MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

# GREENHOUSE MONITORING AND CONTROL SYSTEM

DIPLOMA THESIS

Author: **Alexandru Florin OLIVA**

Supervisor: **L. Eng. Radu MIRON, PhD**

**2018**

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEAN
**Prof.dr.ing. Liviu MICLEA**

HEAD OF DEPARTMENT
**Prof.dr.ing. Honoriu VĂLEAN**

Graduate: **Alexandru Florin OLIVA**

# Greenhouse Monitoring and Control System

1. **Project proposal:** *Development of an IoT (Internet of Things) application that manipulates actuators and monitors the necessary climate factors of a greenhouse.*

2. **Project contents:** *Introduction, Objectives, Specifications, Bibliographic research, Hardware architecture, Software architecture, Implementation and detailed design, Testing and Validation, Installation guide and user manual, Conclusion*

3. **Place of documentation:** *Technical University of Cluj-Napoca, Automations department*

4. **Consultants:** *L. Eng. Radu Miron, PhD*

5. **Date of issue of the proposal:** November 1st, 2017

6. **Handover date:** July 9th, 2018

Graduate _____

Supervisor _____

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

**Declaraţie pe proprie răspundere privind**

**autenticitatea proiectului de diplomă**

Subsemnatul(a) **Alexandru Florin OLIVA**, legitimat(ă) cu CI seria SB nr. 597554, CNP 1950319324787,

autorul lucrării:

Greenhouse Monitoring and Control System

elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la **Facultatea de Automatică şi Calculatoare**, specializarea **Automatică şi Informatică Aplicată (în limba engleză),** din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea Iulie 2018 a anului universitar 2017-2018, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data

Alexandru Florin OLIVA

(semnătura)

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

# SYNTHESIS

Title:

## Greenhouse Monitoring and Control System

Author: **Alexandru Florin OLIVA**

Supervisor: **L. Eng. Radu MIRON, PhD**

1. Requirements:

Nowadays the most popular fields (telecommunications, security, automotive, healthcare, and gardening) are interconnected through the internet, also called Internet of Things (IoT). It is a revolutionary concept which implies that the majority number of devices can be controlled or monitored through the internet.

In consideration of our busy lives or the cruises we take, we tend to forget about the plants we grow indoors. Considering these kinds of problems, the application wants to improve favorable conditions to monitor and control the resources for indoor growing.

2. Solutions:

In order to maximize the lifespan and growth improvement for the plants, we chose to associate software and hardware architectures manipulated by the internet.

3. Obtained Results:

The main result can be defined by an elaborated application, consisting of two clients and one server, implemented with distinct programming languages (PHP, C++, Html, CSS and JS), frameworks (Laravel, Bootstrap, Highcharts) and hardware components (microcontrollers, sensors modules, relays, actuators and a power supply).

4. Testing and verifications:

The application has been subjected to different testing scenarios by using the features that imply to cover all the monitoring and control for the indoor growth.

5. Personal contributions:

A study on the indoor plants growing, usage of different lights, microcontrollers, different shields, and sensors. Planning a clear purpose for the system application and picking the right software and hardware architecture.

6. Documentation:

Websites, scientific articles, documents, books, and authorized courses.

Author _____

Supervisor _____

# Contents

# 1 Introduction

## 1.1 Project context

Agriculture has been one of the most essential world-wide factors from around 12,000 years ago until today when humans became from hunters and gatherers into real farmers. This domain is a strong aspect of the economic field and population development even if in different countries the rate of growing crops or plants is lower, these have been afflicted by different circumstances. This is a crucial food supply area which implies the input and output ratio of products.

We can accept that by the time we will need to find an optimal solution for the climate changes which are approaching very fast due to the temperature differences.

Winter season is expected to be more changeful in terms of temperature, causing an overabundance of morbidity and mortality into agriculture field in the next seasons. Because of the average temperature, increased in the last years, the expected phenomena are evaporation which will lead to greater amounts of precipitation, resulting in major climate changes into storms associated with winter [1]. In central Europe is expected for an annual average temperature to increase with 3-4 °C. Agriculture will be harmed by the risk of floods, different diseases and soil erosion [2].

One of the solutions would be indoor growing and developing greenhouses. Of course, they exist by now but there are other problems that are encountered daily in these processes. Growing different types of crops or plants is alike growing animals, they need a lot of care which implies the fact that we can't leave them unattended for a long time.

The elements used for houseplants and crops are water, fertilizer, nutrients, and light. But here comes the second problem. What is the amount that is needed for growth in order to avoid dehydration, overwatering or plant wilting? Here comes the power of internet which can monitor and control remotely the necessary resources to maintain the desired productivity even if we are miles away on a trip or we neglected to water them.

Analyzing the data collected by the analog and digital sensors through the internet may help to interpret the needs of growing herbs and achieve growth until the final germination with the aid form actuators (valve and light source). We mentioned above Internet of Things has the power to remote control almost every type of actuators and monitor sensors, facts that will help us to finish the main objective.

## 1.2 Objectives

The application's purpose is to ensure a good quality of growing of indoor plants by having a user-friendly interface, which targets to have users of any kind of age and with basic knowledge about vegetation and technical areas.

The proposed plan of the project consists of two parts, which are: a web application, which is meant to monitor the environmental system for herbs. It has two different type

of users (simple users and administrators). The application will have to store in the database the data collected from the second part of the project and make them visible into two different forms (charts and table). The second part implies physical components such as different sensors, actuators, and microcontrollers interconnected for the greenhouse system.

The first part of the project is based on a web application running on a local server which has the power to monitor and control the physical system remotely, through internet. The users will have limited options, only to visualize the data's in charts and table while the administrators can achieve CRUD (create, read, update, delete) operations even on the user's field. I chose this option for the users in order to share with others the status of the indoor growing system.

The main advantage of the web application is that it can be used on an expanded range of devices (e.g. smartphone, tablets and personal computers) running on most any kind of web browsers.

The second part of the project can't be alternated by users because it consists of hardware elements. The D1 WEMOS Wireless ESP8622 microcontroller will have connected three different types of analogical and digital sensors (air humidity, air temperature, soil moisture and light) which are mapped to show the best accuracy of the collected data's through a wireless connection. The other microcontroller named Node MCU WIRELESS ESP8266 will control the actuating parts such as the water pump and fluorescent light, also through a wireless connection requesting a JSON file which will be later parsed.

Other project objectives would be:

- Minimizing the cost of the physical system.
- Aiming to purchase the best sensors and microcontrollers in the future development and research.
- Improve the accuracy and mapping of the sensors.
- Satisfy the users with the best performances based on the collected data's.
- Publishing the website on a public server hosting service while now it's running on a local one.
- Enhance the system security and trigger an alarm if the system will imply damages (flooding or short circuits).
- Adding an ecological powers supply, such as a solar recharging module battery with photovoltaic panels.

Another future important objective would be to control the flow level and the light intensity with different PID (proportional, integral and derivative) controllers based on the sensor, in view of the readings.

## 1.3 Specifications

The monitoring part is based on data collected by the web application through the wireless shields and ESP8266 microcontrollers that can determine if the environment is appropriate for our plants or not, and to look after them remotely.

The front-end interface is handy as it aims to be used by people of different ages. As it needs to have a higher usability grade and efficiency that is why we used the Bootstrap framework for CSS (Cascading Style Sheets).

One of the main goals is to handle the input data's and to fully control the system without any kind of imminent dangers. That is why we had to be careful when we built the physique stand.

The sensors are connected separately on a D1 WEMOS Wireless ESP8622 and connected to the router. We can set the period of the time to get the expected data's. Because it is based on an internet connection, it comes with a delay of 0.1 milliseconds for a time interval of 5 seconds and a delay of 1 second for a time interval of 1 minute. Increasing the period of the data transmission, the delay would increase as well. This is why on the second part of the hardware architecture we used another microcontroller with two, one channel module relays to control the lights and water supply.

Of course, we needed to implement a security system with a register and login page for the two different kinds of users that we have. Security is a very important aspect as we don't want for someone unauthorized to break the security and damage the software data's or hardware which can be controlled remotely, that is why it was used a complex algorithm based on unique tokens for every user and administrator, an API (Application Programming Interface) which will maintain the session, remember the logged users, have the two middleware for the different routes and have the option to recover the password with a, forgot your password, option.

The main goal is based on combining the first software architecture with the other two on the hardware level, monitoring, and control, to fulfill the plant's requirements hence to have the best efficiency, product viability and to design an automated greenhouse.

Internet of things can also imply disadvantages on software and even hardware architecture. Because it is based on the most revolutionary invention of our times and appeared by the beginning of 1970's, the internet also called data communications which caused a lot of chaos in that period [3]. The IoT technology is built on this type of network, which can limit some users and can go down sometimes for a longer period, causing lack of information and danger of the control access in the moment of manipulating the water pump producing a flood, or turning on the lights producing excess of heat and flux of illuminance.

Because the internet is a system of interconnected computers network [4]and a lot of people has access to it, the web application can meet cyber-attacks if it is hosted on a public domain, having an effect of data damage and uncontrolled hardware situations.

The next issue would be on the level of hardware architecture. The monitoring part implies the sensors which are on a limited budget and won't resist more than a few months plugged in for a long period. But we can choose different components in the future as we have a wide area of electronic sensors.

One difficulty that was encountered was the power supply for the microcontrollers, water pump, and lights. We didn't want to put the user in a case when he or she doesn't know how to supply the electronic components. That is why we chose to supply them all from a common power supply with an input of 220 volts AC (alternating current) which has an output of 12 volts and 1 ampere DC (direct current). Afterward to use some voltage regulators LM317, to have the three different desired voltages. Because we work with water also, we had to mount the electronics on the external part of the stand and to solder on prototype boards all the sensors and wires. In these type of projects short circuits, sometimes are imminent, caused by different reasons, like the reliability of the integrated circuits or the resistances and wires used.

Another concern would be the accuracy of the sensors which for indoor growing are acceptable but using them on an industrial level or big greenhouse's may raise multiple questions of the feasibility for the desired prototype. The reliability of the circuits may depend on two factors: qualitative and quantitative. In the research that has been made, it was determined that it was needed to choose quality sensors to the detriment of quantitative ones.

It may not be enough to use only one set of sensors, or only two microcontrollers as the expected result would be an optimal harvest. On the industrial level, it would be needed more than three sensors that we use in the implemented application. Considering the fertilizer used and chemicals reaction on the soil, the level of nitrogen and the level of carbon dioxide.

# 2 Bibliographic research

## 2.1 Internet of Things

First, we need to a get a quick overview of this fact, Internet of Things, and to understand the concepts that are involved. The Internet of Things is a network of physical embedded systems which can collect and interoperate indoors within the web infrastructure. This concept will mostly achieve a connectivity between billions or alike trillions of gadgets and machines in the next years [5]. This notion has not been known for very long, even communications between machines were provided since the 1800s. The name of this area has been officially named only in 1999. One of the first applications was a Coca-Cola soda machine that was connected to the internet and was informing the students of Carnegie Melon University if the vending machine has available drinks and chill temperature [6].

The architecture of IoT is based on a reliable structure which can contain almost any kind of smart devices. This layer can have sensors, RFID and GPS signals, and it has the authority to collect the processed data on WLAN modular level [7].

As this article [5], informs us there are several features that IoT involves such as:

- Artificial intelligence: algorithms, which include a higher complexity of solving different encountered problems.
- Active engagement: is the interaction between the connected devices, a pattern which is the opposite of the type of connections nowadays.
- Connectivity- nowadays is on a large scale with different providers making it easier to achieve almost any kind of connection.
- Sensors- the main core of IoT, acquiring data from a passive network of devices and remodel it to a real-world integrated system.
- Small devices- the economy shows a readjustment in terms of building futuristic devices, which are smaller, cheaper and can offer more precision and versatility.

### 2.1.1 Advantages

This article [7] shows the advantages of the concept:

- Information: humans are undecided when it comes making a decision that is why having more information lend by the interconnected equipment lead us to conclude actions faster.
- Tracking: It provides progressive information about tracking different products in different areas (healthcare, automotive, smart city, smart manufacturing, home, and gardening automation).
- Economic: if the cost of monitoring devices would decrease, IoT will have a striking upsurge.

### 2.1.2 Disadvantages

As with other technology, there are a couple of disadvantages [8]:

- Compatibility: At the moment a standard format for various kinds of signals doesn't exist (Bluetooth, Infrared or Wireless) and neither for operating systems (Android, Windows, Apple) to have a universal compatibility.
- Cybersecurity: Unwanted potentially malware can penetrate some networks, in order to infect all devices and cause inconveniences.
- Complexity: having complex systems, there are still chances of failure. The infrastructure is still developing, and it will take some time to get to on a stable version of the control. The main areas which need the finest security are public safety and business transactions.
- Data management: another issue in this domain is deciding who can manage different types of data's. It must be implemented a policy of rules for users which can interpret and can offer conclusions.

## 2.2 Big Data

The notion of Big Data comparatively to the Internet of Things is pretty new. The concept appeared in the 2000s according to Doug Laney which defines the three V's [9]. It refers that e-commerce encountered the problem of the amounts of data collected, and the storage to be sustainable. The problem was the volume which besides the e-commerce it was met in industrial, social media and public safety fields with a set of large data's (millions or even billions). Velocity also was a problem since then, but by the time the efficiency has increased a lot. Variety was a barrier in the big data analytics, as the format of the received contextual information was different.

In the future, Big Data and Internet of Things will need to get along with a new infrastructure having a reliable base of hardware and software applications to prepare for a major revenue of influx data's and to deal with. This is the part, where big data analytics can be included to generate a proper tool to develop a good management system for handling and interpret the information.

It is predicted by the date of 2020, nearly 20.8 billion devices will be used worldwide [10] and nearly every human will have associated 5 thousand gigabytes. IoT having its evolution growing very fast, but the main problem will not be the technology produced or developed, but people. Because most of the people will not know how to get conclusions based on the data's, to manage them or how to use them.

The solution for an optimal storage might be cloud computing and cloud storage, which have different privacy level. Which can give us another advantage, a backup.

In the presented situation, the concept of big data may be a boundary between the memory and the information acquired from the sensors. It depends only on the delay we put to collect them. If the delay is lower, the database will fill up faster. That is why we chose a delay of five minutes.

## 2.3 Cloud Computing

Cloud computing is a concept that appeared back in the 1990s. The engineers used it as a metaphor, the cloud, which combined the terms: internet and diagrams. Cloud computing is a lineup of networked machines that can store and process information from our embedded system.

It is a simple concept that has on the bottom-line a major complexity. There are several companies that implemented their own software for this kind of computing applications (e.g.: Google Cloud, Apple Icloud, and Microsoft OneDrive). The user doesn't need to know where the physical location and how the system is configured.

The Internet of Things will have to associate in future with cloud computing in order to achieve the synchronization from machine to machine. But still, there are some distinctions between them. IoT doesn't refer to connect only smart devices like computers, phones or tablets, but different appliances like plants, cars, rooms and even heart rates, whether the cloud computing has some terms of privacy. It can be private, hybrid or public, which people tend not to trust in the most cases.

### 2.3.1 Advantages

Advantages we may take into consideration [11]:

- Synchronization between the interconnected devices.
- The information, documents, and data may be processed online.
- The computational bandwidth speed and memory storage are very high.

### 2.3.2 Disadvantages

This approach has some disadvantages [11]:

- It is established only on a stable and fast connection of internet.
- Security may be endangered.

## 2.4 Microcontroller Embedded Systems

When it comes to developing an embedded system, we need to take into account how the project will be developed and what base we will choose for our hardware architecture. That is why in the project proposed we used two different microcontrollers that will do the required tasks, monitoring and control.

The first microcontroller appeared between 1970 and 1971 and was invented by Gary Boone an engineer, who worked at Texas instruments [12]. An approximate of 50% of the total worldwide sales of microprocessors are because of these boards.

Nowadays there is a big variety of MCU's from where we can choose. Most of them contain microprocessors, different types of memory (e.g.: ROM, RAM, and non-volatile types like Flash and EEPROM) operational amplifiers, and dedicated modules. In our case WIFI modules named ESP8266, which are integrated on the motherboard and have the power to communicate through TCP/IP protocols.

The architecture of embedded processors has a high practicability in our days and even if we don't know we can find a minimum of ten microcontrollers in each house, but in smaller versions and used in household appliances.

The lower cost may be the best advantage in this domain, also the power dispersion and the integration in different applications with all the components.

On the other hand, it has limited performances, can't interface higher power consumers directly and can perform only a limited number of instructions.

The boards used by us appeared only in 2013, year which was a revolution for a wide area of application with the internet.

## 2.5 HTTP Requests

Hypertext Transfer Protocol, also known as the acronym HTTP, is the most used and common method to gain access to information from the World Wide Web (WWW). These protocols are under the form of text and are interpreted by the destination machine. Using them means achieving the communication between microcontrollers and server using GET requests. We used this type of request because we want to read the information from the resource without changing it.

On the management part, we have RESTful Web services which have the role to handle all the data with the POST, PUT, and GET methods.

## 2.6 State of the art

### 2.6.1 Internet of Things in high-level applications

To achieve implementation of an IoT system dedicated to greenhouses it was needed to achieve knowledge about the climate factors, which we will need to monitor. In this article [13] the author explains that we need to manage the aspects that we already mentioned above and also the gases (Carbon Dioxide (CO2) and Oxygen (O2)) that have an impact on the environment. It will be an additional feature to add these kinds of sensors and a PH sensor for the soil, but in the case presented it doesn't apply as we focus on only one single plant.

In the agriculture field, many hobbyists and farmers choose to use more than one sensors and to put them in different depths of the soil. Also optimizing the irrigation in the open field would be a futuristic option, to prevent evaporation. That is what Katsoualas and Bartzanas presented in 2017, an online system for a scheduled irrigation [14].

In figure 2.6.1 is exampled an overview of a modern example with an IoT solution for a greenhouse. The solution of the outside facilities is uploaded to a cloud and later on interpreted on any kind of smart device.

In the next years, the main intention of IoT globalization is to supervise, not only limited spaces like buildings but animals, humans, and tracking of the food supply. By

owning different drones with artificial intelligence algorithms implemented and having various tasks of monitoring and reporting to the custodian.

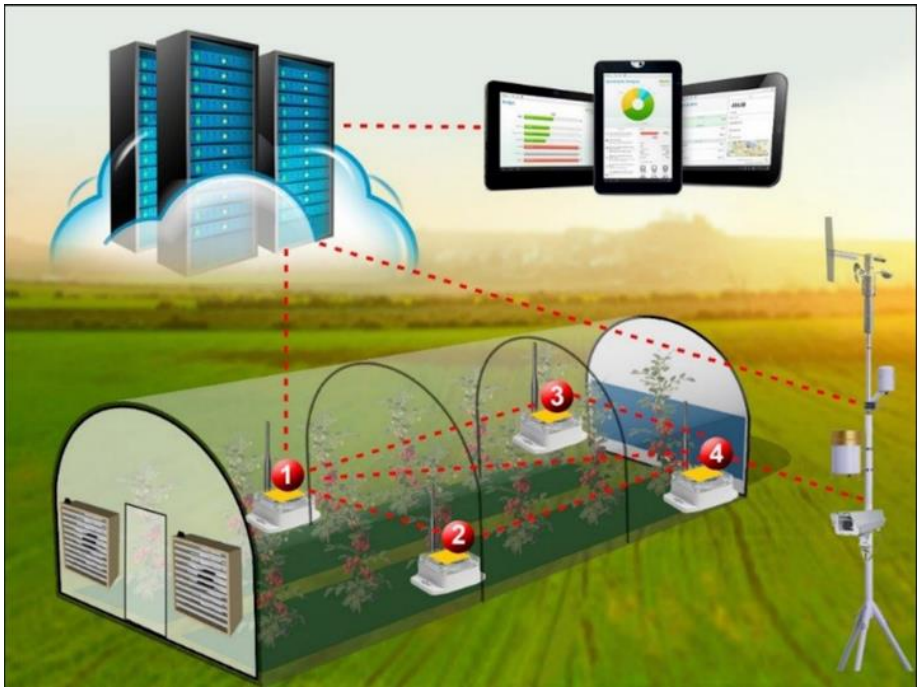


*Figure 2.6 Integration of Internet of Things into agriculture*

### 2.6.2 Industry 4.0

The fourth model of industrial revolution is based on the information from the physic layer of a system with characteristics of statements, intelligence, and personalization illustrated in the manufacturing, automotive and safety field.

Internet of things is one of the concepts we hear the most, associated with Industry 4.0. The use of these intelligent objects is allowing a real revolution within companies that are now always connected and exploit data to optimize their production processes.

The workforce global orientation will change and will demand more skills for the workers. That will enforce the next generations and even the old ones to focus on professional transformations and to overwork. Also, by having this kind of development, the companies will have to prepare for optimal education and trainings for the human society. Seeking interpretation, flexibility, and accommodation of working with different machines. Once again Internet of Things has an important part in development, in different industries.

# 3 Hardware Architecture

The hardware architecture is divided into two major categories: climate tracking and resource control. It has at the base layer two different microcontrollers that have distinctive purposes and are supplied by a common power source to make the assembly easier for users. The two microcontrollers are the clients that communicate with the server through a PHP script and a JSON file.

It is divided into two parts because it isn't supposed to interfere with the signals of the sensors with the ones from the input commands. It could be an improvement to use a master-slave option in the future development by implementing an automated watering and lighting system with the received commands from the master board and adjust the actuators to have an ideal feedback. That is an applied case only if the hygrometer and light sensors modules return very precise collected values.

Before starting the assembly of the physical part for the monitoring system, it was needed a thorough investigation and research for choosing the best sensors with a good ratio between budget and quality. Also choosing the microcontrollers boards was an extra challenge, because of the transfer rate and speed of connectivity.

For the control system, it was needed an analysis of the type of lights that should be used on the desired plants, of course, it varies from one species to another. The second actuator is a small submersible water pump which has the necessary flow for the appliance on this kind of scale.

In the end, the chosen development boards were: WeMos-D1R2 board and Node MCU. Both of them being important open source devices that communicate over TCP/IP protocols through the ESP8266 WI-FI module tuned in, which in turn has incorporated a 32 bits microcontroller and have an internal contact with the server through a wireless connection.

## 3.1 Client monitoring system

The tracking system monitors the environmental factors which influence the plant's behavior. It is conceived by three main sensors, but before purchasing them it was needed a test case and matching view of point. That is why it was used as an open source designing software for electronic parts name Fritzing. Which has the option to test and manipulate core parts in any way it is desired. From designing on a simple breadboard, PCB, schematic and even simulation of testing codes on fictive platforms. Concluding to conceive an overall picture before building.

After all the essential information from testing and using different devices, was acquired about the necessary parts, the arrived point was to buy the electronic components for the designed schema in figure 3.1, which is the last prototype for the demonstration.

Two digital sensors that will monitor the value of air humidity, air temperature, and the value environmental light. An analog soil moisture module that can also work on the digital pins but can't define a precise state of the ground, only true or false values.

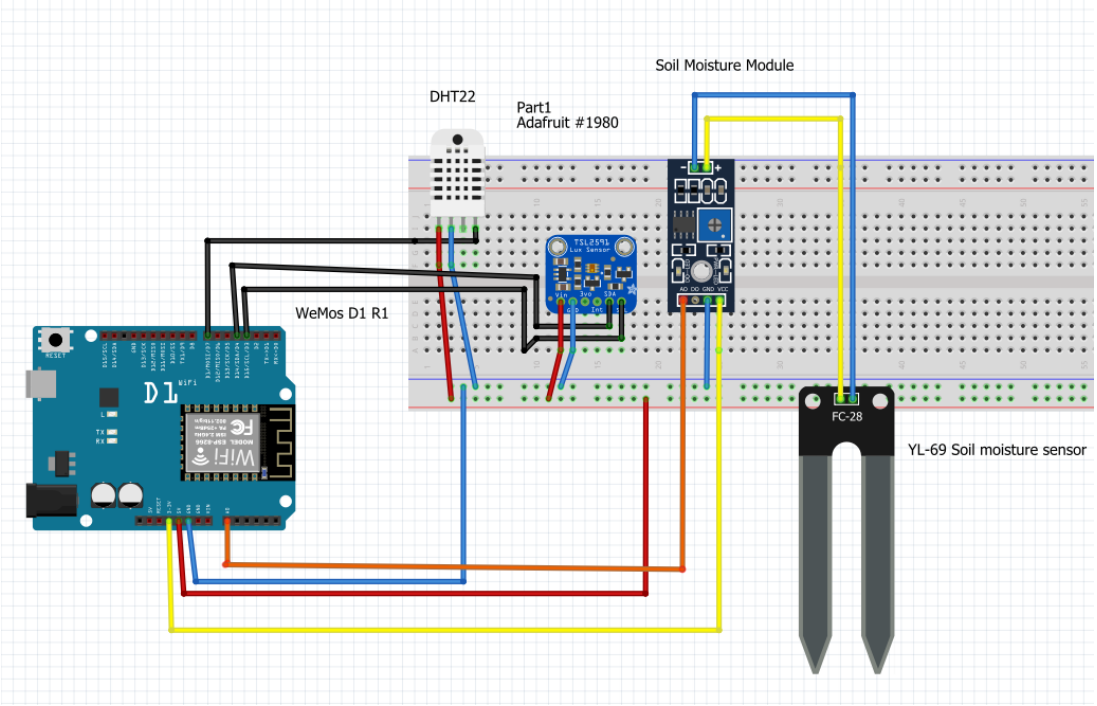


*Figure 3.1 Design of the monitoring system*

### 3.1.1 ESP8266WEMOS-D1R2

Right off the bat, for the monitoring part and first client, was tried the Arduino Uno microcontroller board with an ATmega328p incorporated processor which has a memory of 32 kb and 14 digital input/output pins. Stacked with an Ethernet shield, the board worked at its full limit. It was discovered that the utilized gadget didn't have the essential transfer rate and lacks due to establishing a connectivity between the server and the shield. The speed was around to 16 kilobytes/s, not 10 megabytes/s specified in the specification datasheet, the fact that will make impossible to expand the postponement of the information committed to the server. Besides, it required a local-area-network connection through a cable connected to an internet router, entangling the architecture a lot. The only advantage, in this case, would be an SD card slot that the shield has incorporated and can give a base reinforcement to our information gathered.

The monitoring job was not a matter of problem, but rather when it came to control the shield was an aggregate disappointment. By just attempting to power up a led remotely, the delay of the input command was almost five seconds, causing a noteworthy issue in the process. That is why another board was used, which will be explained in the next sections.

IP strife can occur in a network when two communication protocols have the same static address assigned into the DHCP range. This was another issue when it was endeavored to establish a bridge connection between the board and server.

In conclusion, the tested Ethernet shield didn't fulfill our requirements, turning out that it is not flexible, volatile and neither efficient.

The board used at the moment, ESP8266WEMOS-D1R2, which is presented in figure 3.1, is a WIFI based standalone with an ESP8266 microchip incorporated that can work either with TCP or UDP protocols. It is built on the same footprint as Arduino Uno but with the possibility of a wireless connection, the ideal type of microcontroller for Internet of Things projects.

It is cheaper and faster than the Ethernet shield and doesn't require a local-area network cable making it more user-friendly. The only disadvantage is that it has only one analog general-purpose input pin.

The integrated development environment in which the source code was developed for the monitoring system is Arduino IDE which is an open-source software, thusly it has at the base a Java code machine that converts the script for the interpreter.

It is using C++ language which is a subset of the C language library for embedded systems. But it is not like the usual language for implementing different algorithms as it is working with different electronic components, pulse width modulation, different types of communications (serial, Bluetooth, tcp/ip), inputs, and outputs. It has distinctive particular instructions and control structures for this kind of sketches.

This Ide is capable of working with multiple boards even with the ESP8266WEMOS-D1R2 and Node Mcu. With this option, the required sketches are implemented with the help of this software. As any code, it has a set of libraries for the different sensors (DHT, Adafruit TSL 2591) and modules (ESP8266WIFI, SPI, Wire) implemented with the microcontroller.

Void setup is a default method used in every sketch, where one important instruction for testing is declared, Serial begin (baud rate value). The baud rate esteem is the rate of bits transferred in every second for serial information transmission. Actually, that will help to test in the serial monitor, the received data from the sensors and wireless module. It can have different values, but we chose 9600 baud/second because is optimal and has an error of only 0.2 percentage. This method is running only once because it's starting all the threads necessary for the serial monitor, temperature sensor, and wireless connection. It can be reset only from the button on the board.

The void loop is the second default instruction used in every implementation. As its name is saying, it runs the script over and over again achieving manipulate the system and enabling the program to change and react.

At the top of the sketch, it was needed to declare a WiFiClient client object to accomplish our request to the server with the collected information. The client object prints the values of the declared variables into a GET request and passes by to the server,

which has a PHP script that stores all the received data into the database with three tables (light, moisture, and soil).

To find out what IP should be assigned to the board, it was used the command ipconfig in the command prompt. With the essential information printed on the terminal, we nominated the board's IP value with one higher than the IPv4 Address which is the one that will be used for the constant of type char server.

The communication between the client and the server is taking place through the port with the value 80.

Another three variables used are: ssid, password, and mac which will ensure the necessary credentials to establish a connection between the server and client. The mac address is alike the IP of the client and should be unique in the entire network.

Toward the end of the sketch it is the instruction delay (ms) which can be set at any desired time to interrupt the transmission the data collected from the client to the server. Ms is the value in milliseconds. The Ethernet shield passed only two test cases: any time of delay that was under one minute. This is one of the reasons why the shield was supplanted with a remote one, and in addition, the time to connect to the server is lower.

### 3.1.2 Sensors

#### *3.1.2.1 Air temperature and humidity*

The first sensor used and tested for monitoring the air temperature and humidity was DHT11 which is an ultra-low-cost and slow sensor. After, being replaced by another digital sensor named DHT22 which has a higher price and better accuracy.

The first sensor has a temperature interval between 0 and 50 degrees Celsius for temperature and a range between 20 and 80 percentage of relative humidity. The only advantage besides DHT22 is that it has a frequency of transfer rate of 1 Hertz making it faster than the other which works on 0.5 Hertz, meaning it can have one data collected on every two seconds.

The relative humidity is the physical dimension of the report between the quantities of vapors that are present in the air at a certain temperature. The formula for this proportion is presented in equation 3.1. According to [15] the actual vapor density is the gravitational mass over the cubic meter, same for the saturation vapor density. These two dimensions are indirectly proportional with the temperature. In case the temperature will rise, the capacity of air will fall and hold less moisture, until at a certain point where the water will condense if the temperature hits a dew point [15].

$$\text{Relative Humidity} = \frac{\text{actual vapor density}}{\text{saturation vapor density}} * 100\% \qquad (3.1.2.1)$$

For indoor growing, some plants need a percentage of more than 80% if they are found in misty forests, but most of them need only over 50%. The winter season rises up

problems for indoor farmers as the relative humidity falls below 20% and is needed to find ways to increase it.

The hygrometer is composed by two parts: a capacitive humidity sensor and the other one is a thermistor, encapsulated into a plastic shell.

The second measured physical quantity is the temperature, using the scale of Celsius. Being also an important factor for indoor growing, it should be between the interval of 21.1 and 26.6 degrees.

The first tests for DHT 11, showed that only the relative humidity is changing due to climate changes, but the temperature stayed the same, which made difficult to accomplish the primary objective, an accurate framework for monitoring the herbs. Be that as it may, with the DHT22 the things were somewhat unique, demonstrating a superior exactness for the readings, but also the disadvantage of a higher cost and sometimes a wrong temperature and humidity value, because of the voltage drops.

The factors that can affect the future readings of the sensor causing outlier data of temperature and humidity readings are voltage and current fluctuations.

For the implementation of these sensors, it was used a library, the DHT.h, which is a repository dedicated especially for the DHT family sensors and other unified sensors by Adafruit industries.

In conclusion, the chosen sensor does the needed job, but it may be replaced with a more expensive one in the future.

#### *3.1.2.2 Illuminance*

Light is one of the most important assets for indoor growing. As it is used a fluorescent artificial light, it needs to be estimated with a well-designed sensor.

Alike for the air temperature and humidity it was used a light sensor, LDR (light dependent resistor) that is supplied on input voltage with 5 volts and works either with analog and digital inputs. This is an ease sensor and can gauge the energy of the light. In this case, the sensor didn't read a precise value and changed only at extreme points of light energy. It is composed of an indicator light to notify if it works or not, a sensitive adjustment module, 3 pins, and a photoresist.

After a series of tests, the verdict was a clear one. The sensor couldn't read correct variables and neither to be mapped in a way to integrate it the monitoring system. The data collected were simple analog or digital signals that couldn't tell a precise level of illuminance in lux.

In the end, it was chosen another sensor, an expensive but very sensitive and accurate one called TSL2591 from the Adafruit, that converts light to a digital signal output in lux with a wide range between 188 micro Lux and 88,000 Lux [16].

TSL2591 has five output pins and two diodes (spectrum and infrared) [16] :

- Vin- which is used to power the module with a voltage between 3-5 volts direct current

- Gnd- the ground which is used between all the other sensors, sharing a common logic and voltage level
- 3vo- the output pin, that supplies a voltage of 3.3 volts direct current, from where we can pull a current intensity of near 100 milliamps
- SCL- clock line
- SDA-data line

The library used in the sketch for this sensor is Adafruit_Sensor.h which is offered by the distributor and comes with three different methods to measure the level of illuminance.

It was picked a medium gain that is multiplied by 25, an integration of 300 milliseconds and an advanced method to read the lux values.

In this case, the illuminance is measured in lux, which is the measure of the amount of light spreading over a particular zone or encountered by different objects, in our case, the plants. More exactly the luminous flux is composed by lumens (lm), radiated in a strong edge of one square radian by a uniform point source having a power of one candela. Projected on one square meter, the ratio between the lumen and the area results the international measurement unit named lux or level of illumination.

After a time of research, the chosen source of illuminance for our application is a fluorescent tube light bulb, that has a power of 8 watts, 400 lumens. And after calculation of the lux value with these two formulas: (3.1.2.2) and (3.1.2.3) [17], it was concluded that this type of light can provide a result of 4000 lx, which is adequate for the plant's environment.

$$\Phi_{v(\mathrm{lm})} = P_{(\mathrm{W})} \times \eta_{(\mathrm{lm/W})} \qquad (3.1.2.2)$$

$$E_{v(\mathrm{lx})} = P_{(\mathrm{W})} \times \eta_{(\mathrm{lm/W})} \,/\, A_{(\mathrm{m}2)} \qquad (3.1.2.3)$$

In the first formula, (3.1.2.2) it is needed to be calculated the value of lumens by multiplying the power of the light source, in our case 8 watts with the luminous efficacy, which is the ratio between the generated lumens and the power of the source, having a total of 400 lumens.

In the second formula it is calculated the value of lux, which will be 4000 by dividing the first formula to the area section, which is 0.1 square meter. By having these two formulas is demonstrated the ratio between the lumen and area.

An aspect that can cause very large data variations is the sudden change of the light flux by only exposing the sensor to a very bright light source or by obstructing the sensor with different objects.

In conclusion, the optimal sensor is one from the Adafruit industries, having a good precision of reading illuminance with multiple choices of high sensitivity methods. The generated number of lux is 4000 and can ensure an ideal case of lighting and grow an indoor plant.

#### *3.1.2.3 Soil Moisture*

Soil moisture is another important factor for the monitoring application as it needs to inform the user when to water the plants. That is why a hygrometer sensor is used, which can monitor either analog or digital, the state of the soil.

The YL-69 sensor and HC-38 module is a low cost but efficient sensor that has a medium accuracy of reading the values. It is operating at a direct current with a voltage between 3.3 volts and 5 volts. In this case, it was used the 3.3 voltage.

It is composed by a module that has four pins, an operational amplifier LM293, which in fact is a comparator, and a potentiometer that can be adjusted for the desired sensitivity. Like the other sensors, it has a pin for the voltage input, one for the ground, one for analog logic and the last one for digital. The digital one, can measure if the moisture of the soil has a high level or very low, printing Boolean values. That is why for the implementation of the monitoring system it was used with the analog signal, which may vary from 0 to 4.2 volts.

In the end, having the option to map the value read from the sensor and putting a range of constrain. The interval of reading analog input data is from 400 to 1023, then mapping it with an instruction and deciding that the 100% value of the soil moisture will be 1023 analogic signal, respectively the 0% will be 400.

The percentage data collected, is not a physical dimension or neither an international unit measurement, it is just an orientation value with the purpose to inform the user when the soil is too dry or too smooth.

A factor that can influence the readings of this sensor is the depth of the pads introduced in the soil. If they are introduced just a little, the readings will be wrong leading to confusion. That is why the pads must be introduced in its entirety.

In conclusion, it wasn't necessary a comparison between this one and other sensors as this one fulfills the needed requirement of informing the client with the necessary state of the soil.

A future improvement would be to buy a more expensive sensor. The main impediment is that, on the market, other hygrometers are excessively costly and would cost more than the entire monitoring and control equipment architecture, concluding that there isn't a middleware option of a sensor with a ratio between price and quality.

#### *3.1.2.4 Alternative components*

Overall, after the comparisons and tests of the sensors, an alternative would be a weather shield that has integrated three sensors: a barometric pressure, humidity/temperature and light sensor. But in this case, it wouldn't be necessary the pressure one, needing a soil moisture sensor.

Another feature that could be implemented would be to add an extra sensor to read the water level in the tank, and then to inform the users about the absence of the water.

The main disadvantage of this shield is that it isn't reconcilable with ESP8266WEMOS-D1R2, only with the Arduino Uno which has the only option of an Ethernet shield.

Another disadvantage is that if the shield suffers an overvoltage, will lead to a broken board and will damage all the sensors.

### 3.1.3 Monitoring system deployment architecture

After putting all the components together, we have the deployment diagram in figure 3.1.2.

The figure presents the system as a whole, namely how the process takes place and how the connections are established by creating nodes between the physical parts and the web. The monitoring system is the first client that is connected to the server, in any case, it can be observed that the prototype is sending data's through the internet and then stored into a database that is hosted on the local server. Both connections are based on HTTP requests creating a bundled environment for the web application and microcontroller.

On the local server are hosted the website application and the database, which can be used also from other smart devices that support different web browsers

At the end of the sequence, we have the users that will interpret the given data and will make decisions, accordingly to them.

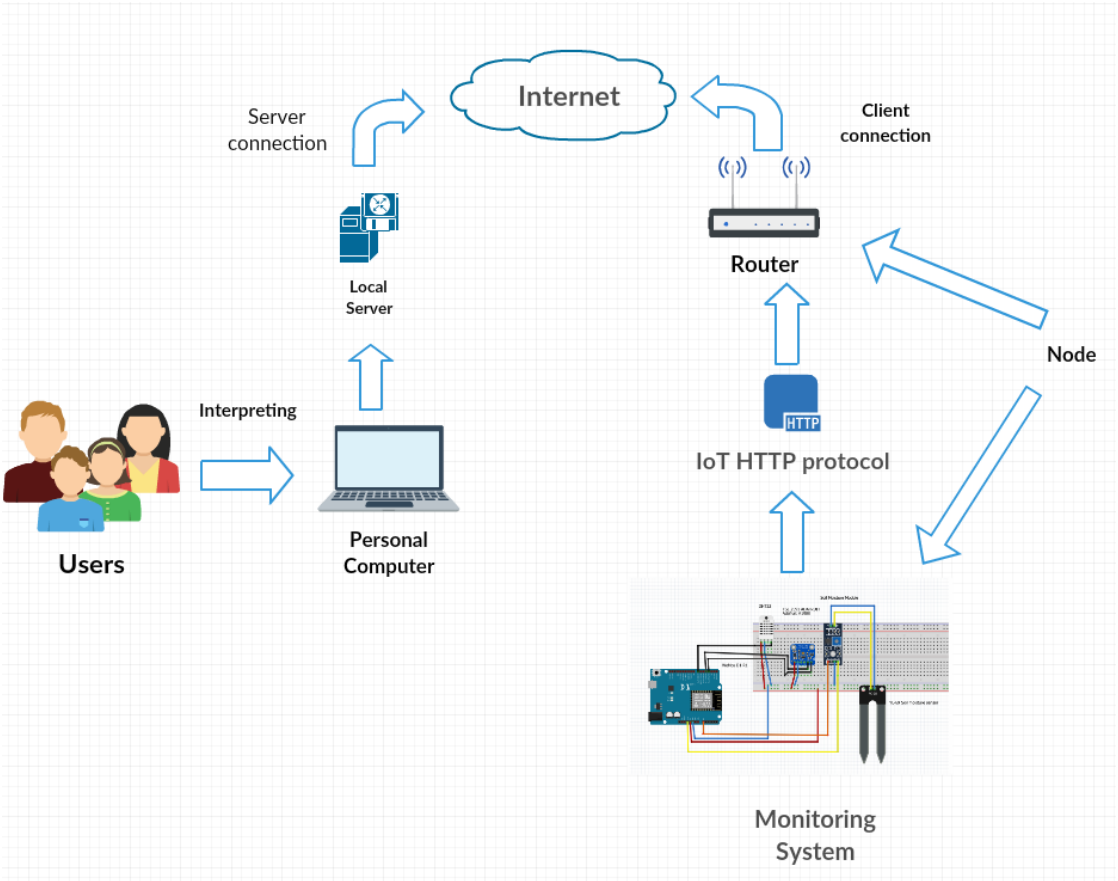


*Figure 3.1.2 Hardware deployment diagram for the monitoring client*

## 3.2 Client control system

The second client of the application is the control system which can be analyzed in figure 3.2.1. It is composed by the NodeMCU ESP8266, two relays with one module channel, a submersible water pump, a fluorescent light source, two power supplies, and DC-DC step down regulators.

The light source, water pump, and the microcontrollers share the same power supply that converts the 220 volts alternative current into 12 volts direct current, then is divided into three parts and converted with DC-DC step-down regulators for the desired voltage input of every current consumer.

### 3.2.1 NodeMCU ESP8266

By attempting to develop the monitoring client system, it wasn't required any furthermore investigation in which kind of microcontrollers to use, that is why NodeMcu ESP8266 is the arrangement for this situation. Compared to the other board, this internet development board doesn't have a voltage pin of five volts, making impossible the necessary voltage supply for the other sensors, therefore the ESP826WEMOS-D1R2 is used for the monitoring client system.

It is composed of ten GPIO pins that can provide power width modulation signal, one of them being the analog one. The ESP8266 microcontroller is the base for this development kit and can interpret C++ and LUA languages.

The chosen language is also C++ in this case, but the problem was addressed differently.

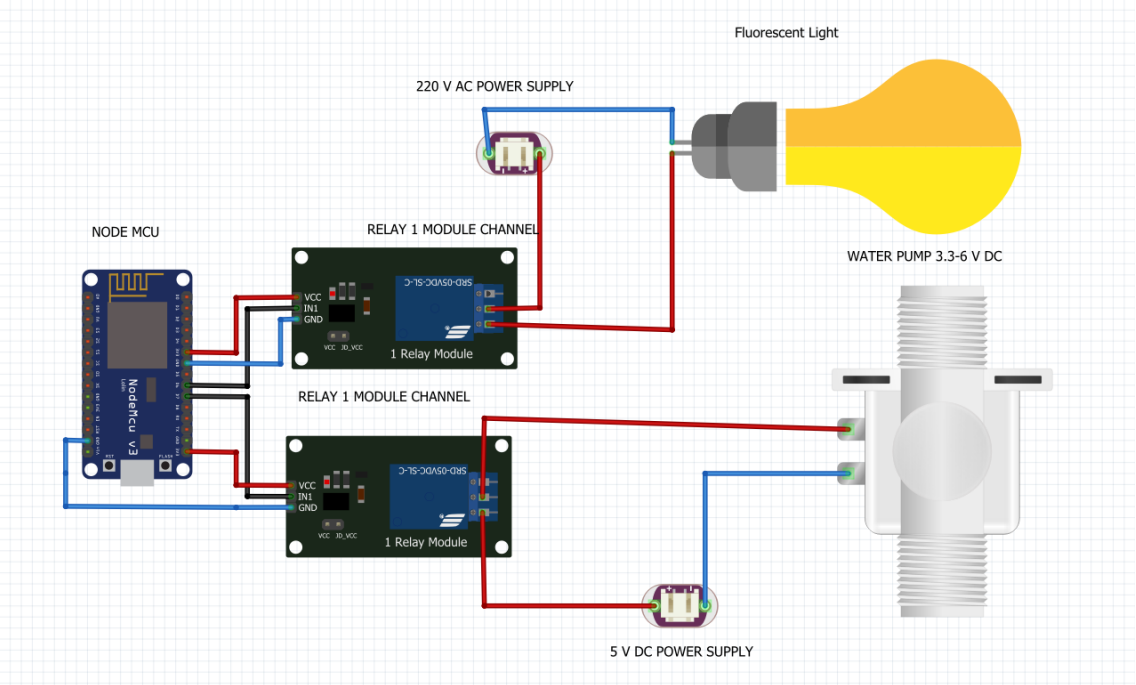


*Figure 3.2.1 Design of the control system*

The problem was addressed differently because the implemented sketch is reading a JSON type format text file. Based on this file the microcontroller will trigger or halt the relays that are connected to the microcontroller. It can be said that any electronic equipment can be manipulated remotely by microcontrollers and internet.

In the heading of the script, uploaded on the microcontroller, are the necessary libraries for the wireless client and JSON encryption.

Alike on the ESP8266WEMOS-D1R2, it was used the same hosting address and password, and port number value 80, as it is needed to establish the same connection to the server where the web application is hosted.

Declaration of the pins are also included in the header, being only two of them, it is used the pinMode command. They are declared with the value HIGH, which means that the pin will receive an input voltage, but in this case, the logic is reversed because of the relays.

It is used an HTTP GET request for parsing a JSON file that is edited from the Control website page. This website can be accessed only by the administrators of the page. After the JSON file is parsed the result will be converted into an array of characters.

The main implementation is when the array is compared with the value from the if statements, using the strcmp command. Being also a reverse logic because of the relays.

The logic is made for the resources providers to work alternatively, and not at the same time to avoid an overflood for our plants and to damage the equipment.

It is used this logic because if the lights are turned on, the water pump will automatically turn off.

Another improvement would be to add an extra sensor near the pot of the plant, that will trigger an alarm if reads any presence of extra humidity. In this way, the used sensor will provide the necessary information that the water level has exceeded the limits.

### 3.2.2 Relay

The module relays used are acting on only one channel and supplied with the necessary voltage of 3.3 volts provided by the microcontroller. The role of these electronic components is to trigger the on or off statement with a small input voltage for electrical devices that are using a higher alternative or direct current (e.g.: the fluorescent light which works with a voltage between 220 and 240 AC), whether the transistors can’t work with these types of inputs or outputs.

An advantage of the relays is that it prevents controlled devices to be in direct contact with the microcontroller, avoiding short circuits or damage from over voltages.

They are used in almost any industry of electronic and even for traffic lights. The usage of relays has become a very important factor for automotive industry also as they behave like fuses, being one of the many advantages. Other advantages would be that they are reliable, flexible, and can work with a direct and alternative current.

One of the greatest impediment is that it cannot switch quickly the statements of the devices, having a delay of a couple of seconds.

### 3.2.3 Actuators

#### *3.2.3.1 Water pump*

For the water supply, it was used a submersible water pump that has a range of voltage input between 3 and 6 volts, and a flow rate of an interval between 80 and 120 liters per hour.

It has a fairly small size, but it suits perfectly into household applications, making it ideal for the control system.

Because the actuator works as a DC motor, the flow rate is directly dependent to the voltage input, that is why an optimal value of 3.3 volts was chosen as the power supply to be ensured that the water level will not exceed when the users try to activate it.

The main advantages of the submersible pumps are that they can be submerged into the water tanks, and don't need additional configurations. The only additional part that was needed to fulfill the watering control system was a hose, through which the water would pass. Another advantage would be the way it pumps the water, as it doesn't need a lot of pressure to ensure that the fluid will cover more distance.

The hindrance of the system is the short wire linked to the pump, being manufactured in that manner from the provider, it will be needed to use a low height of the water tank.

Since is working with water, breaking the wire could lead to short circuits leading to a damaged water pump. Another fissure could happen for the hover, being a bigger issue for the other electrical components that can be messed up because of water leaks.

Another alternative would be a 12 volts water pump that would need a supply input and output hose, involving a more complicated hardware architecture. Being such a powerful water pump, the output flow rate would be inappropriate for this kind of application

#### *3.2.3.2 Compact fluorescent lamp*

For indoor growing is not important from where the water is supplied or what it has in its component, as it is on how we distribute it. On the other hand, the facts aren't the same for the artificial source of illumination.

It is important to know what kind of light source the indoor plants demand and how they react to it.

There exist four electrical artificial light sources: incandescent, compact fluorescent lamp, high-pressure sodium and led.

The incandescent light bulbs are the least efficient for indoor growth due to the propagation of the wavelengths and the lifespan. But when it comes to the costs, led bulbs are the most expensive and with a lifespan 21 times more than incandescent types, and 3 times more than the fluorescent ones.

After taking in consideration the ratio between efficiency, costs and lifetime of the different bulbs, it was taken a middle way of choosing a compact fluorescent lamp which

was for the first time fabricated in the 1890s, and has a power of 8 watts, 400 lumens, and illuminance flux of 4000 lux.

Alike the water pump, the light bulb will also be commanded by a relay which will be in turn commanded from the server web application giving commands to the microcontroller.

### 3.2.4 Alternative improvements

Another way to achieve the functionality of the control system would be to add a self-care autonomous option. In this type of feature, the controller acts on the data collected from the sensors and makea decision after the interpretations. But this isn't a safe option as it needs an extra caution from users, and sometimes the sensor can send wrong values caused by current dumps or overvoltage.

Some other components that could imply the replacement of the actual actuators would be a band of LEDs surrounding the stand, offering a better flux of illumination and water pump that can be submersed deeper into a water tank.

Lastly, the best feature to add for this kind of application would be a solar power supply to reassure an improvement for the ecological power and safety. Working it out at a bigger scale on building this kind of greenhouses can have a major impact for the ecosystem.

It was taken into consideration the fact that if the internet connection is lost in a moment of a command input, to stop any relay. With this condition, it can be assured that it wouldn't be caused any damages.

### 3.2.5 Control system deployment architecture

For the control system deployment architecture, things are a little different as the only individual that is empowered to make decisions is the administrator. The second microcontroller client is receiving through a HTTP GET request the commands for the actuators. And even for this type of connection between the client and internet, is created a connection node.

In figure 3.2.5 is highlighted the fact that shows how easy is to manipulate the actuators through the internet and the association between the physical parts, like the wireless modules and the server.

Interfacing the internet with the web browsers commands, and microcontrollers can display how easily it can be extended this wide area of controlling remotely any kind of devices, from home appliances to medical equipment and cars. Generating an additional resource of help for the careless people to the disabled ones.

The JSON text file makes possible the interpretation of the input command, parsed by the NodeMCU microcontroller, passing on the command to the relays that empower the actuators to activate or deactivate.

At the end of this sequence is the plant that demands the resources necessary for a good maintenance. Both the monitoring and control system are depending on the climate factors and environment for indoor growing improving the quality and lifespan of plants.

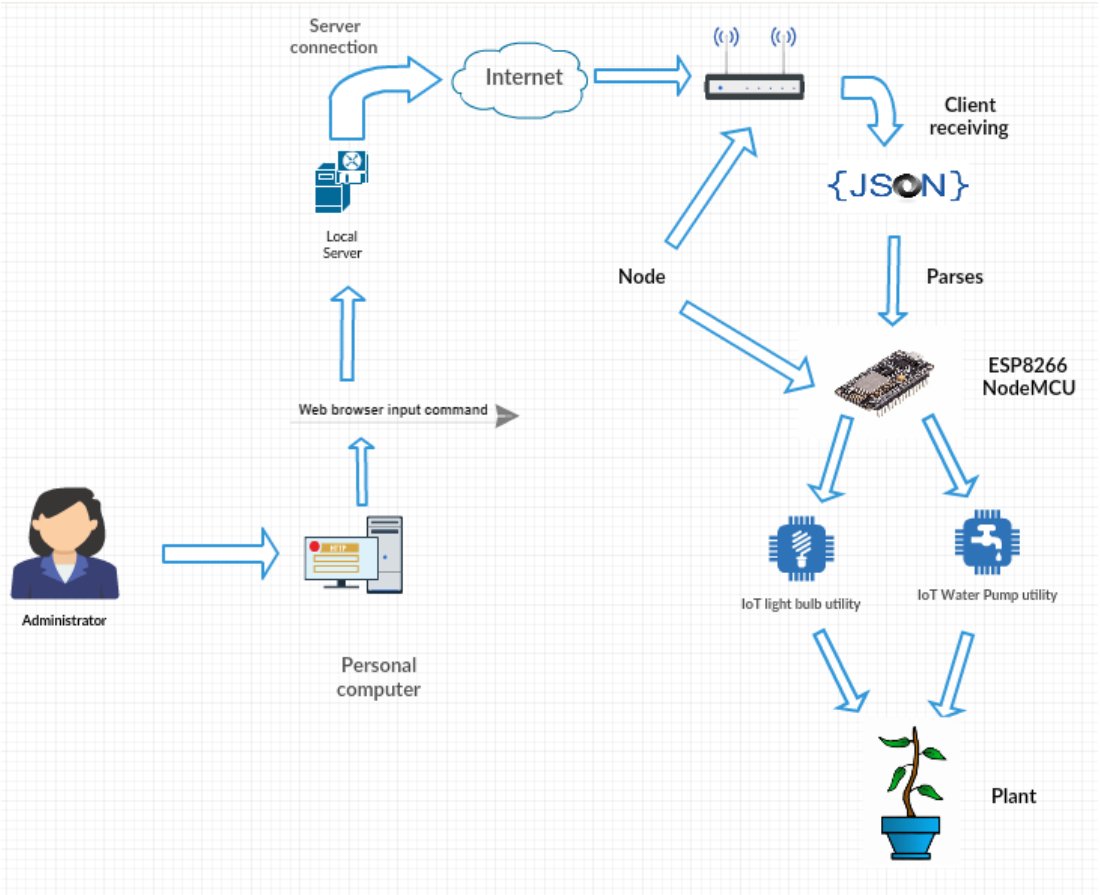


*Figure 3.2.5 Hardware deployment diagram for the control client*

### 3.2.6 Assembling and implementation of the physical framework

Finally, after the control and monitoring system were implemented, they needed to be assembled together to create the application in its final form.

That is why it was needed to take in consideration a lot of variables, improvements, and disadvantages.

The stand is composed by a wooden framework which holds all the electronic parts in the exterior. Arranged in this way to be free from any kind of accidents produced by the water pump or short circuits. Disposed into electrical capsulated boxes, which are meant to protect against different factors and to not let the users to interact with them.

The water pump and the microcontrollers have the same power supply which converts from 220 volts alternative current into 12 volts direct current.

The input supply voltage is divided with a perfboard that has one input and three outputs soldered, composed by terminal screwed connectors which assure that none of the wires will unwrap. Also, all the sensors are soldered into a separated perfboard.

The stand is mounted in such way as to confer users the option just to plug the system into an electrical socket.

In figure 3.2.6.  can be observed the stand in its final form.

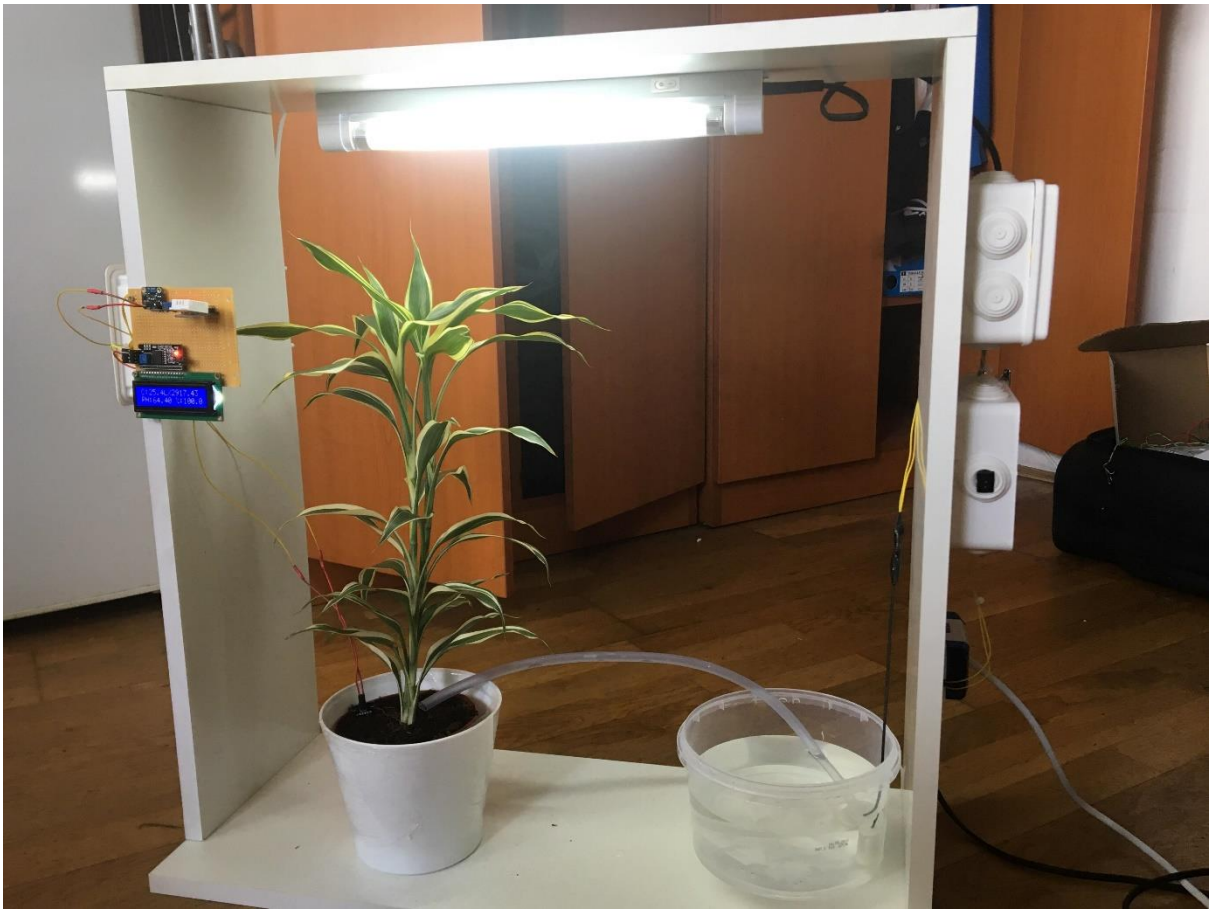

*Figure 3.2.5 Assembled physical framework*

# 4 Software architecture

## 4.1 Model View Controller Design Pattern

The Model View Controller design pattern also known as MVC, is a concept that appeared in the 1970s and was developed by Trygve Reenskaug, who worked for the company Xerox Parc. Later in the 1988s, the MVC concept has been approached in a different manner by Krasner and Pope. They wanted to confer in their publication, with this pattern, a way of implementing applications, much easier by working in a modularity manner. In this way, building applications is accessible, because the packages are more organized, and the developer knows exactly where to find what he wants, without interest to the other modules [18].

The MVC design pattern is used in almost any programming language dedicated to web development, such as Java, .Net, Ruby on Rails and at last, Laravel which is a PHP framework used in the web application for the monitoring system.

This pattern is divided into three main parts, which are:

- Model- which is the responsible for the business layer and the logic requests, which are between the controller and the model classes. The word model itself represents the archetypes of the objects used in the real life. (e.g.: the light object, which stores into the database, the illuminance status of the environment). It is tightening to the databases having a direct relation to the data that the developers are working with. In conclusion, the model reuses and controls the data handling, abstraction, and validation [18] and establishes an object-relational mapping between models and database structures.
- View- as it shows in figure 4.1.4 is the interface that puts the user in direct contact with it. This the graphical layer also called front-end which consists of the elements such as different buttons, tables, galleries, etc. On the base of these views are the HTML sketches, followed by Javascript which defines the functionalities of the elements, and lastly the CSS which consists by defining the design of web pages. Of course, using these programming languages isn't mandatory as nowadays there are plenty of frameworks which can be used (e.g.: Bootstrap, React.js, Angular and many more). The role of the view is to request data from the controllers and show it to the users, in short, to meet the user's commands.
- Controller- is the bound layer of the system that fulfills the request of the view layer and exports data from the model to the view. For each model is assigned a controller to handle the events and functionalities. It can work with different HTTP requests and Ajax calls by parsing JSON files. It can arrange the dynamic and static structures in a similar time and can handle all the business logic to achieve a final output [18]. That being said, it can be concluded that is the core of most of the applications.

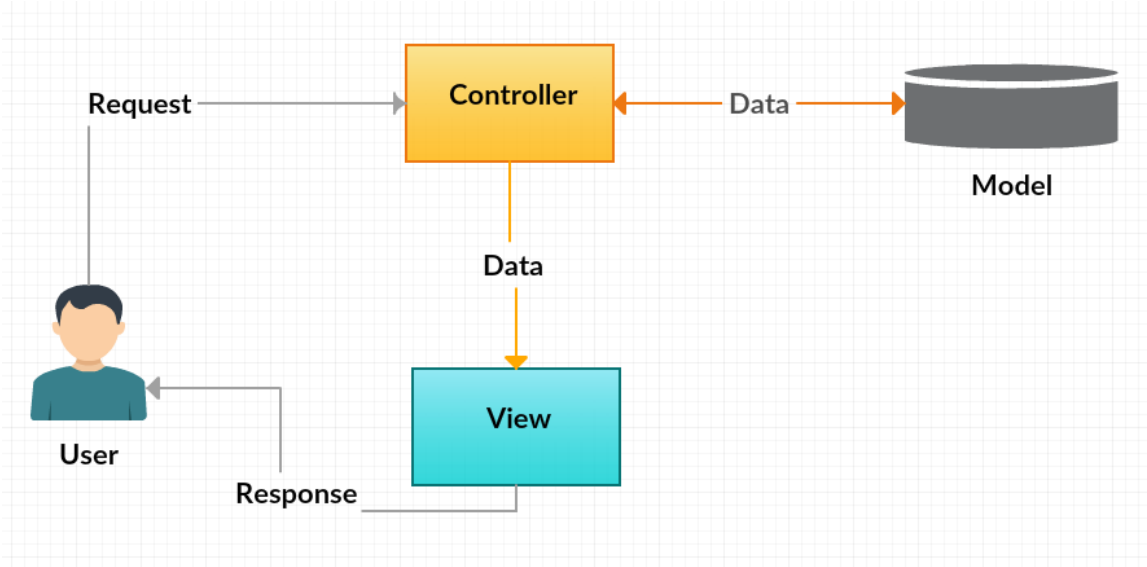


*Figure 4.1.4 MVC architecture*

## 4.2 Laravel

Laravel is one of the most popular and stable framework nowadays, created by Taylor Otwell in 2011 [19]. This framework is based on the PHP language and is making some implementations easier for web developers by using the artisan commands, routing, and any other functionalities. It's an open source framework and is easy to work with it, as it's using the MVC design pattern.

It works as an intermediate system and manages the logic and the styling of the back-end contents. It stands in the front, so security rules can be implemented from the front level and can control the routing and process the HTTP requests.

It can run on any kind of operating system and can work with different database services, and different front-end frameworks like Bootstrap. It's flexible, versatile and very easy to work with. With the components used, it offers a set of functionalities that make the achievement of a project much faster and easier [20].

If any developer knows the PHP language at a basic or advanced level, with this framework, will cut the time spent to build an application and will provide several security implementations [20].

The are several advantages of this framework [20]:

- Scalability
- Time saved by reusing different components
- Interfaces and namespaces that help to organize the scripts
- Artisan commands that are used in the command prompt

Other features that we may take in consideration that the Laravel framework implies are [20]:

- Modularity
- Routing
- Testability
- Query builder
- Configuration Management
- Schema Builder
- Template Engine
- Queues
- E-mail
- Authentication

Laravel is a powerful groundwork as its strengths are the blade templating engine used which implies that neither PHP and HTML languages will be mixed, different layouts, partial views, and the possibility to work with control structures.

That being said, Laravel is the finest option to build the required web application to monitor and control the stand.

### 4.2.1 Composer

To achieve the installation and a workspace environment for the Laravel framework, it was needed a tool named composer that allows the users to create the projects and to manage the dependencies in a JSON text file. All the third-party libraries and plugins are managed and handled very easily with the help of this tool [20].

The composer.json file can be located in the folder of the created project, then in the "Illuminate" folder, with the other components generated by the artisan commands.

In this script, the version of the framework and the used PHP language version, and also the Github address which provides the source code, different issues and resources can be found.

### 4.2.2 Artisan

This keyword is a command line used in the command prompt interface, which in turn contains a batch of more commands. This kind of commands help the users of this framework to add different contents to the developed applications [20].

By using commands like "tinker", "migrate" and "package". It can also be used the command "php artisan make:auth" to create the authentication API and configuration [19].

That being said, this is what exactly the web application contains for the security level and middleware for the different users. It also provides the register, login view blades from scratch and hashed password and tokens generated in the database.

One important aspect to point here, is the used guards for the authentication, which in Laravel are like cookie stored for different users, in this case, users and administrators. Based on the assigned guards, each user or administrator have a middleware route of

accessing the pages intended for them. It can be declared any number of guards, as long as they have the necessary models and SQL tables created on the server.

### 4.2.3 Blade templating

The blade templating lets the developers define the front-end by using the extension ".blade.php" files and which are usually found in the resources file. Blade views are compiled by the PHP interpreter and can be modified even the changes are cached. Two advantages of this views are the inheritance and template sections [19].

This templating engine is a type of syntax that can be used inside views to be more readable and easier to display data. The view files have a strong relationship with the "web.php" file which defines the routes to them.

### 4.2.4 Eloquent object-relational mapping

As in the MVC chapter, the models were mentioned, the implementation of the monitoring web application is supposing to create the necessary entities for the database tables in which are stored the sensor data. In Laravel the object-relational mapping is achieved by Eloquent, which is a relational interface between the database tables and objects created in the project. The queries are written in a simpler form and doesn't need furthermore declarations to link the models with the SQL tables, because creating a model with an artisan will also actualize the database schema.

Two important aspects of the Eloquent ORM are the conventions. The first one is the declaration of a new model implying the creation of a database table with a name that has a suffix defining their plurality. The second convention is the timestamp attributes "created at" and "updated_at" which are a requirement for the Laravel framework [19].

The models used in the web application are:

- Admin
- User
- Light
- Soil
- Temp

All these entities listed above are PHP objects that extend abstract class "Model". Each of them has assigned a controller and a table in the database. The first two represent the type of users that can access the application based on their credentials. The last three are the data collected from the sensors, according to the real ones, which can be managed by the administrator. The files providing the models can be found under the folder "app".

### 4.2.5 Routing

As in the last chapter, the routes were mentioned, they are the pathways to the front-end. This method is very relevant for this framework as it binds all the functionalities, middleware, HTTP requests and methods.

In figure 4.2.5 it is one of the instruction commands that adds a route to a view template blade. In its component is the "get" word which means that is making a HTTP request to the "light" page. It is using the controller "LightC" that calls the function "index".

The index is the function declared in the controller Light which has the task to display all the light values recorded in the database.

At the security level, the keyword “middleware” defines as its name says, a middleware for whom will let to access the page. In this example, the only ones that are allowed to view the page Light, are the administrators.

The “name” keyword defines another name that can be used to ease the work of the developers. It’s like a name for a variable, used place to place in the back-end of the application.

Routing is making the connection between the View and the Controller parts, in the MVC pattern.

```
Route::get('light','LightC@index')->middleware('auth:admin')->name('light');
```

*Figure 4.2.5 Routing in Laravel*

### 4.2.6 Controllers

From the MVC pattern, it is known that controllers have the main objective to associate the views with the model’s data. In Laravel, the controllers can handle the logic in different control files, and not in the route files as in plain PHP programming, encouraging a better organization of the layers. They can be found under the file “app/Http/Controllers”, and all extend the same base class named “Controller” [19].

In the implementation of the application, there are a couple of controllers associated to the models specified in a previous chapter. All of them include the necessary methods to implement the CRUD operations for the managing system data, and also the functions to display views.

In figure 4.2.6 it is shown one of the functions implemented in the Temperature controller. This method displays all the values stored in the database in a descending order by “id” and returns the web page where all the data is listed. It has a public access modifier because it needs to be used by other classes that are not in the same package.

It is using the “Temp” model class and requests the values with a HTTP get request. The “$temp” variable is retrieving a list with all the temperature values from the database through the model. In turn, the “temp” view will receive the task to list all the values stored in the “$temp”.

There are several other functions implemented into the controller’s classes that provide the necessary functionalities, for the CRUD operations.

All these functions are working in parallel with the front-end javascript functions, which are the ajax calls with the JSON bodies and the models declared in the separated classes.

This is what in figure 4.1.4 is shown overall, the controller is the middleware that connects the Model and View structures in a logical manner. By calling the user input

commands, and extracting the necessary information from the model, delivering it to the view.

```
public function index(){

    $temp= Temp::orderBy('id', 'DESC')->get();
    return view( view: 'temp.temp',['temp'=>$temp]);

}
```

*Figure 4.2.6 Controller index method*

## 4.3 JSON

JSON or also known as JavaScript Object Nation is a standardization for the transfer exchange of information. These kinds of files have a specific format composed of bodies which represent specific data structures or objects. JSON serialization can be parsed by almost any language in way or another, in this case, PHP and C++, and can be used by calling AJAX requests [21].

This technology is an advantageous alternative in addition to XML because it has a more compact format and doesn't need additional libraries.

In the sensor data management system, and in the graphical view of the collected data, the processed JSON files are transferred through ajax calls, which allows updating the webpage and making a difference between CRUD operations.

Also, the microcontroller which controls the actuators works with a JSON file, parsed by the C++ language to find the what commands are written and demanded by the user.

## 4.4 Jquery AJAX

Jquery is a javascript framework library which can help a lot in need of compatibility between web browsers. It is used for different tasks like animations, effects, events, minimizing the javascript codes and Ajax calls.

An ajax call is an asynchronous HTTP request to the server that has several functions to implement the CRUD operations like load, get, put and save. It works with a couple of required commands like [22]:

- Type-which demands the type of the request (e.g.: get, post, put, etc.)
- URL- the address of the manipulated view
- Data- the variable with the listed encoded JSON file
- Success- the return if every instruction worked fine

An important aspect for the Jquery framework is that it is required to have a minimum knowledge of how to work with CSS.

## 4.5 RESTful API

The representational state transfer application programming interface makes conceivable the requests from the clients to the server, is built on this type of architecture. Having two clients, one for monitoring, and one for control, calling the necessary requests to the server through a Rest API.

Both the web application and microcontrollers use the standards of Rest API's. Representation is one of them and implies the fact of exchanging data is based on JSON files. Thus, this principle implies another one, which is that a REST web service is stateless, meaning that in a transfer, both the server and the client incorporate all the necessary information to perform the demanded request [23].

The most used HTTP request in the whole implementation is the GET request because the server demands a resource from the monitoring microcontroller, which gathers the information from the sensors. On the control system, the second microcontroller calls a GET request to retrieve the commands from a JSON file stored into the server. It can be said in an informal way, that the monitoring system store information into the server and the controlling one interprets it. An advantage of this type of request is that even if the connectivity is lost between the client and the server, the request will reload.

For the web application, where the other functions were used (create, read, update, delete) for the persistence of the models. Were used other types of requests like:

- POST- creates a new resource
- PUT, PATCH- modify the content, but can also act as a POST request
- DELETE- clears data, also known as destroy command

## 4.6 Graphical interface

### 4.6.1 Cascading Style Sheets

Cascading style sheets also known as CSS, defines implemented design layer over the HTML elements. It is used to make the user interface as friendly as possible.

The advantages are that the external CSS files can spare a lot of time, in case it is wanted to edit multiple pages, and can define different animations and effects for the graphical output [24].

It uses different tags to define properties of the elements such as color, resolution, size, font, and style.

In Laravel, the default used CSS framework is Bootstrap which will be referred to in the next chapter.

There are three different ways to implement the CSS scripts [24]:

- External- that can change multiple web pages and the script location is in another folder
- Internal- that can change only one web page and it's declared in the same script
- Inline- that can change only one element from a webpage

### 4.6.2 Bootstrap

Bootstrap is a front-end framework that mixes HTML, CSS and javascript into a single library. It is very popular nowadays and it is used for the user interface in the web application.

It is very easy to use and helps most of the developers with less experience into this field. It can define a wide range of elements from buttons, tables, fields and many more.

In the user interface, there were reused some elements from Bootstrap. The administrator and user dashboards contain graphical components related to this framework.

A component that makes the application more intuitive, suggesting to users what it's the purpose of the implementation, is the carousel. A slideshow of three different pictures representing the needs of a plant.

### 4.6.3 Highcharts JS

To establish a graphical view of the data collected from the sensors, a javascript framework named Highcharts was needed.

It is an open-source and free API which can create interactive and dynamic charts related to timestamps and are written in pure JavaScript. It appeared in 2009 and confers a way to display data on a multitude of different browsers and devices [25].

It has the options to plot different type of lines with different colors and legends. The main function that makes possible the implementation is the command "$getJSON" that uses a json body file with all the values stored from the targeted SQL table.

An important aspect to mention here is that after the timestamp will be queried from the table, it will be converted into epoch UNIX time stamp, which is a convention.

In figure 4.6.3 the collected data from the light sensor are shown and displayed in a graphic that contains the timestamp and the lux value of the illuminance flux.

The sensor was tested for an hour with a delay of 5000 milliseconds. But it concluded that the best amount of time to monitor the units is a delay of 5 minutes.

The method used for this graphic is called "Time series" which is also zoomable and has incorporated into it's component a drag point that shows exactly the value associated directly with the time.

It has the advantage of plotting the necessary graphics, to assure the user with the option to interpret the data received in the system.

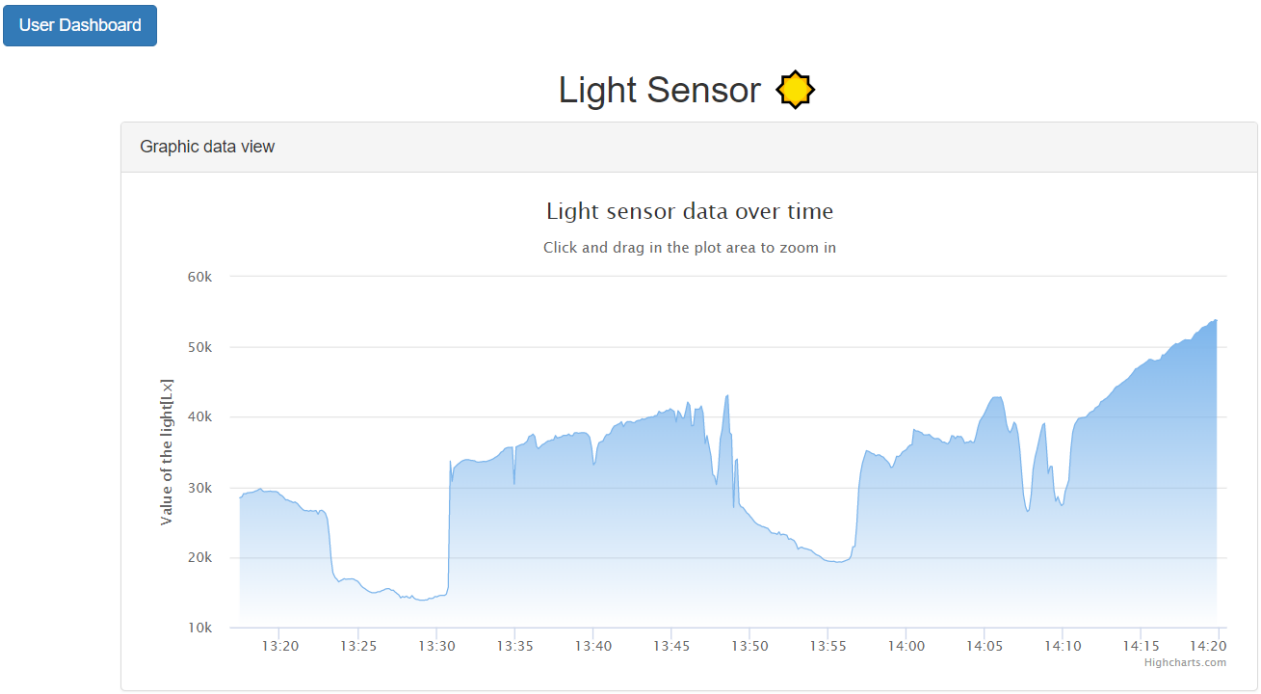


*Figure 4.6.3 Plotting light sensor data*

## 4.7 XAMPP Server

The local server used, that hosted the web application and made possible the connectivity between the two microcontrollers clients and the server is XAMPP.

It is an open source and free software that was implemented by Apache Friends in 2002. It consists of different language interpreters such as PHP and Perl and has also in its component a database service named MariaDB [26].

The Apache server runs on ports 80,443 and the MySQL database runs on 3306. Without these configurations the connection between all the three components included in this project, wouldn't be possible.

Under the folder where XAMPP is installed, it is included the PHP file that establishes the connection with the monitoring microcontroller and requests the information from the sensors. Here also can be located the other files for the web application.

# 5 Implementation and detailed design

## 5.1 Database structure

The database used for this application is MariaDB which is in XAMPP software's component. It is easy to use, doesn't need an additional installation and can be accessed straightly from the web browser.

In figure 5.1 shows a database schema called "licenta" which contains seven different tables that will store the necessary information to implement the essential functionalities of the application.

The following tables: users, admins, password_resets, are storing the accounts created by the users and administrators. All of them have a common row named "token" which has the task create unique values for each other unique emails registered into the database. The password_resets tables is registering each time a user requested to change their passwords.

Another important table is "migrations" which is automatically created simultaneously with a new Laravel project. It keeps track of every migration the developer committed, managing the work and making easy to know what changes has been made to the project.

The tables at the bottom of the figure are registering data collected from the sensors. Each of them has a directly bind to the assigned sensor and a timestamp to know when the request has been sent.

All of them have in common the primary key "id" which is a unique value that can give the option to distinguish the values. The "updated_at" timestamp tracks when changes were made.

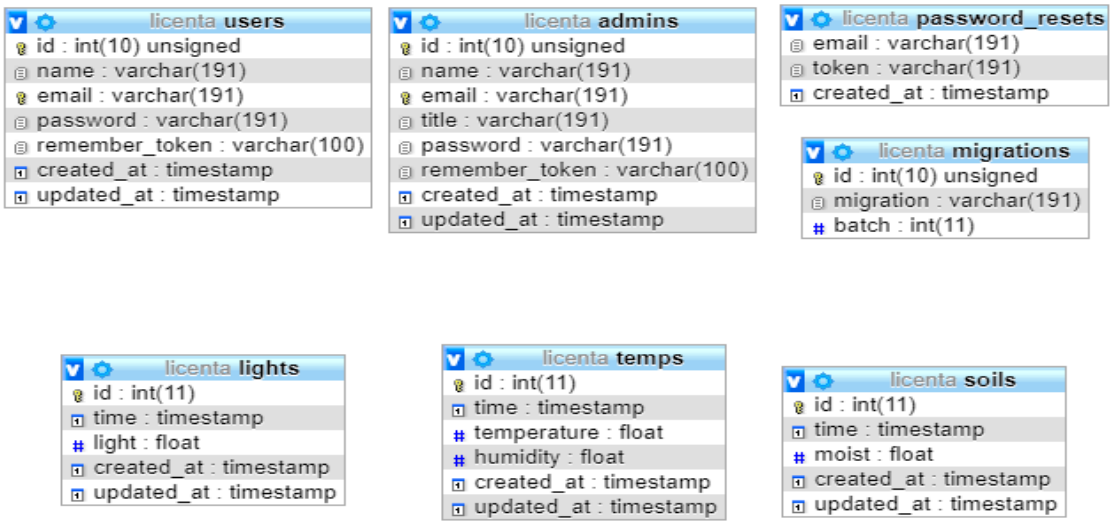


*Figure 5.1 Database structure diagram*

## 5.2 Use Cases

In the following diagram 5.2, it is illustrated how the application is divided into two parts. Each one having special users targeted.

### 5.2.1 Users

For the users it is highlighted the fact that there are constrained options. They can register into the application, login and visualize the data gathered from the sensors. It is a way for administrators to share the status of their indoor growing plants with different individuals.

### 5.2.2 Administrator

The administrator has the options to manage the entire system. It has full control and can decide which users can access the application or not. To manage the sensor data in the web application means to use the first client to request data from the microcontroller.

Besides managing the system, it also benefits of the controller system, which imply the second client microcontroller which can change the state of the actuators. The "Control actuators" webpage sends a command request from the server to the client.

The administrator doesn't have a register option because the necessary credentials will be granted by the developer of the web application.

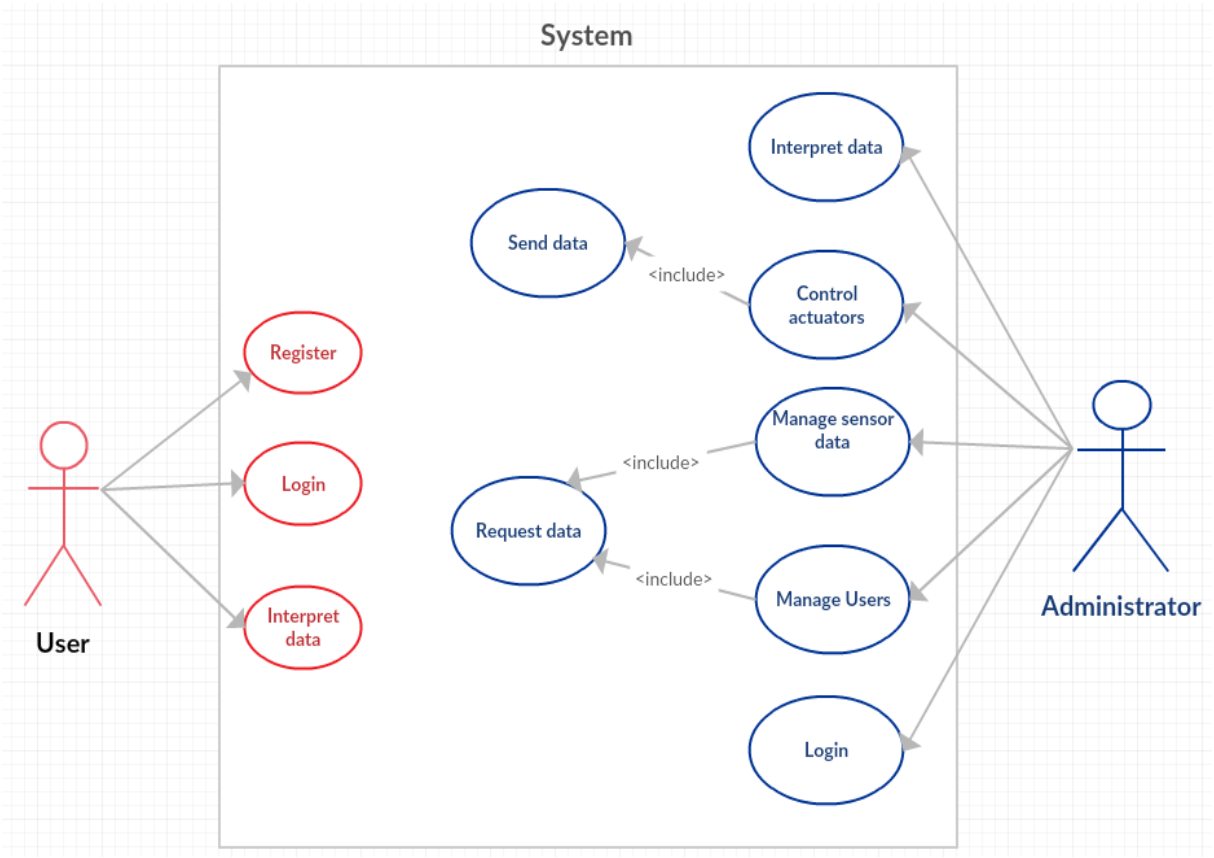


*Figure 5.2 Use Case diagram*

## 5.3 Object interactions

### 5.3.1 Unified Modeling Language

The UML diagram provided by the PHPStorm IDE is the architecture of the web application. It displays how the application was build and what classes and interfaces it contains for the model entities.

In figure 5.4 it is displayed how the classes: Light, Temp and Soil extend the abstract class Model which is created by default in Laravel. They contain the attributes that correspond with the tables created into the database: id, timestamps and the measured value, all stored in a class property array called “fillable”.

The abstract class Model implements all of the interfaces to achieve the connection to the database, creating routes, converting objects into Json’s and other implementations. It is the base model class that does the job for the new models created with the Eloquent. It makes possible the object relational mapping. The most important attributes from this classes are: connection, table, primaryKey etc.

The classes: Admin and User, are extending indirectly the interface Authenticable. There are two classes with the same name because one is defining the type of individual and one is created by default in the authorization artisan. Each of them contains a set of attributes and constructors. Both Admin and User class contain the protected attributes “guard”, “fillable” and “hidden” which contain the attributes necessary for the register credentials.

The interfaces that are implemented by the User class contain the necessary methods to create an authorization system which gives the confidence that can be hardly accessed by unauthorized persons.

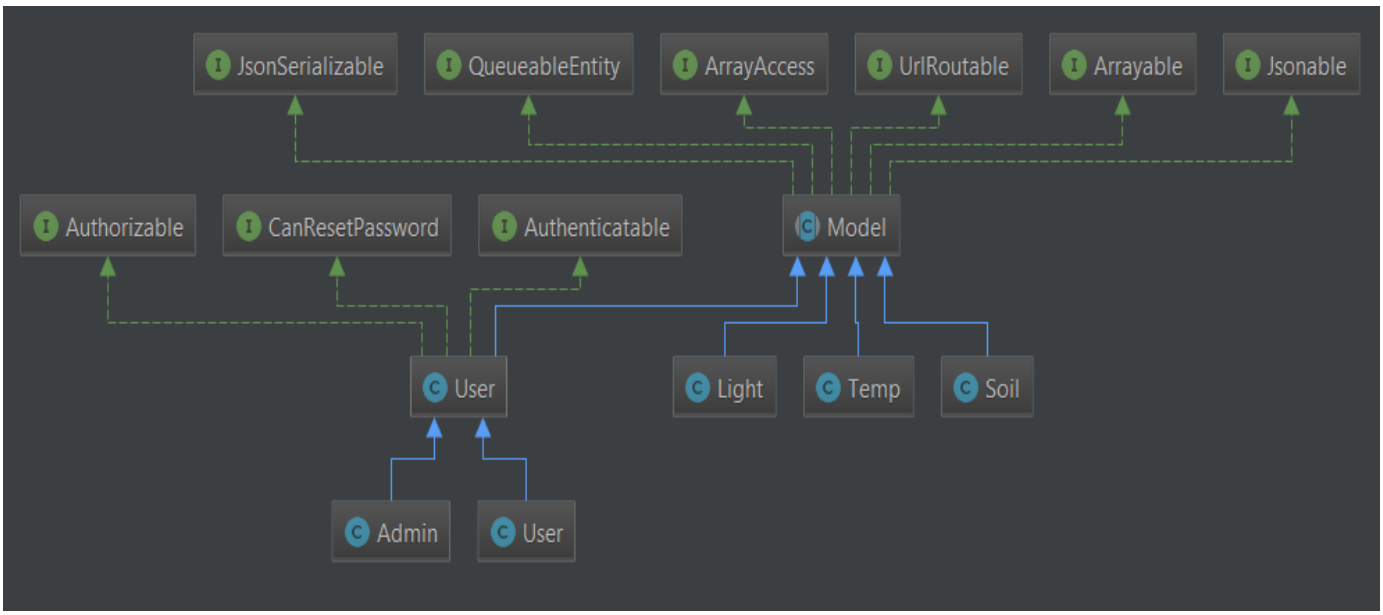


*Figure 5.3 UML diagram for models*

In figure 5.4.1 is the UML diagram for the control part of the application. Each controller is extending the class “Controller” which in turn extends the abstract class called by default also “Controller”.

The abstract class “Controller” contains the protected attribute “middleware” which establishes the routings for users and administrators. It also contains the functions to get the middleware routes and call different actions.

The class “Controller” that extends the abstract class is using different traits that aren’t displayed in the UML diagram, but having several functions that intends to use different methods in group, reducing the implementation time.

The “HomeController” and “AdminController” are the controllers used for administrators and users. They contain the functions required for the views routing mentioned in a previous chapter and the function that redirects the individual to the targeted view if the credentials entered in the “login” form are valid or not. The constructor used for both classes is the one that defines the middleware, the only difference between both of them, is the used guard.

The other controllers: SoilC, LightC, UserC, TempC are used for the entity data models used. Each model having assigned a unique controller.

They contain public functions that are making possible the management system to work, in other words the CRUD operations that are demanded for such a system. One of the functions was explained in a previous chapter. Each of the functions is requesting an ajax call from the front-end part of the application.

In conclusion, the software architecture is combining the model, view and controller components, in a very understandable way for the developers, with the Laravel framework.

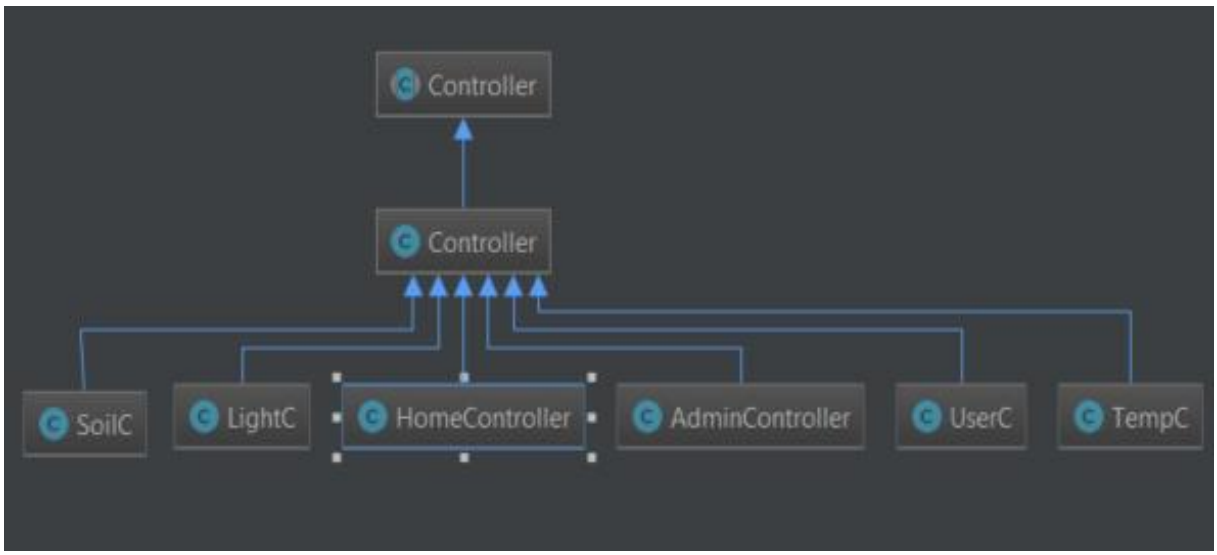


*Figure 5.3.1 UML diagram for controllers*

### 5.3.2 Sequence diagram

The following sequence diagram in figure 5.2, lines up how the data requests are working. The workflow is showing how the models and controllers interact with different types of individuals. The administrator has multiple options to handle more events than users. It is shown by the horizontal lines how the requests are made on different models or how they are managed by the controllers. The extremities of the arrows are different because they represent if they are synchronous or asynchronous based on the action made by the administrator or user.

The "get sensor data" request is asynchronous because it depends on the monitoring client when the server will receive the demanded information. The other actions are asynchronous too, because the time isn't known when the individuals will decide to make any actions.

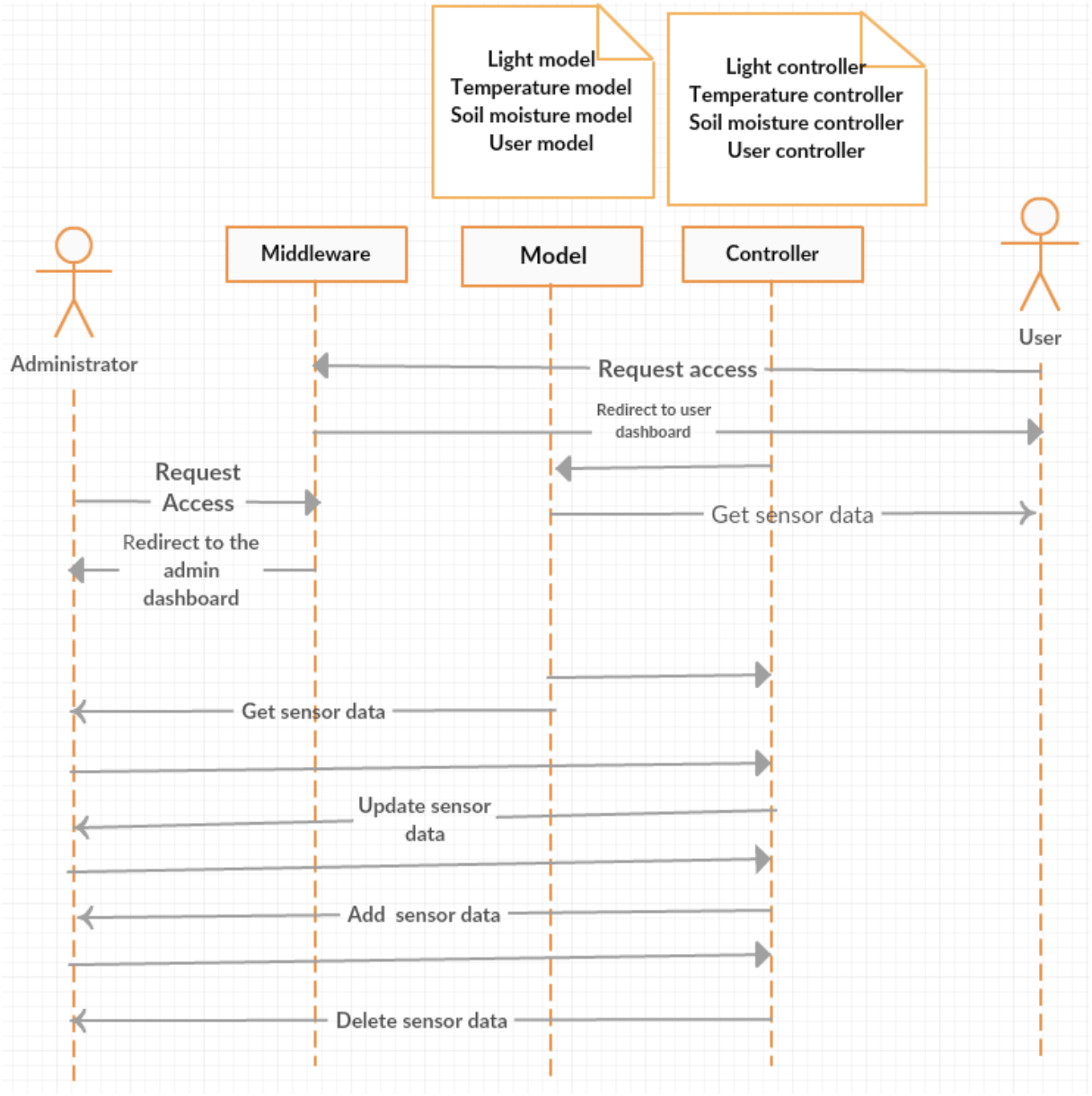


*Figure 5.3.3 Sequence diagram*

# 6 Testing and validation

To achieve the testing for the monitoring system it was needed to connect the client and the server on the same internet connection. In this way the communication through the IP protocol would have place.

The client's behavior was tested from the Serial Monitor in Arduino Ide. In figure 6 it is displayed how a request has been sent to the server with the sensor readings. In this way it could be observed if the connection was established and the readings were accurate and concrete.

Another factor that had to be taken into consideration, was the baud rate of the serial transmission, because there weren't so many data to send, it was chosen a middle value of 9600 bits per second.

If the client couldn't connect to the local server, it would print a message informing that the connection has failed and would try again after the imposed delay time. Also, another message would have been printed if the development board couldn't connect to a wireless connection.

It was tested the case when the delay between the readings would be one second. In this case the time wasn't so accurate because the microcontroller didn't have enough time to acquire and send the data in that interval. So, it was decided to let the delay time to be five minutes, being an adequate value for this kind of monitoring system.

The next step was to verify if the data taken from sensors were stored into the database and if the delay was the one imposed.

```
You are connected to WIFI
---------------------------------
Gain:          25x (Medium)
Soil moisture[%]: 22.00%
Ambiental light[lx]=
700.85
 Air temperature[C]= 26.90
Air humidity[RH]= 47.50
<--------------->
You are connnected to the SERVER
REQUEST HAS BEEN MADE
```

*Figure 6 Arduino Serial Monitor Request*

In figure 6.1 is displayed the home page of the web application which is composed by different hyperlinks and elements to make the user access and understand how to start using the application's purpose.

Under the title is a component of notifications that informs the users, who is logged into the system.

It contains the options to login and signup for the users and only the login hyperlink page for the administrator. The "Github" referral leads to the source code of the web application.


*Figure 6.1 Home page of the web application*

The next testing case was to try the functionalities implemented for the security layer.

Like any other signup page, there were needed several fields to be filled to achieve a complete registration. Figure 6.2 displays the web pages, that a user should access to gain credentials to enter the application. The administrator doesn't need any registration page because the accounts will be delivered by the developer.

The "Forgot your password" functionality was tested with a free software called mailtrap.io which is a free smtp testing server. It provides a temporary inbox that catches all the mails sent from application to the users. In this way it wasn't necessary to disturb anyone to validate this function.

To improve the application in the future, a paid smtp service like SparkPost or Mailjet would be needed to send the mails with forgotten passwords requests directly to the persons using the application.

It was tested also the case that the password would be less than 6 characters and if the used email was already taken by somebody else. In this context, the register page would notify the person who wants to register, demands to change the invalid fields.

The login page was tested with the same testing units as the register page, and the specific guard for each route. Because it was an important fact to assure the basic users couldn't get unauthorized on forbidden pages.

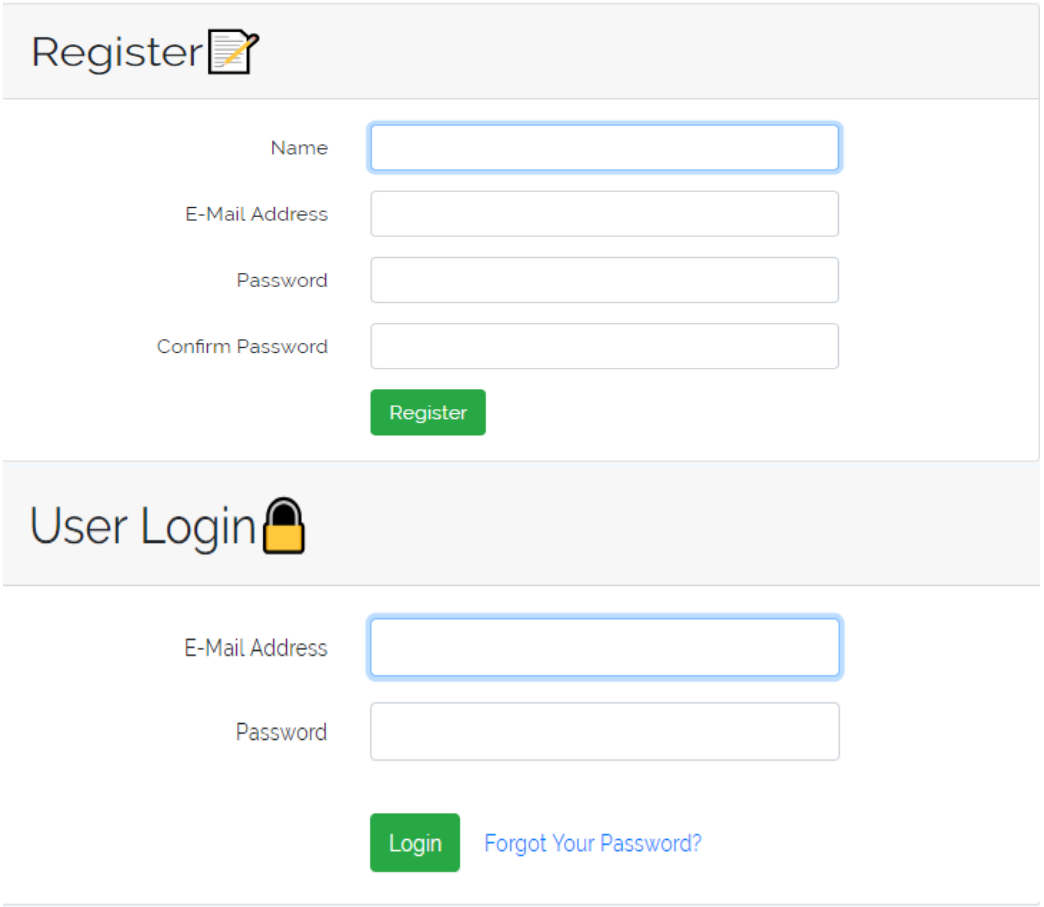


*Figure 6.2 Login and register panel*

If the credentials were correct submitted into the login page, the users would be redirected to the corresponding dashboards.

The administrator dashboard is displayed into the left part of figure 6.3. It consists of three categories: control, sensor data management and graphic data view. Each subcategory containing different buttons with suggestive symbols, respectively to the measured climate factors. It was needed to validate the redirections of these buttons to the specified web pages.

In the right section of figure 6.3 is displayed the control page that has the input commands represented by bootstrap buttons. Another testing case was to guarantee that the JSON file is encoded properly and parsed effectively by the NodeMCU microcontroller. Likewise, the physical system required some test cases in the serial monitor to check if the relays would change the power supply status. It was necessary to add an extra switch for the water pump relay because, plugging in the system to a power supply would make the relays to activate the actuators leading to an unnecessary supply of resources.

A disadvantage of the web application and control system is the postpone time between the input given command and the moment when the water pump is spreading the liquid.

The control page can't start simultaneously the actuators for safety reasons. It will turn off automatically if the any of them will change their state.

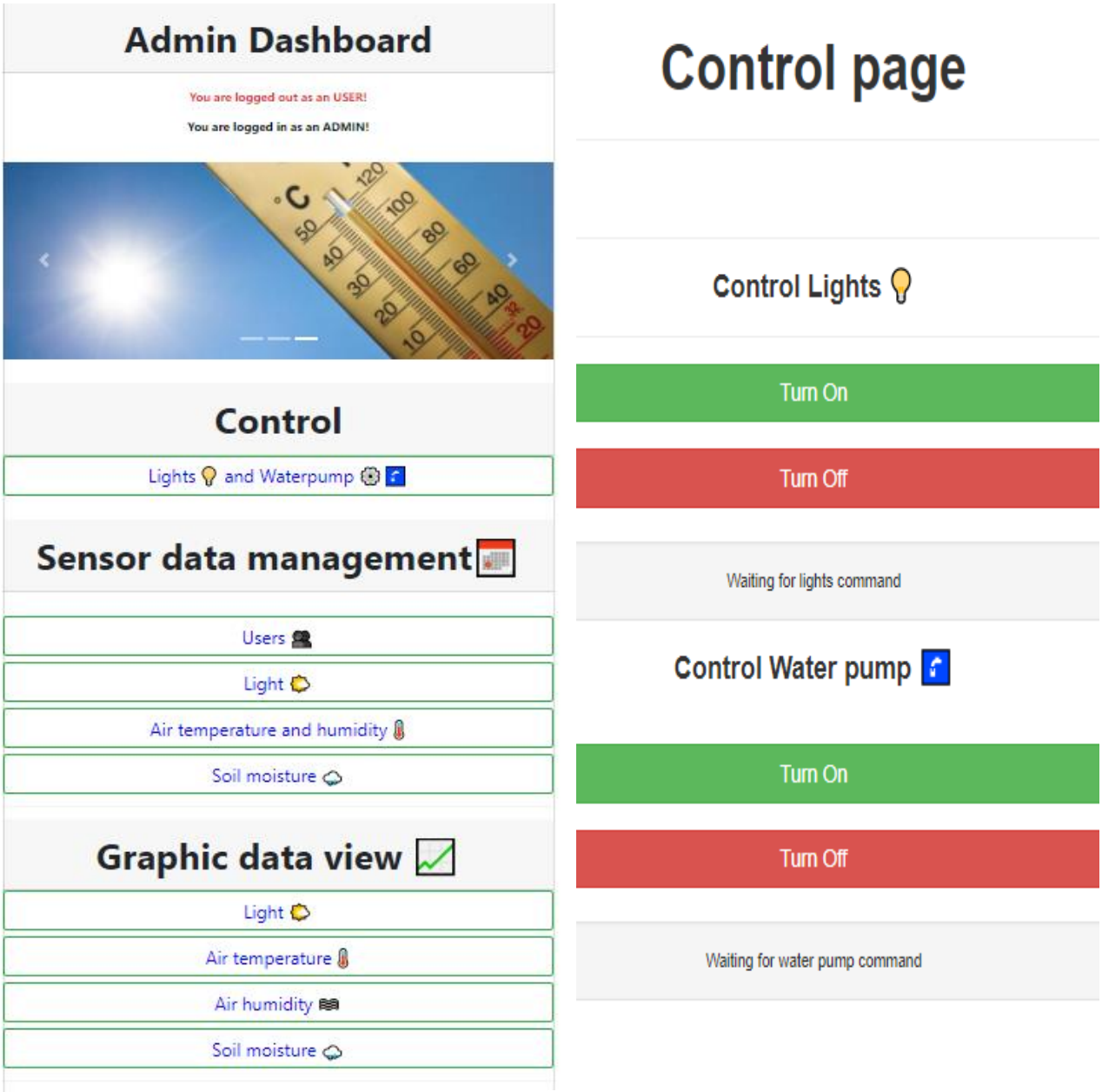


*Figure 6.3 Admin dashboard and control panel*

The user dashboard is similar to the one in figure 6.3 but with limited options. It doesn't have the control page and neither the option to manage the users or any other sensor data.

The last testing case was to try the management system from figure 6.4, and all the CRUD operations implemented. Each functionality was tested separately to reassure that all the http requests tested before with POSTMAN were correct.

The users don't have any of these options, only to auto-refresh the page. The auto-refresh is an approach to view the changes of the climate factors live.

An improvement at this topic would be to implement a more organizational way to view the data. A search bar with a date-time picker would improve the management of the system.

The option to view the sensor data in different plots was displayed in figure in 4.6.3. A graphic that has on the x axis the timestamp and on the y the measured climate factor.

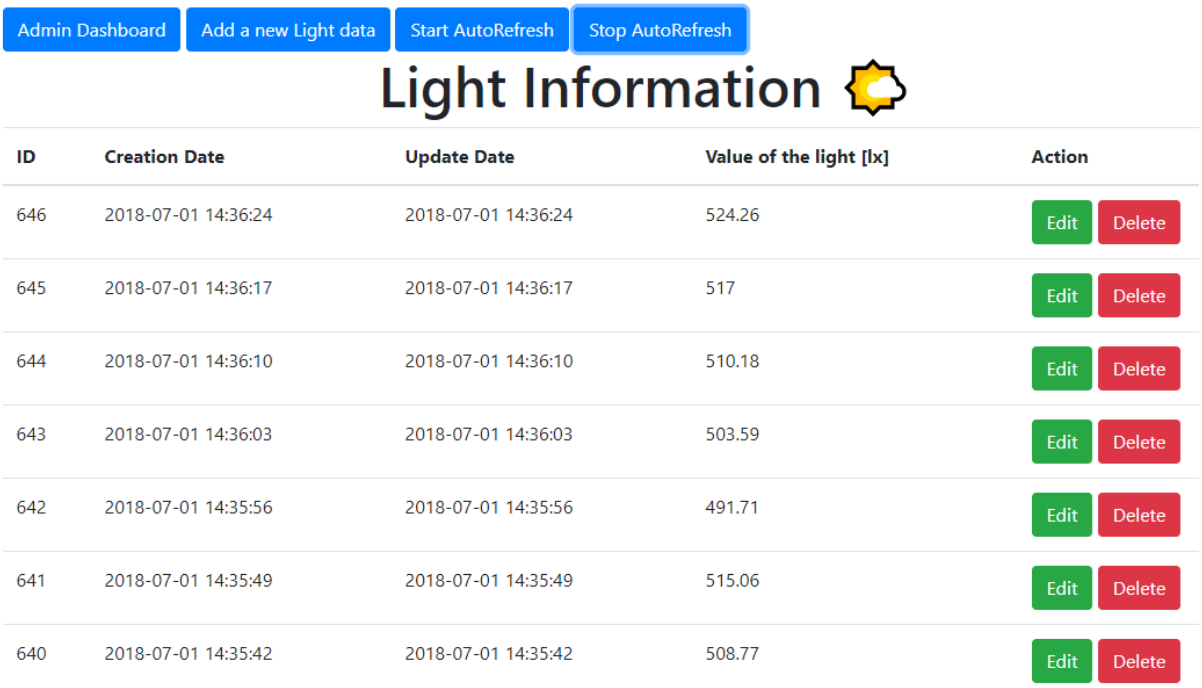


*Figure 6.4 Management of the light sensor data*

# 7 Installation guide and user manual

## 7.1 Installation

To achieve a successful installation for the entire system, a wireless internet connection is needed. Without it, the application can't provide the functionalities.

For the web application is needed a local server software like XAMPP or any other alternatives and the Composer dependency manager for PHP. If the application would have been hosted on a paid server these two wouldn't be necessary.

Another important aspect is to know the IP of the hosting computer or the name of the website if it is hosted on a public server. Then the Arduino scripts should be updated with the new IPs.

To edit the used files for the software implementation it is desired to install Arduino Ide and JetBrains PHP storm or any other text editors.

Any elements that belong to the physical framework can't be changed by the users, as the stand was built to have two current plugs, which are meant to power the entire system and the sensors were soldered into prototyping boards.

## 7.2 User Manual

The application is dedicated to users with minimum technical knowledge. The front-end interface is friendly and easy to use. Intuitive because of the buttons and commands input, it is understandable and intuitive for users experiencing the first contact with the web application.

The administrators are the ones who own the monitored plants and have granted access to the control commands and to the management system.

It is suggested to have only one administrator and more users because in this way there won't appear conflicts at the input command. Therefore, the application's pattern was designed to maintain only one admin and one user connected simultaneously in the same session of authentication.

The most important detail of the application is that the users and administrators can observe the live data collected and the plots of the climate factors. In this way it is ensured that the environment is optimal for the plants.

# 8 Conclusion

The concept Internet of Things is an important factor nowadays that will expand in more areas and will develop more embedded systems. It is used in present together with many more concepts like: Big Data, Cloud Computing and Industry 4.0.

If any research and further development will be used in agriculture, the efficiency of the greenhouses will increase implying the fact that also the marketing and economical fields will have better gains.

Many software technologies, frameworks and electronic knowledge were used to develop this project and aims to improve the lifespan and indoor growing. Leading it to a larger scale could help different farmers or hobbyists to monitor and control greenhouses remotely, reducing time, money and energy.

Considering all the mentioned factors, it can be said that the greenhouse monitoring and control system is an application implemented for different individuals with basic or technical knowledge.

## 8.1 Further improvements

We can take in account adding three more sensors for the monitoring system: one for Ph level, one for water level in the tanks and another one placed near the pots for water drops. Also, a webcam would be a quite good improvement to know exactly how the plants behave.

For a household applicability, a Bluetooth connection would be safer and simpler achieving a connection directly from device to device. For industrial and bigger greenhouses this type of connection would be a disadvantage.

Another improvement would be to embed the functionalities of the two microcontrollers into one, reducing the cost.

For the software implementation, a date-time picker would be helpful for a more efficient way to manage the data sensors and users. A paid smtp service would be required to send the emails requests with the forgotten passwords directly to the users. Also, a Facebook API could be added to the web application, or any other social API alternative would be an upgrade on the online community field, allowing the users to share their plant's status.

The application is using at the moment, a local server. A paid hosting service would be the best improvement in order to expand the area of remote utilization and an elaborated community of users.

# 9 References


[1] "Preventing cold-related morbidity and mortality in a changing climate," 2011. [Online]. Available: https://www.maturitas.org/article/S0378-5122(11)00116-2/fulltext.

[2] "Climate change and food safety: An emerging issue with special focus on Europe," 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278691509000714.

[3] "A short history of internet protocols at CERN," 1995. [Online]. Available: https://www.researchgate.net/profile/Ben_Segal2/publication/245061001_A_short_history_of_Internet_protocols_at_CERN/links/00b7d52b05f310a304000000/A-short-history-of-Internet-protocols-at-CERN.pdf.

[4] D. D. Roger, Network management system using interconnected hierarchies to represent different network dimension in multiple display views, 1990.

[5] "Internet Of Things(IoT):Research, Architecture and Applications," 2018. [Online]. Available: http://www.ijfrcsce.org/download/browse/Volume_4/March_18_Volume_4_Issue_3/1520502532_08-03-2018.pdf.

[6] "A Brief History of the Internet of Things," 2016. [Online]. Available: http://www.dataversity.net/brief-history-internet-things/.

[7] P. Mehul, "Internet of Things (IoT): In a Way of Smart World," 2016. [Online]. Available: https://www.researchgate.net/publication/303809222_Internet_of_Things_IoT_In_a_Way_of_Smart_World.

[8] A. A. B. K. S. Karan Saxena, "Internet of Things," 2015. [Online]. Available: http://ijesta.com/upcomingissue/04.04.2015.pdf.

[9] L. Doug, "Application Delivery Strategies," 2001. [Online]. Available: https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

[10 ] S. Ravindra, "Understanding the relationship between IoT and Big Data," 2017. [Online]. Available: https://jaxenter.com/relationship-between-iot-big-data-138220.html.

[11] S. Herle, "Lecture 13 Revolutions and Evolutions in Production," [Online]. Available: http://rocon.utcluj.ro/sorin/CIM_c/C13.pdf.

[12] "Microcontroller," 2017. [Online]. Available: http://ethw.org/Microcontroller.

[13] R. BOUAFLA, "IoT Based System for Greenhouses Remote," 2017. [Online]. Available: https://bu.univ-ouargla.dz/master/pdf/MECHALIKH-BOUAFIA.pdf?idmemoire=6690.

[14] K. B. K. Tzounis, "Internet of Things in agriculture, recent advances," 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1537511017302544.

[15] N. R, "Relative Humidity," [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/Kinetic/relhum.html.

[16] A. industries, "Adafruit TSL2591 HIgh Dynamic Range Digital Light Sensor," [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-tsl2591.pdf.

[17] R. Rocha, "How to convert lux to watts (W)," [Online]. Available: https://www.scribd.com/document/214483834/How-to-Convert-Lux-to-Watts-W.

[18] D. P. Pop, "Designing an MVC Model for Rapid Web Application Development," 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187770581400352X.

[19] T. Otwell, "Laravel Documentation," [Online]. Available: https://laravel.com/docs/5.6.

[20] S. Ndende, "Laravel," [Online]. Available: https://www.tutorialspoint.com/laravel/laravel_overview.htm.

[21] D. Crockford, "Introducing JSON," [Online]. Available: https://www.json.org/.

[22] D. Methin, "Jquery Ajax," [Online]. Available: http://api.jquery.com/jquery.ajax/.

[23] L. Hoguin, "REST principles," [Online]. Available: https://ninenines.eu/docs/en/cowboy/2.1/guide/rest_principles/.

[24] J. Azasoo, "WEB Technologies and internet Programming," African Virtual University, pp. 37-38.

[25] T. Honsi, "Highcharts," https://www.highcharts.com/about/.

[26] S. Kai and K. Vogelgesang, "About xampp," https://www.apachefriends.org/about.html.

Figure 2.6 Integration of Internet of Things into agriculture https://ars.els-cdn.com/content/image/1-s2.0-S1537511017302544-gr5.jpg