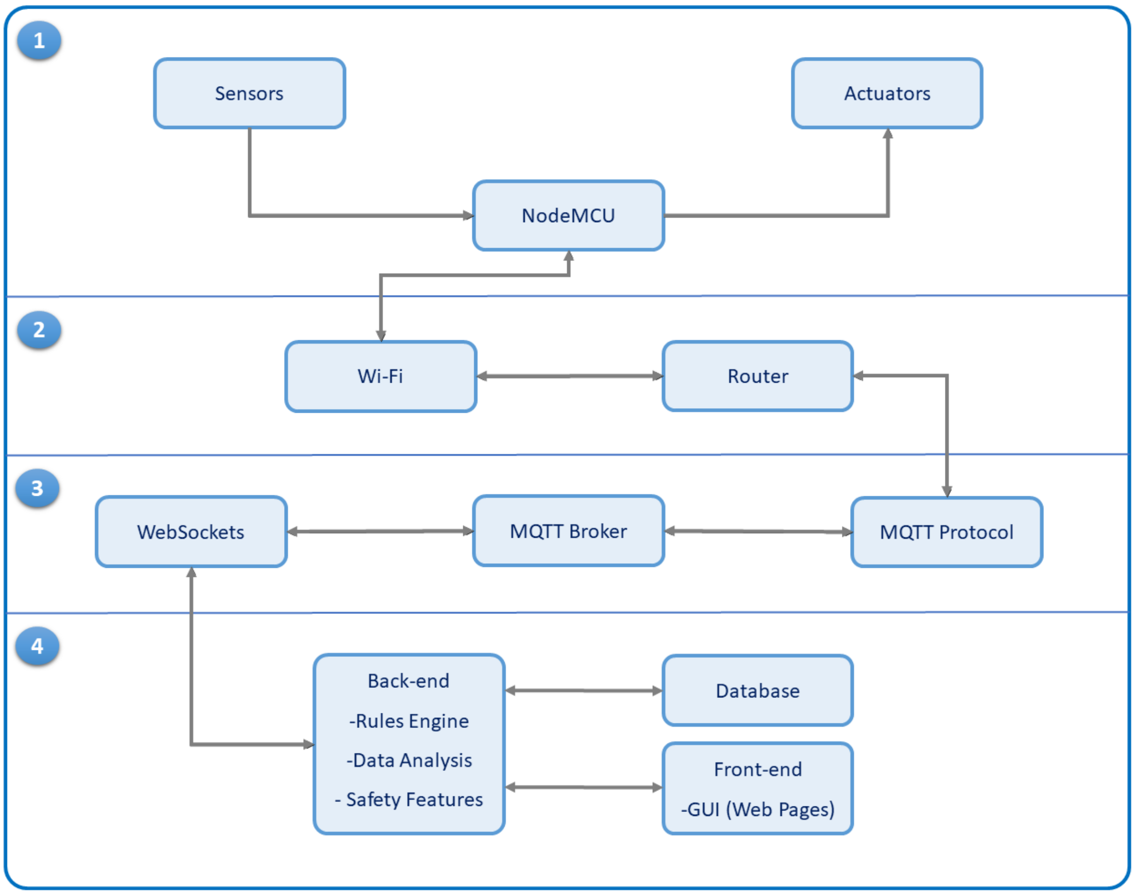


Figure 4.2: Greenhouse Monitoring and Control System Based on IoT Technologies.1) Device Layer, 2) Application Platform Layer, 3) Network Layer, 4) Application Layer.

in Figure 4.4.

4.2.2 Network Layer

The network layer is in charge of communication with the internet. The wireless transmission medium chosen for the project is WiFi. According to Shi [4], although WiFi is one of the most energetically expensive technologies, it also has the highest data transmission speed, and this is relevant if we want to connect tens or hundreds of nodes to the network. This layer not only transmits to the next layer the information related to the environmental factors of the device layer but also transmits to the device layer the control commands of the application layer so that the devices take the actions corresponding to each command. This layer remains unmodified compared to the standard network layer of wireless systems WiFi 802.11 used commercially by routers found at home.

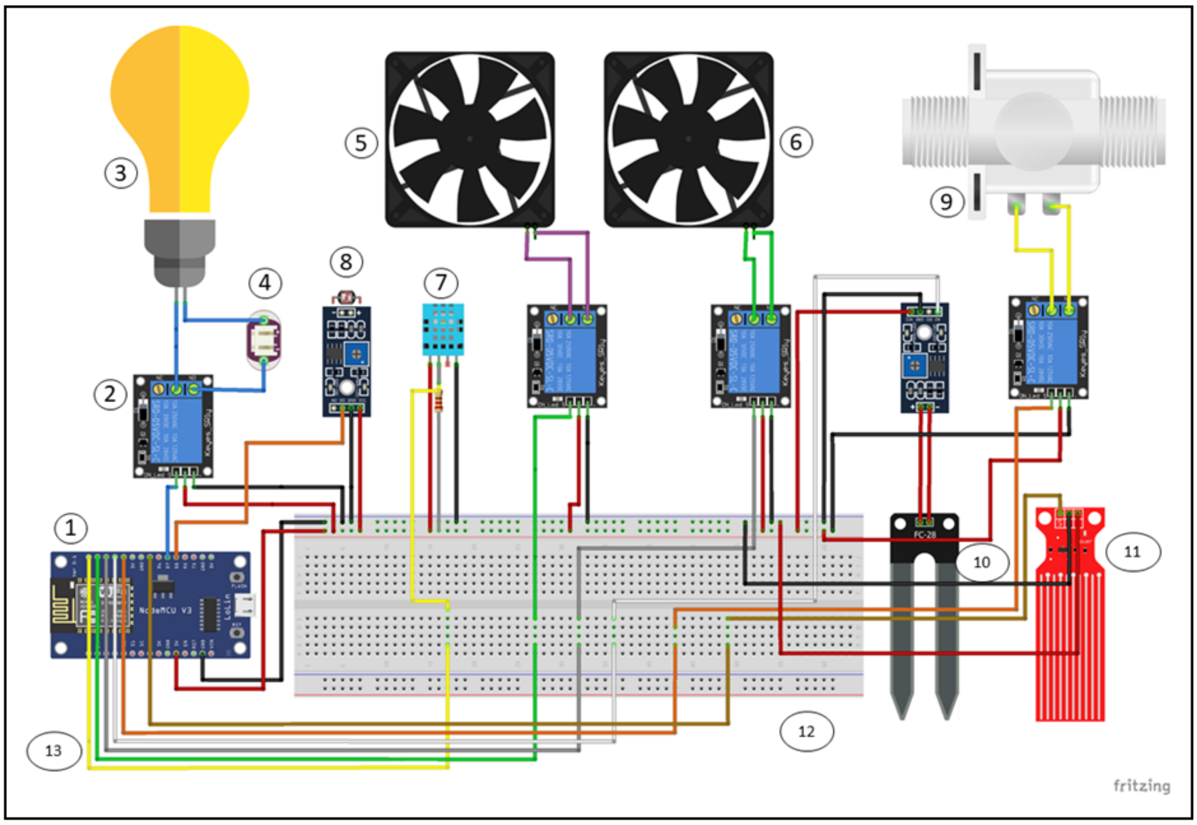


Figure 4.3: Circuit Diagram. 1) NodeMCU 2) Relay 3) Artificial Light 4) Plug 5) Intractor Fan 6) Extractor Fan 7) Humidity and Environment Temperature Sensor 8) Light Sensor 9) Water Pump 10) Soil Moisture Sensor 11) Water Level Sensor 12) Protoboard 13) Jumper Wires.

4.2.3 Application Platform Layer

This layer is in charge of data sending-receiving protocols and the application that manages it. The most famous case is to use HTTP as a protocol with an HTTP server. However, it is not very useful in some IoT operations. To save energy and to obtain better efficiency, it is necessary to search for other protocols and protocol manager alternatives.

The protocol used for data transmission is Message Queuing Telemetry Transport protocol (MQTT), an ISO standard protocol (ISO/IEC 20922) developed by IBM in 1999 [21]. MQTT is a machine-to-machine (M2M)/IoT connectivity protocol designed as an extremely light publication/subscription message transport since it can work with low network bandwidth [20]. Communication between the broker and the IoT platform is carried out using the WebSockets protocol, a technology that provides a two-way, full-duplex communication channel on a single TCP socket [31].

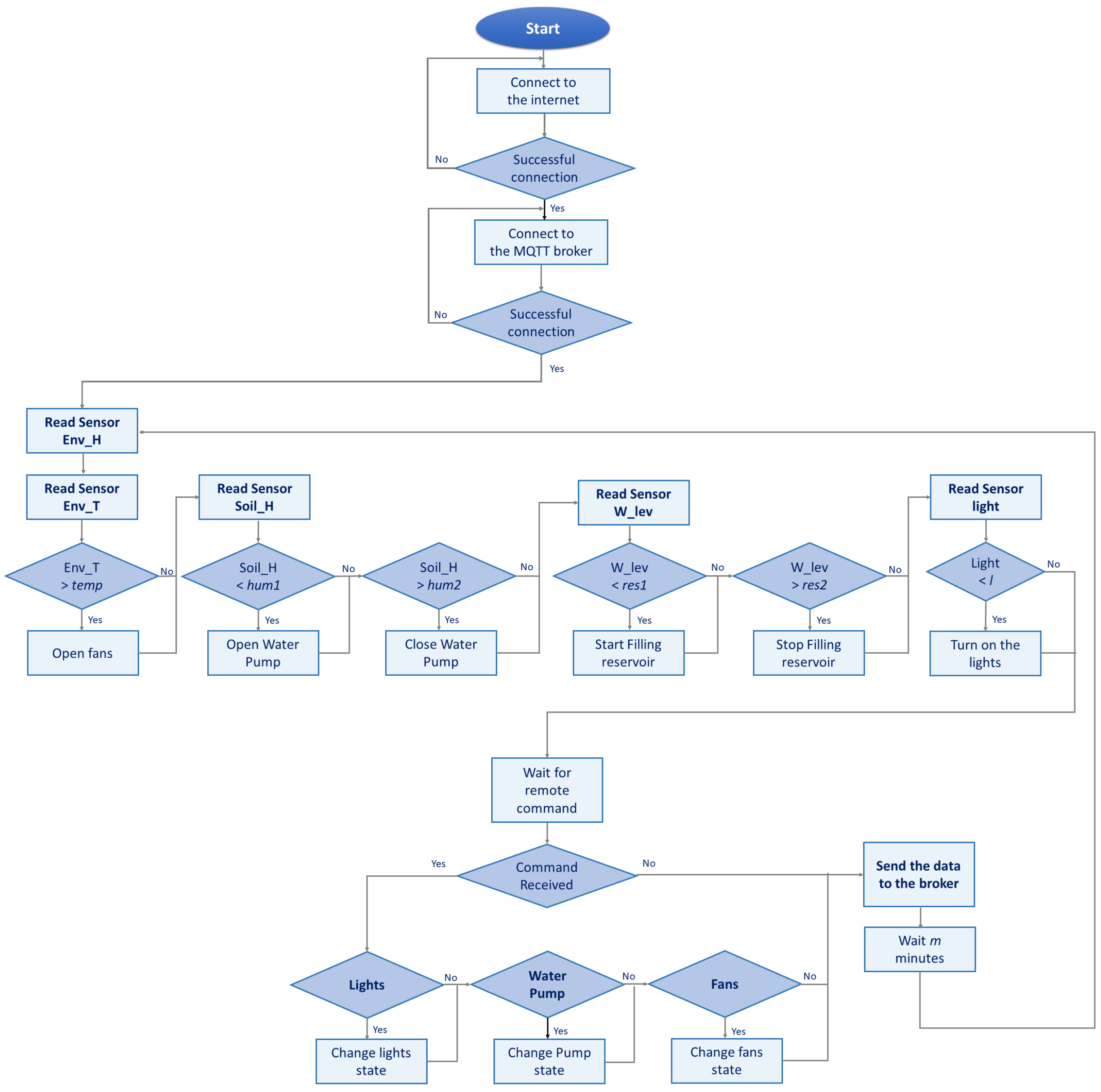


Figure 4.4: Flowchart Showing the Operation of the Monitor and Control System.

4.2.4 Application Layer

This layer is responsible for storage, decision making, summary, and statistics. Here, the algorithms and models for intelligent control and prediction are established. This layer is composed of cloud computing, database, rules engine, machine learning algorithms, and other essential processing technologies. As mentioned in Chapter 2, this web application is based on the LAMP stack. In Figure 4.5 you can see the structure of the application architecture.

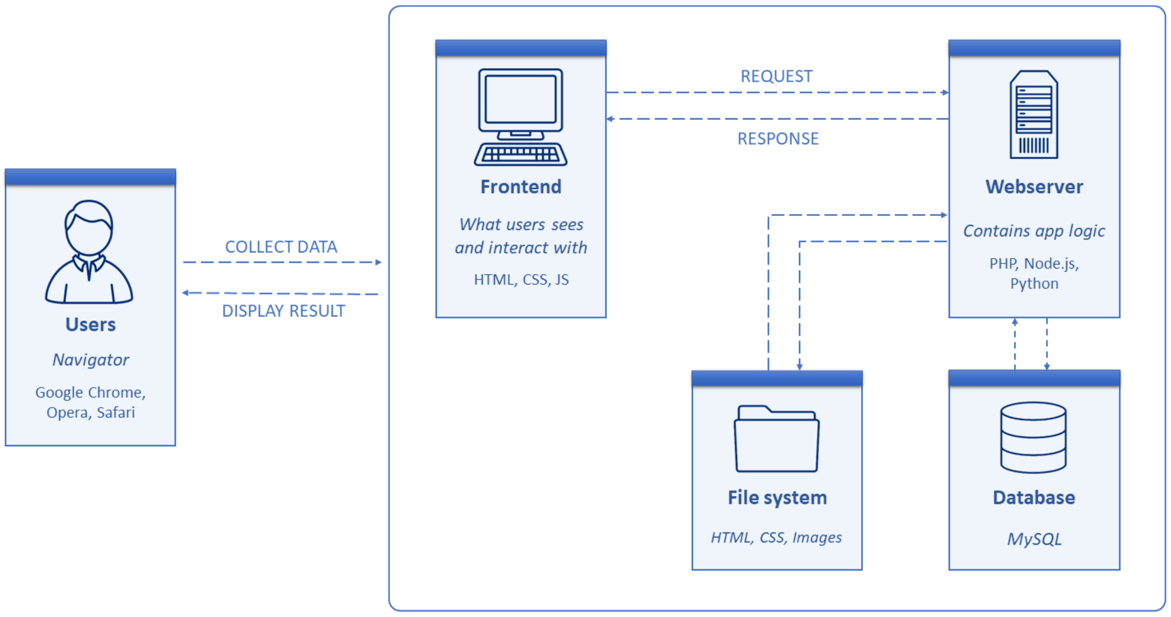


Figure 4.5: Structure of the Application Architecture

Back-End

We use a relational database (MySQL) to store all the data coming from the different sensors. All the procedures in the backend use the PHP programming language and, to retrieve the data from the database, we used Node.js, which acts as an execution environment for JavaScript. EMQX broker was used to sending and receiving data using the MQTT protocol. The web application components were hosted on a VPS at Amazon Web Services.

Data Analysis

For data analysis and prediction, we used a machine-learning algorithm called Long Short-Term Memory (LSTM). This method allows us to produce a forecast of a time series with discrete values. The programming language for developing the LSTM neural network was Python, as it offers convenient tools for fast and easy development of AI algorithms along with the TensorFlow backend. For training the model, we used the platform Google Colab, which offers a high-capacity remote computer specially dedicated to training AI algorithms free of charge.

The LSTM neural network is a variation of the traditional recurrent neural networks (RNNs) developed by Hochreiter [32]. The objective of LSTMs is to overcome the problem of long-term dependencies, which makes RNNs not viable for processes that depend on long term memory, such as time series prediction. An LSTM neural network is formed by a chain of LSTM units which share information like an RNN. The main difference lies in the LSTM unit which has three gates that efficiently manage the information.

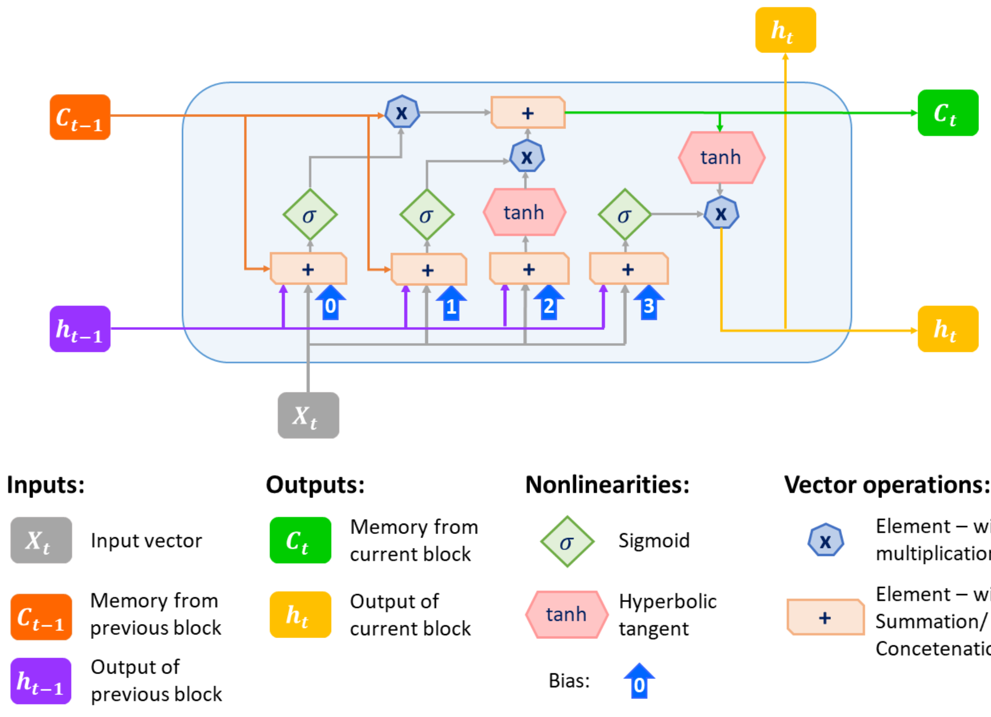

C_{t-1}
h_{t-1}
X_t
h_t
C_t
h_t
x
+
x
tanh
σ
σ
tanh
σ
x
+
+
+
+
0
1
2
3
Inputs:
X_t Input vector
C_{t-1} Memory from previous block
h_{t-1} Output of previous block
Outputs:
C_t Memory from current block
h_t Output of current block
Nonlinearities:
σ Sigmoid
tanh Hyperbolic tangent
Bias: 0
Vector operations:
x Element – wise multiplication
+ Element – wise Summation/ Concetenation

Figure 4.6: Structure of the LSTM Unit

In an LSTM unit [33], the essential communication line is the one that connects the previous state of the cell, C_{t-1}, with the new state, C_t. Through this line, useless information will be deleted, and update relevant information will be added. The gates in the LSTM are composed of a sigmoid layer and a pointwise operation. In the first gate, also called "forget gate," the information from the previous unit, h_{t-1}, and the previous layer is received, the gate decides what information will be deleted by the equation: equation 1. The second gate, called "input gate," adds information to the new state; there are two parts in this process, first, the sigmoid layer decides what information will be added by the equation 2, then the tanh layer gives a list of possible values by equation 3. The output of the gate is given by equation 4. With the information of the first and second gates, the new state is equation 5.

The output for the next unit and the next layer is taken from the new state information. A sigmoid layer decides what information will be the output by equation 6 and a tanh layer put the values from C_t between -1 and 1, the output is given by equation 7.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4.1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4.2}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4.3}$$

$$i_t * \tilde{C}_t \tag{4.4}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4.5}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4.6}$$

$$h_t = o_t * tanh(C_t) \tag{4.7}$$

Front-End

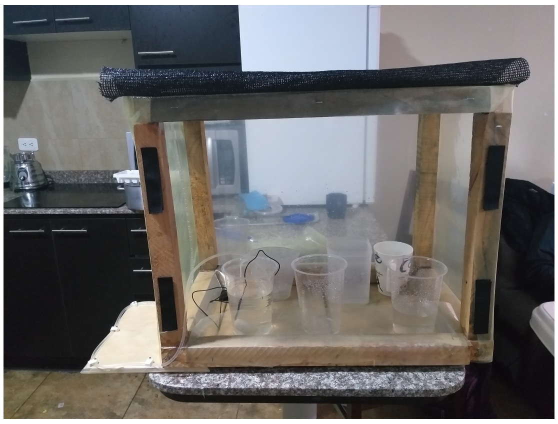
This is the highest level of architecture and the place where the IoT value and usefulness are most evident. This layer is where crop monitoring and control, early warning, and disease management are performed. In this layer, there is an authentication system to perform user control. In the control panel, the data from each sensor, the buttons to activate or deactivate the control devices, and the real-time graphs of the data can be seen through a graphical user interface.

4.3 Experimental Setup

The project described below represents the solution to the initial approach to the problem. This solution starts with the construction of a greenhouse model. The built structure has the appropriate technology to automate each of the processes to allow the optimal conditions for the growth of the crop. The development of this project can be done in two ways: in an already existing greenhouse or the construction of a greenhouse from scratch. For this project, it was decided to opt for the construction of a scale greenhouse and collect data from inside.

Once the model was built, we adapted the connections, both physical and wireless, of the sensors and actuators with the communication nodes, which were then connected to a global gateway. The second part of the project is the implementation and adaptation of the software in charge of the wireless transfer of messages, for which we need to choose a protocol and an application to handle it. For this project, the application is an EMQX Broker that uses the MQTT protocol. The final stage of the project is developing our web application, which includes the rules engine, the database, the graphic interface, and the data analysis.

After each phase of the project, general checks were made to perform failure tests and modifications to adapt to the expected growth of the platform.


(a) Inside View of the Greenhouse Model

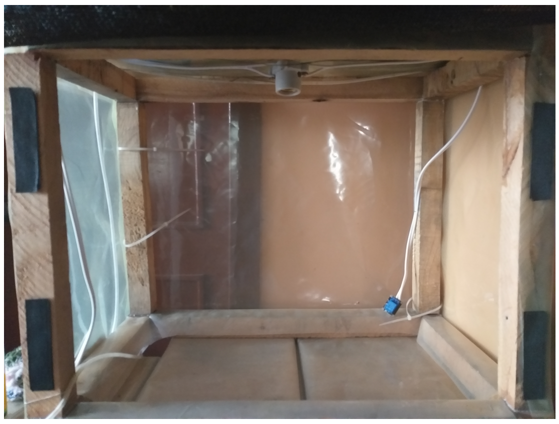

(b) Outside View of the Greenhouse Model

Figure 4.7: Views of the Greenhouse Model

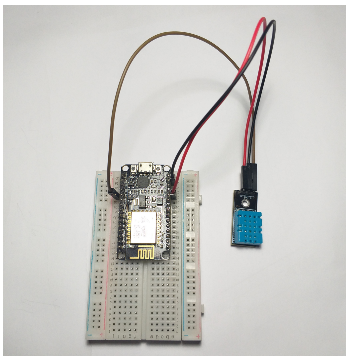
4.3.1 Preparation and Construction of the Greenhouse Model

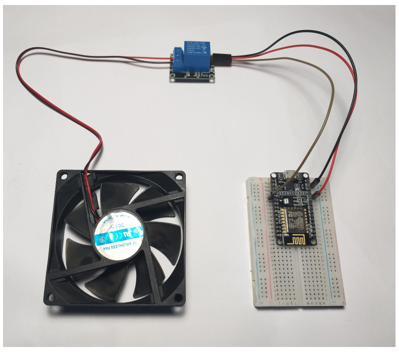
In order to have a reasonably good approximation of a real greenhouse, the same materials and considerations were used to make the data collection very reliable. The most common greenhouses are chapel-type greenhouses, so our model construction was based on one of them. The primary materials for the construction of the model were wood, polyethylene mesh, thick greenhouse plastic. The final structure had the following dimensions 0.5m x 0.7m x 0.5m as can be seen in Figure 4.7.

The sensors and actuators used organize into three systems that include the measurements of all the essential variables to monitor and control the crops. The air conditioning system is in charge of the ventilation of the interior of the greenhouse. The irrigation system is in charge of the irrigation and humidity measurements. Finally, the lighting system is in charge of activating the lights according to the structure luminosity.

Climatization System

The climatization system aims to monitor the humidity and temperature of the environment through sensors. In the IoT algorithm that runs in the communication nodes, thresholds were defined for each of the variables. If the sensor measurements exceed these values, the fans are activated to give a higher flow of air and reduce the temperature and humidity inside the structure. This system has a pair of fans, an extractor responsible for expelling the air inside the greenhouse and an intractor that introduces air from outside to the structure. Figure 4.8a shows the climatization system components. Besides controlling the previously mentioned environmental variables, it also fulfills the function of reestablishing the CO2 levels, vital for the correct development of the plants. The electronic devices used for the implementation of this system are shown in Table 4.1:


(a) Temperature/Humidity Sensor DHT11


(b) Intractor Fan - Relay Actuator

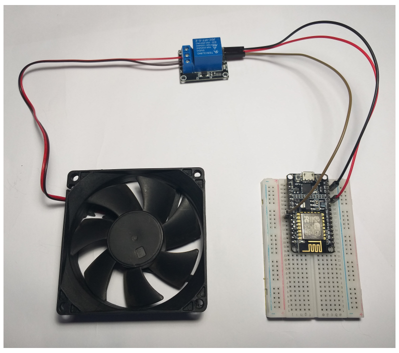
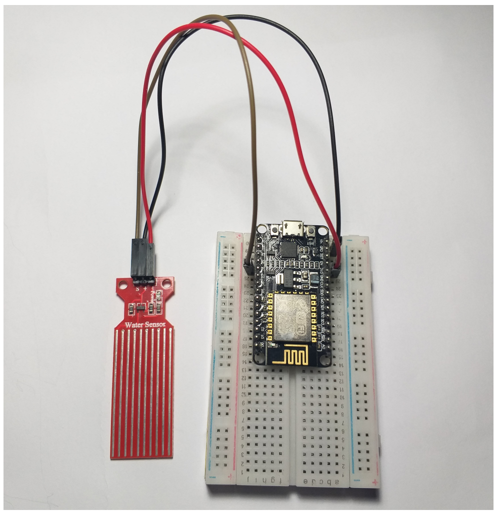

(c) Extractor Fan - Relay Actuator

Figure 4.8: Climatization System

Irrigation System

The irrigation system is probably the most critical system of all and also the most complex. This system measures the humidity of the growing substrate and continues watering it according to the IoT algorithm specifications. Since irrigation is the most critical factor, it must be implemented carefully. Our irrigation system is divided into two parts, the sensors, and the water pipes. The first part consists of the soil moisture sensor, which gives the value of the percentage of moisture in the soil, and the water level sensor that measures the amount of water in a container. These components are shown in Figure 4.9. The second part of the system consists of the water pump that absorbs the water from the reservoir, and the fine plastic hoses that are connected to the pump and distribute the water through drip irrigation, these components are shown in Figure 4.10.


(a) Water Level Sensor

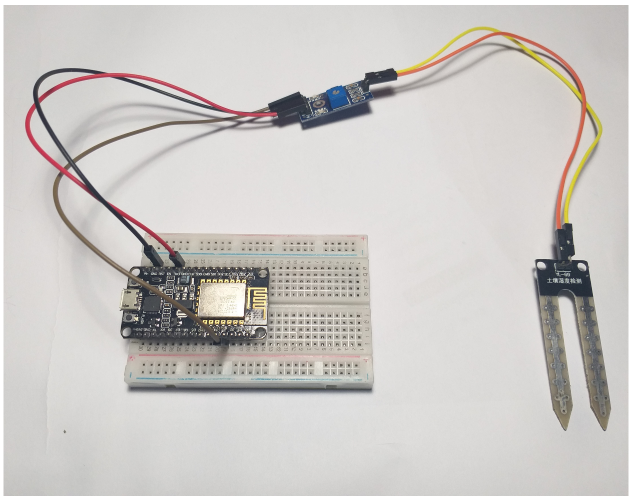
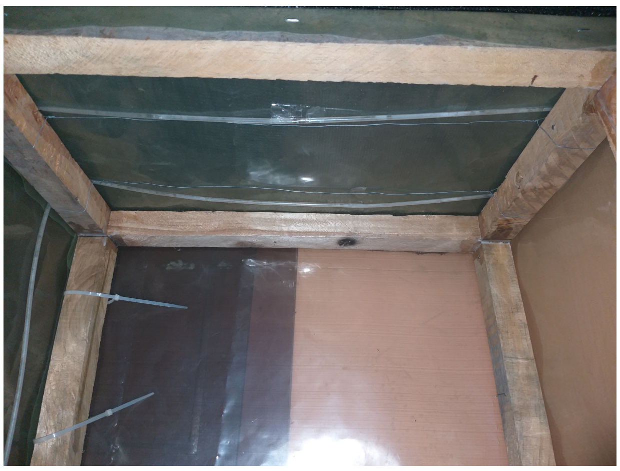

(b) Soil Moisture Sensor

Figure 4.9: Irrigation System - Part 1

(a) Water Pump Connected to the Hoses


(b) Drip Irrigation - Top View

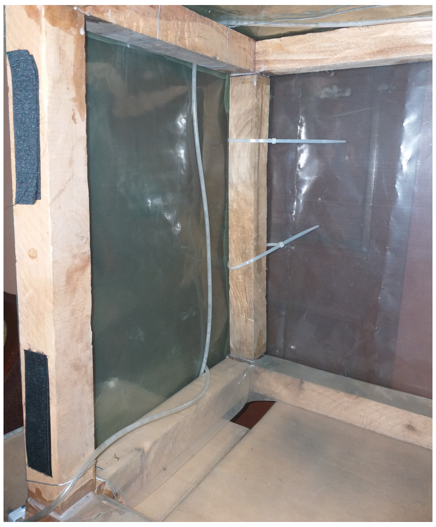

(c) Drip Irrigation - Side View

Figure 4.10: Irrigation System - Part 2

Device	Description
Relay Module 5v	A small and powerful relay module for safely switching larger voltages. This module works on 5V and can switch up to 10A.
DHT11 Sensor	DHT11 module can measure the temperature and humidity of the environment. This has a sampling rate of a maximum of 1 hz, whereby the output is sent digitally. In addition, a borehole (3 mm) has been provided to easily attach the module.
Cooler Fan 12v	Cooler fan to create an airflow that introduces air to keep components cooler. This fan works on 12V and has a connector with 2 pins.
NODEMCU-v3 ESP-12E	NodeMCU is a small Arduino compatible Wifi board. It is mounted around the well-known ESP8266 ESP-12 and exposes all its pins on the sides. It also offers more advantages such as the incorporation of an integrated voltage regulator, as well as a USB programming port. It can be programmed with LUA or with Arduino's IDE.

Table 4.1: Climatization System Components

Lighting System

The lighting system monitors the amount of light entering the greenhouse through a light sensor. Depending on the values obtained by the sensor, a relay turns on or off a light bulb or series of light bulbs that act as a light source. Since there are varieties of plants that depend on light hours to switch from one growth phase to another, the brightness threshold can be changed manually using a potentiometer connected to the light sensor. The components of this system can be seen in Figure 4.11.

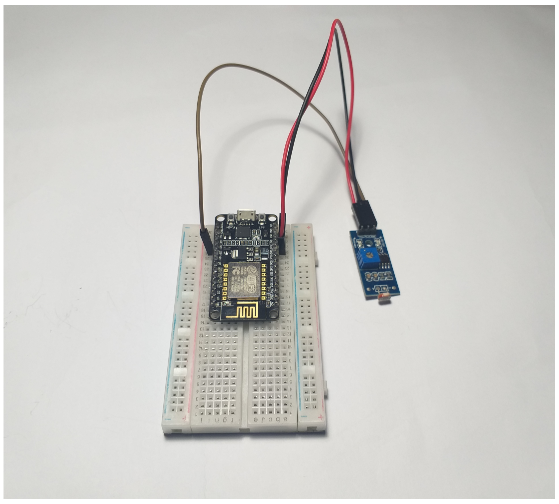
In the following portion of code, we can see the first part of the IoT algorithm responsible for managing the sensors and actuators. In this section, you can see the initialization of the analog and digital ports and a part of the primary functions that each sensor has.

```
void setup() {

  pinMode(2, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(8, INPUT);
  Serial.begin(9600);
  dht.begin();
```

Device	Description
Submersible Pump 2.5V - 6V	Small pump to move water or other liquid. This submersible pump is placed in the liquid, where a hose of 7.5 mm can be placed at the outlet can be connected to propel the water up to 1 meter.
PVC Hose 7.5mm Transparent	This hose has an outside diameter of 9 mm and an inside diameter of 7 mm. The hose is made of strong and flexible PVC with a transparent color. The hose is sold per 2 meters, with a decrease of several meters the hose is interrupted per 2 meters.
Ground Humidity Ssensor Module	It can be used to measure the humidity in the ground. This makes it possible for the NodeMCU to give a notification if the plant needs water. This module has both the possibility of digital and analog output, whereby the digital output threshold can be set by means of a potentiometer.
WATER LEVEL SENSOR	This sensor is used to indicate the water level in a tank, with this we can avoid the overflow of water, and at any time we can know the water level in a reservoir.
NODEMCU-v3 ESP-12E	NodeMCU is a small Arduino compatible Wifi board. It is mounted around the well-known ESP8266 ESP-12 and exposes all its pins on the sides. It also offers more advantages such as the incorporation of an integrated voltage regulator, as well as a USB programming port. It can be programmed with LUA or with Arduino's IDE.

Table 4.2: Irrigation System Components


(a) Light Sensor

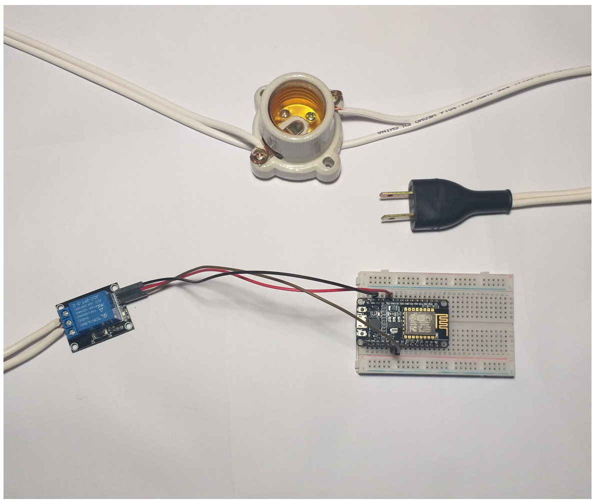

(b) Light Source

Figure 4.11: Lighting System

Device	Description
Light Sensor	This Light sensor module based on photodetector GL5528 detects the light intensity of the environment. As the resistance of the sensor varies depending on the amount of light it is exposed to, the output voltage changes with the light intensity.
Plug and Socket for spotlight	Common plug and socket for spotlight to connect light source.
Relay Module 5v	A small and powerful relay module for safely switching larger voltages. This module works on 5V and can switch up to 10A.
NODEMCU-v3 ESP-12E	NodeMCU is a small Arduino compatible Wifi board. It is mounted around the well-known ESP8266 ESP-12 and exposes all its pins on the sides. It also offers more advantages such as the incorporation of an integrated voltage regulator, as well as a USB programming port. It can be programmed with LUA or with Arduino's IDE.

Table 4.3: Lighting System Components

```

}

void loop() {

//water level

  int valo = analogRead(A0);

  Serial.print("water level: ");
  Serial.println(valo);

//relay

  int relay = 2;

  if (  digitalRead(8) == HIGH) {
        digitalWrite(2, HIGH);
  }
  else{
        digitalWrite(2, LOW);
  }

//humidity-temperature

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  Serial.print("Hymidity: ");
  Serial.println(h);

  Serial.print("Temperature: ");
  Serial.println(t);

//light sensor

    byte valor=digitalRead(8);

    if (  digitalRead(buttonPin) == HIGH) {
        digitalWrite(ledPin, HIGH);

    Serial.print("Light sensor: ");
    Serial.println(valor);

//soil moisture

  int lect = analogRead(A2);
  Serial.print("Soil Moisture: ");
  Serial.println(lect);

//Water pump

int state1 =HIGH;
digitalWrite(3, state1);
Serial.print("light: ");
Serial.println(state1);
delay(2000);
}
```

4.3.2 Web Application Development

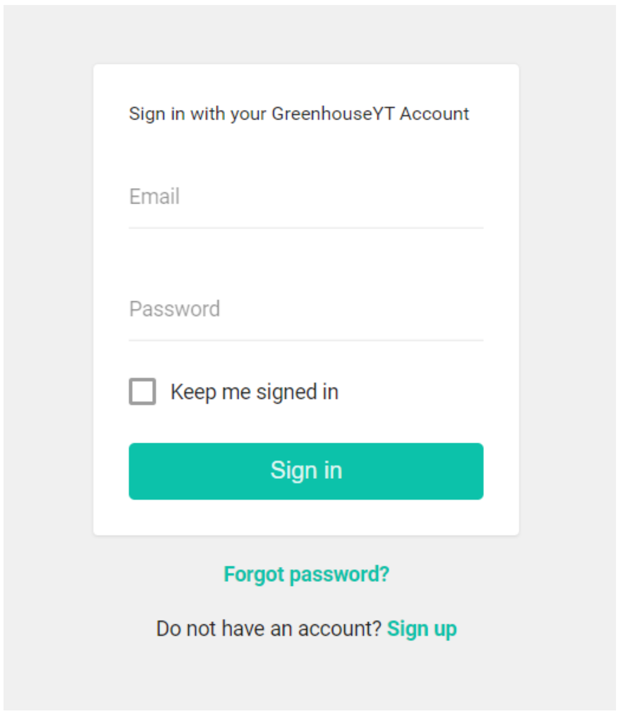
Interface Design

The first task of the application is to display an authentication form that asks for the e-mail or username and password. The data collected in this form is sent to the server to be compared with the existing user list; otherwise, the user is allowed to create a new account. The page shown on the screen also has ”Keep me signed in” buttons that remind the user and password values so that the session is not closed when the user leaves the page. At the end of the questionnaire, there is the phrase ”Forgot password? which allows the user to recover the password through the e-mail. The authentication form can be seen in Figure 4.12.

If the user passes the authentication system, the application displays a page with the main dashboard can be seen in Figure 4.13 and Figure 4.14.

On this page, first appears the monitoring section at the beginning, in which we can see two rows. In the first row, we can see the states of light, water, and ventilation through individual boxes containing the words ”ON” if the device is activated or ”OFF” if the device is turned off together with icons representing each variable. The second row contains boxes indicating values through dynamic graphics that are updated periodically. These graphics show a numerical value surrounded by a colored line that indicates the percentage that this number represents together with the name of the variable. This section is shown in Figure 4.15.

In the second part of the screen, we can see the control section, here we can see boxes like the ones at the beginning, but with interactive buttons that serve to activate or deactivate each actuator. These buttons allow us to modify the variables over the background working IoT algorithm, which allows us to modify the system temporarily without having to alter the general algorithm. This sections is shown in Figure 4.16.



Sign in with your GreenhouseYT Account

Email

Password

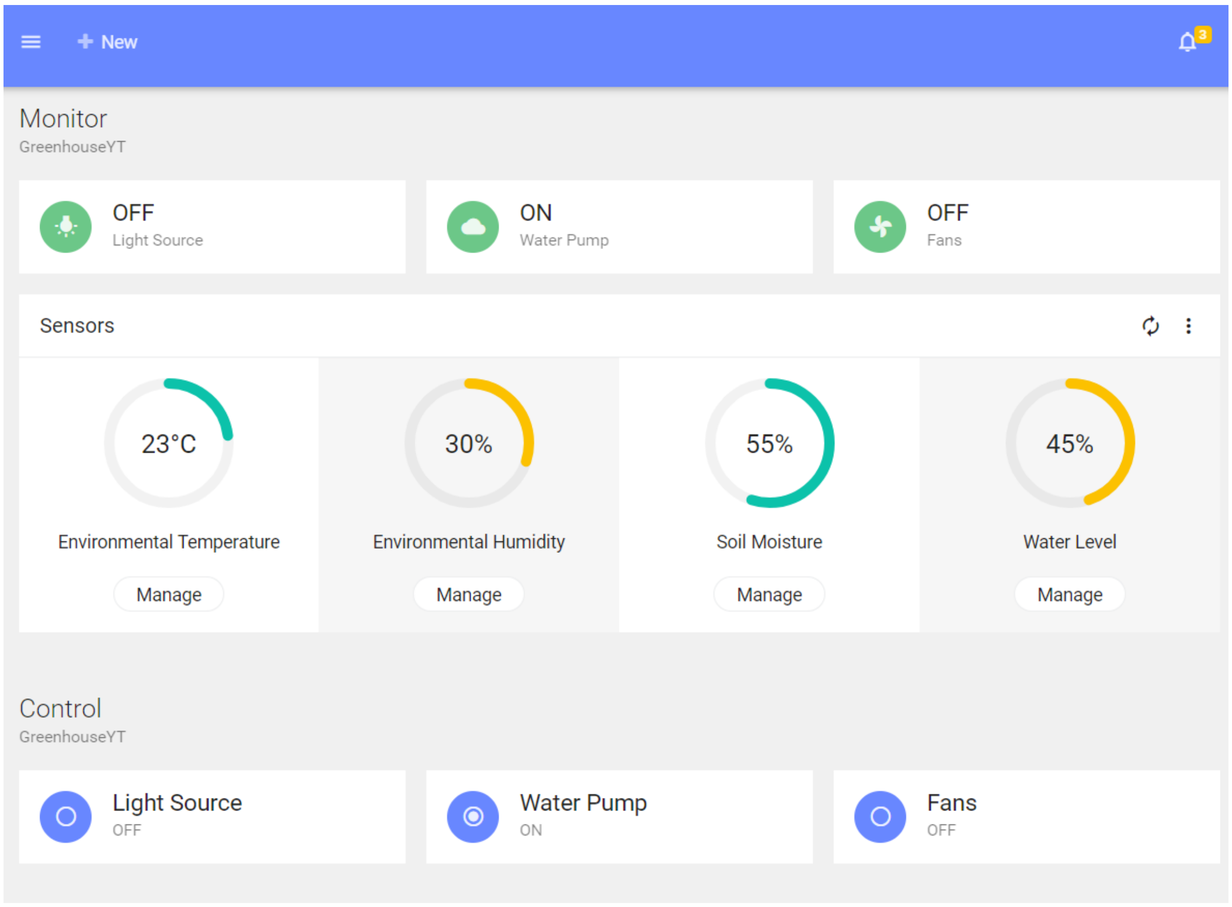
Keep me signed in

Sign in

Forgot password?

Do not have an account? Sign up

Figure 4.12: Authentication System



Monitor

GreenhouseYT

OFF Light Source	ON Water Pump	OFF Fans

Sensors

23°C	30%	55%	45%
Environmental Temperature	Environmental Humidity	Soil Moisture	Water Level
Manage	Manage	Manage	Manage

Control

GreenhouseYT

Light Source OFF	Water Pump ON	Fans OFF

Figure 4.13: Dashboard - Page 1

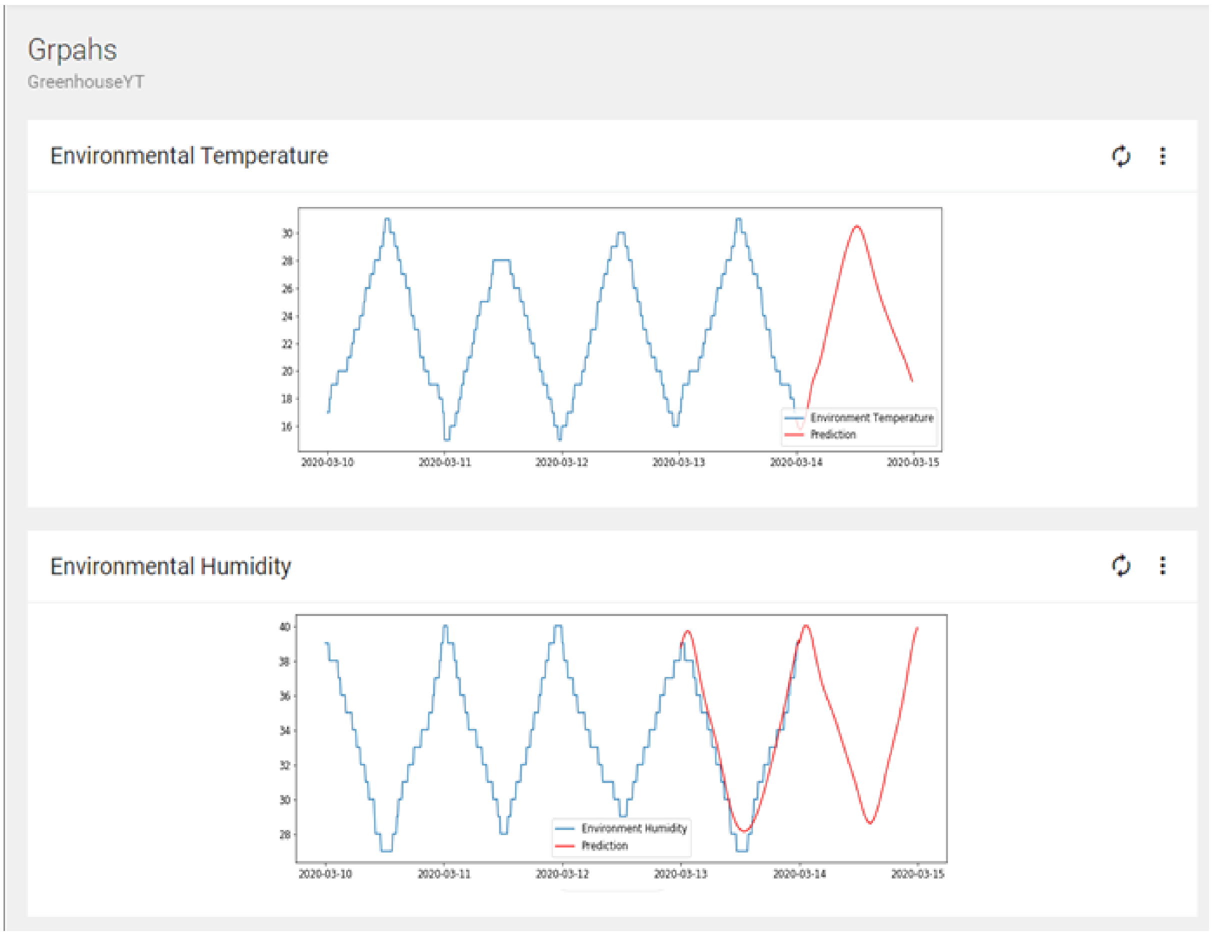


Figure 4.14: Dashboard - Page 2

Finally, in the third part of the page, we can see the real-time graphs of the values taken periodically by the sensors. These graphs are significant because they allow us to visually analyze the data and analytically and obtain predictions through data analysis methods such as neural networks. In our case, a recurrent neural network was used, known as LSTM. In Figure 4.14 we can see in blue lines the values of the sensors while in red lines the predictions obtained with the previously mentioned network. Measurements and forecasting is shown in Figure 4.14.

Database

For this application, a relational SQL database with one table was employed. In this structure, the values of each sensor, the states of each actuator, and the exact date of these records were stored. The structure of this database is shown in the table 4.4 below, together with the type of data and a brief description.

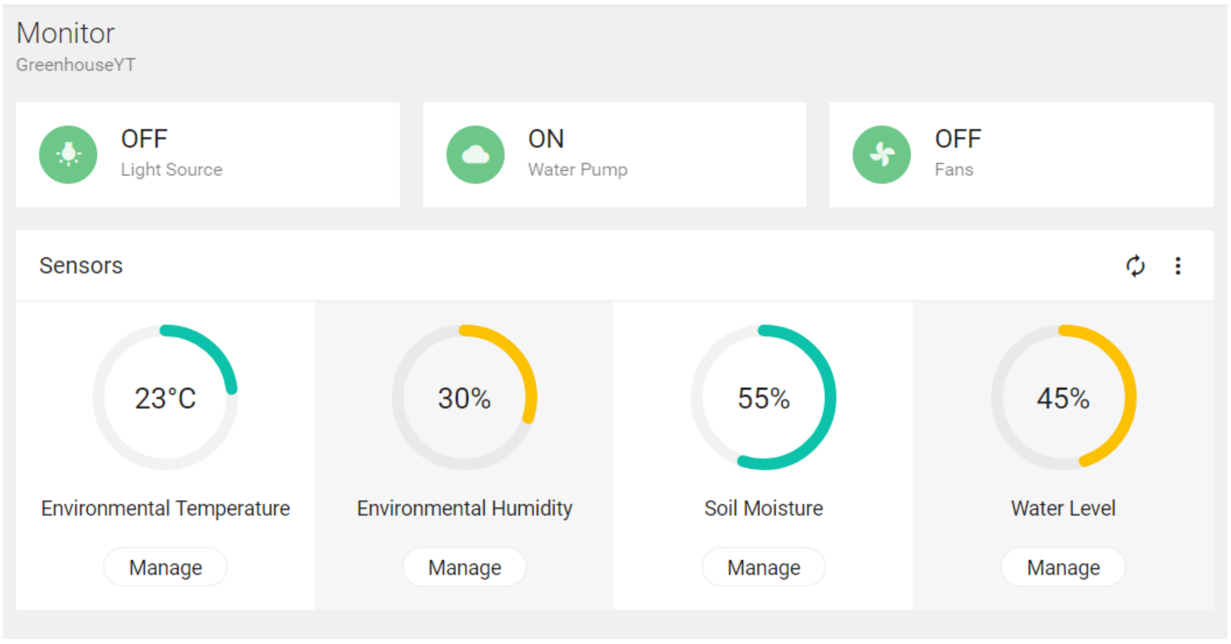


Figure 4.15: Monitoring Module

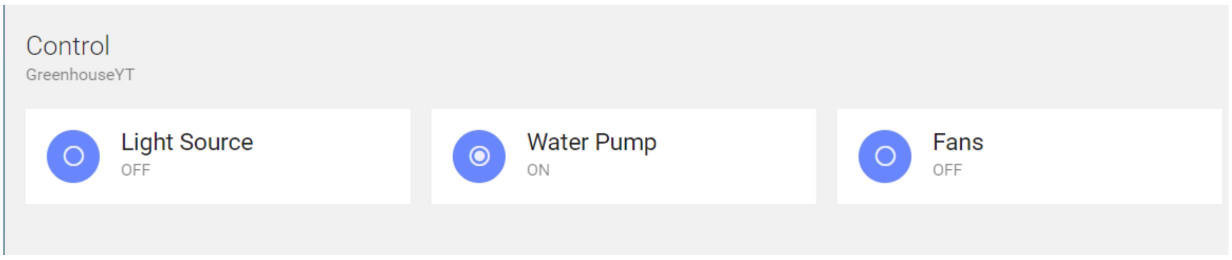


Figure 4.16: Control Module

Variable	Data Type	Description
date	TIMESTAMP	A value indicating the exact date and time of data recording.
hum-env	INT	The percentage of humidity in the environment.
temp-env	INT	Temperature in degrees centigrade.
hum-soil	INT	The percentage of soil moisture.
s-light	INT	The number of lumens.
l-water	INT	The percentage of the water level.
light	CHAR	The word "ON" or "OFF" depending on the light status.
water-p	CHAR	The word "ON" or "OFF" depending on the state of the water.
fans	CHAR	The word "ON" or "OFF" depending on the status of the fans.

Table 4.4: Database Structure

Back-end

This part is in charge of the intelligence of the application. Here, the rules engine was developed, and the connection with the broker to be able to project the data to the user in the graphic interface. For this part, we used a mixed stack based on LAMP, to which we added Node.js to retrieve values from the database. It is also in charge of making predictions and updating the page with the values in real-time.

Chapter 5

Results and Discussion

In this section, we present the data generated and acquired by our proposed system. The data presented here was gathered from the sensors during a period of thirty days of continuous working in intervals of ten minutes. This data was collected, then transmitted to the database, and then used as training and testing data for the machine learning algorithm. The collected data consists of 4752 entries from each variable. The validation set was created using 500 entries of the dataset while the the rest was used for the training set. With the trained network we made a forecast for one day in the future for every metric, reaching a maximum accuracy of 96.63% and a minimum accuracy of 88.90%. Figures below show the sensor measurements on the blue line and the predictions made on the red line.

Figure 5.1 shows the measurements of the environmental humidity collected with the DHT11 sensor we talked about previously. Furthermore, the prediction generated by the LSTM network can be seen. In the Figure 5.1, we can see a pattern of rising and fall because, throughout the day, the humidity in the environment varies. In the presence of the sun, there is a drier environment. At the same time, in its absence, higher percentages of moisture are found. In the prediction model, we were able to obtain 96.63% accuracy concerning the real data.

Figure 5.2 shows the measurements of the environmental temperature collected with

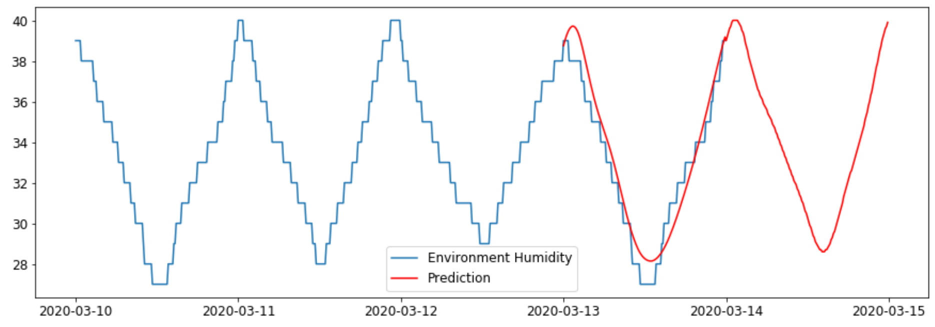


Figure 5.1: Environmental Humidity Measurements and Predictions

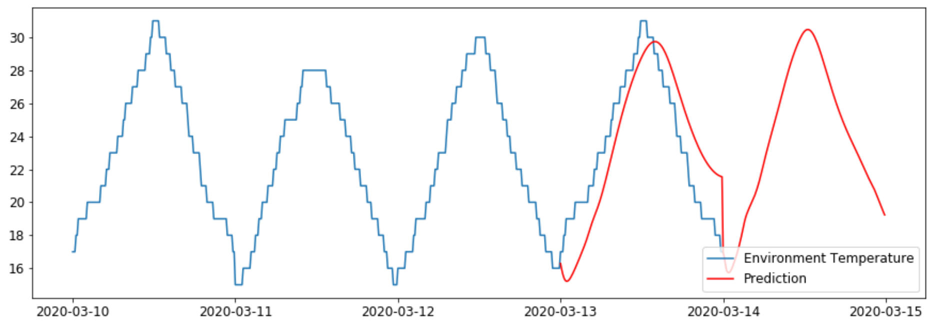


Figure 5.2: Environmental Temperature Measurements and Predictions

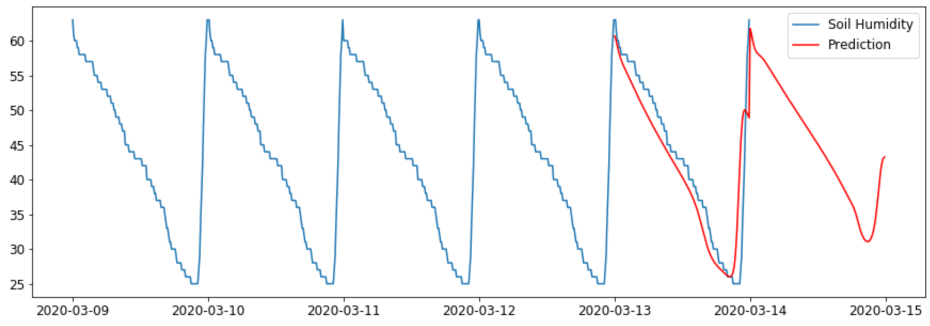


Figure 5.3: Soil Humidity Measurements and Predictions

the DHT11 sensor together with the prediction of the LSTM network. In this graph, we can see the appearance of patterns due to the variation of temperature throughout the day. As expected at noon, the highest temperatures are recorded while in the early morning hours, lower temperatures are recorded. In the prediction model, we were able to obtain 88.90% accuracy concerning the real data.

Figure 5.3 shows the soil moisture measurements collected by the FC-28 sensor together with its prediction. In this graph, we can observe another type of pattern, a rapid rise due to the relatively short irrigation times and, on the other hand, a gradual decrease that means the absorption of the plants and the drainage of the water through the substrate. In the prediction model, we were able to obtain 91.59% accuracy concerning the real data.

Figure 5.4 shows the measurements of the water level in the reservoir obtained with the water sensor and the prediction. This graph is a particular case because although it shows an apparent pattern, it would not be correct to define it as such. The sensor measures the water level in the reservoir, and since it is not a natural variable, it depends on other factors such as the shape and the time of filling the reservoir. The system does not contemplate the filling of the reservoir since this depends on the type of water supply the greenhouse has. Taking into account what was mentioned before, it would not make sense to predict the water level that can be obtained at a specific moment. In the graph, we can

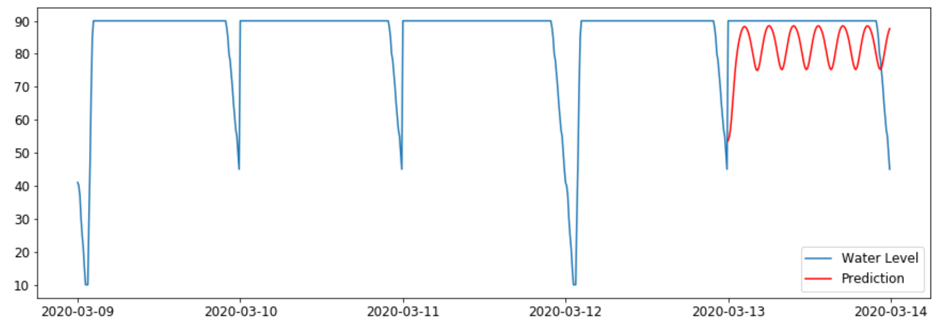


Figure 5.4: Water Lever Measurements and Predictions

observe the erratic behavior that the LSTM network produced.

Before using the NodeMCU module, we use the ESP-01 WiFi module because of its reduced size and price. At first, it worked well, but having only one analog port to receive data, another board had to be used to connect all the devices. As far as the programming is concerned, we also had to make arrangements to be able to send all the data as if they were only one and in the server to separate the data again, complicating, even more, the mechanism. Finally, we decided to use the NodeMCU board because it is a compact board that allows us to have connected several digital sensors, analog sensors, and triggers without significant problems, also allow us to connect to a network wirelessly and through this transmit data over the Internet. The price of the NodeMCU is superior to the ESP-01. However, the difference is too short of making a difference.

Something to keep in mind is the updates of the programs or libraries used. During the implementation, one of the libraries stopped having a function that we used, so we had to reevaluate the code base. A recommendable option is to have a repository with these functions or programs that could be updated, so we prevent possible incompatibilities.

When we graphed the magnitudes and their predictions, we could notice something interesting. Although water consumption is gradual because the plants make continuous use of water, filling it is not, as the system is designed, the farmer is the one who has to fill the reservoir manually, and unless it is always filled at the same time of day the graph does not show periodicity and therefore the predictions are not useful. In the rest of the magnitudes, we can see patterns that repeat over time. This is an operation constraint that could be solve including an automatic control of that part of the process.

Chapter 6

Conclusions

Proper implementation of modern techniques of wireless networks, web development, and neural networks made it possible to implement an autonomous system for crop management through greenhouses. The right choice of the right sensors, actuators, and transmitter nodes is fundamental for the collection and transmission of data, as well as for the activation of control devices.

The sensors and actuators showed efficient operation after more than a month of uninterrupted work. The data transmission device NodeMCU performed adequately. The only drawback is its overheating when exposed to high temperatures such as midday temperatures, which in the long term could influence its lifetime. Fixing the overheating requires the use of heat sinks on the nodes.The use of the Arduino platform to develop the main program of the transmission node is beneficial because it is a widely developed platform with handy libraries for handling sensors and actuators.

The use of the MQTT protocol and an MQTT broker is suitable for data transmission. This protocol is widely recommended for the IoT due to its low bandwidth consumption compared to other transmission protocols such as HTTP. For the implementation of our system, it was beneficial to choose a broker that allows us a simple implementation, fast handling, option to connect a significant number of nodes, an optimal data management and that works according to the needs of our system.

Hiring a virtual private server that goes according to the requirements is critical at the time of accessing the service provided by a system on the web, and our case was no exception. The high speed at which the service can be accessed and the variety of components available is an essential point to take into account and a decisive factor for the proper functioning of the system components. For the development of our system, we employed a server in AWS, and the performance was as expected in these cases.

The implementation of the code and good interaction between the components is something that in this project defines its proper development. One of the things we take care of is not to ”tire” the server, and for that, we use Node.js to recover the data from the database. This execution environment is better than PHP for data handling.

An essential part of the cultivation process is the quality of the irrigation. It is not enough just to water the plants. Crops also need fertilizers, but not just any fertilizer. The needs of the plants differ depending on the variety and the stage of growth. One of the most critical parameters of irrigation is the PH for the absorption of a correct nutrient, in this way a PH sensor could be implemented, as well as one that measures parts per million and electroconductivity in order to have effective control on the quality of nutrients supplied to the crops.

Another significant factor is the lighting. Through the correct management of the illumination, we can obtain bigger plants and bigger harvests. Having a lighting system that regulates itself according to the needs of each plant could make the difference, especially in times of the year when the amount of hours of natural light decreases.

As future work we intend to adapt our LSTM-network-based forecasting to process streaming data from the sensors, this adaptation is particularly important for our architecture in order to work without interruptions. We want to implement this idea while maintaining the precision of the forecasting yielded by the network. Another idea that we want to explore is the integration of real-time weather information with the information provided by the sensors. We believe this could potentially improve the accuracy of our system.

Bibliography

[1] Paolo Varutti, "Evaluation of IoT-platforms and Definition of the IIoT- platform Architecture," 2019.

[2] R. Joshi, P. Didier, J. Jimenez, and T. Carey, "The Industrial Internet of Things Volume G5: Connectivity Framework," p. 129, 2017.

[3] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling Technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[4] X. Shi, X. An, Q. Zhao, H. Liu, L. Xia, X. Sun, and Y. Guo, "State-of-the-art internet of things in protected agriculture," *Sensors (Switzerland)*, vol. 19, no. 8, 2019.

[5] P. Mishra, D. Puthal, M. Tiwary, and S. P. Mohanty, "Software Defined IoT Systems: Properties, State of the Art, and Future Research," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 64–71, 2019.

[6] Z. Jingjing, "Priors for time series forecasting," *Applied Mechanics and Materials*, vol. 263-266, no. PART 1, pp. 171–174, 2013.

[7] N. K. Ahmed, A. F. Atiya, N. El Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, vol. 29, no. 5, pp. 594–621, 2010.

[8] P. G. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[9] IEEE Systems Council and Institute of Electrical and Electronics Engineers, "ISSE 2017 : 2017 IEEE International Symposium on Systems Engineering : Vienna, Austria, October 11-13, 2017 : 2017 symposium proceedings," 2017.

[10] B. Corneliu, G. Oana, and P. Marius, "Considerations on the Development of IoT Systems," *2019 International Conference on Sensing and Instrumentation in IoT Era, ISSI 2019*, pp. 13–19, 2019.

[11] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," *NCM 2009 - 5th International Joint Conference on INC, IMS, and IDC*, pp. 44–51, 2009.

[12] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019.

[13] R. Belsare, "Smart Green House Automation," vol. 5, no. 12, pp. 1127–1129, 2014.

[14] A. Karanjit, "MEAN vs. LAMP Stack," p. 100, 2016.

[15] C. Sevilleja and H. Lloyd, "MEAN Machine A beginner ' s practical guide to the JavaScript stack ." p. 254, 2015.

[16] K. Chodorow and M. Dirolf, *MongoDB: The Definitive Guide.* O'Reilly, 2010.

[17] P. Kozlowski and P. B. Darwin, *Mastering Web Application Development with AngularJS*, 2013.

[18] E. Rosebrock and E. Filson, *Setting Up LAMP: Getting Linux, Apache, MySQL,*, 2004.

[19] M. Gousset, B. Keller, A. Krishnamoorthy, and M. Woodward, "Get more out of," *Management*, 2010.

[20] S. Quincozes, T. Emilio, and J. Kazienko, "MQTT protocol: Fundamentals, tools and future directions," *IEEE Latin America Transactions*, vol. 17, no. 9, pp. 1439–1448, 2019.

[21] M. Queuing and T. Transport, "INTERNATIONAL STANDARD ISO / IEC Information technology — Message Queuing Telemetry Transport (MQTT)," vol. 2016, 2016.

[22] P. Saint-Andre, "Extensible messaging and presence protocol (xmpp): Core," 01 2004.

[23] E. Oriwoh and M. Conrad, "'Things' in the Internet of Things: Towards a Definition," *International Journal of Internet of Things*, vol. 4, no. 1, pp. 1–5, 2015.

[24] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.01.010

[25] H. Derhamy, J. Eliasson, J. Delsing, P. Priller, and A. V. L. L. Gmbh, "Intracavitary chemotherapy (Gliadel®) and oral low-dosevp16.pdf."

[26] S. Lucero, "IoT platforms: enabling the Internet of Things," *IHS Technology*, vol. Whitepaper, no. March, pp. 1–19, 2016.

[27] M. Zdravkovi, M. Trajanović, J. Sarraipa, R. Jardim-gonçalves, and M. Lezoche, "Survey of Internet-of-Things platforms," no. February, 2016.

[28] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Cla, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, 2016.

[29] P. P. Ray, "A survey on Internet of Things architectures," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 3, pp. 291–319, 2018. [Online]. Available: https://doi.org/10.1016/j.jksuci.2016.10.003

[30] J. Guth, U. Breitenbücher, M. Falkenthal, F. Leymann, and L. Reinfurt, "Comparison of iot platform architectures: A field study based on a reference architecture," in *2016 Cloudification of the Internet of Things (CIoT)*. IEEE, Nov 2016, pp. 1–6.

[31] A. Melnikov, "The WebSocket Protocol," *Internet Engineering Task Force (IETF)*, pp. 1–71, 2011.

[32] S. Hochreiter, "Long Short-Term Memory," vol. 1780, pp. 1735–1780, 1997.

[33] J. Castro, P. Achanccaray Diaz, I. Sanches, L. Cue La Rosa, P. Nigri Happ, and R. Feitosa, "Evaluation of recurrent neural networks for crop recognition from multitemporal remote sensing images," 11 2017.