INSTITUTO
UNIVERSITÁRIO
DE LISBOA

SoilIoT - Iot for precision agriculture - Soil Characteristics monitoring

Diogo Filipe Guedes Soares

Master in Telecommunications and Informatics Engineering

Supervisor:
PhD Octavian Adrian Postolache, Associate Professor with habilitation,
ISCTE

Co-Supervisor:
PhD Pedro Joaquim Amaro Sebastião, Assistant Professor,
ISCTE

October, 2023

TECNOLOGIAS
E ARQUITETURA

Department of Information Sciences and Technologies

SoilIoT - Iot for precision agriculture - Soil Characteristics monitoring

Diogo Filipe Guedes Soares

Master in Telecommunications and Informatics Engineering

Supervisor:
PhD Octavian Adrian Postolache, Associate Professor with habilitation,
ISCTE

Co-Supervisor:
PhD Pedro Joaquim Amaro Sebastião, Assistant Professor,
ISCTE

October, 2023

Acknowledgements

I would like to thank Professor Octavian Postolache, and Professor Pedro Sebastião for providing the material and supervision during whole master period, I also want to thank to the PhD students that and to thank the PhD students that provide me also some help associated to my work. Special acknowledgements go to Iscte-Instituto Universitario de Lisboa e Instituto de Telecomunicações, IT-IUL that provide me conditions to develop my master thesis work.

I would like to thank my family and my girlfriend for providing me support and strength whenever I need it.

And finally, to my girlfriend's family for providing me support and for letting me make tests in their house.

Resumo

Atualmente, devido às condições das alterações climáticas, secas prolongadas e diminuição do campo de cultivo, devido ao aumento exponencial da população, a agricultura necessita de uma abordagem mais sustentável. Soluções que atendem à sustentabilidade estão associadas à agricultura de precisão.

Neste contexto, a avaliação objetiva das condições do solo e o mapeamento utilizando sistemas de informação geográfica são de extrema importância. Assim, a medição da concentração de nutrientes no solo como azoto, fósforo ou potássio bem como a humidade e a temperatura são essenciais para controlar o nível de stress de água e de nutrientes que permite uma grande eficiência produtiva. As soluções de Internet of Things que são representadas por redes de sensores para monitorizar as características do solo, com ou sem contato, são caracterizadas por LoRa, protocolos de comunicação do tipo Wi-Fi e NB-IoT. Os dados dos sensores são úteis para o desenvolvimento de modelos preditivos, para a disponibilidade de nutrientes e para a otimização da gestão da água, sendo considerados também os atuadores na administração ótima de água e nutrientes. Assim surge esta proposta de projeto, que visa desenvolver um ecossistema IoT para agricultura de precisão.

Neste projeto foi realizado três grandes partes do sistema IoT: a base do projecto, que consiste na obtenção de dados dos sensores e posterior envio dos mesmos para o servidor, um servidor que recebe, trata e armazena a data numa base de dados e finalmente o website, para visualização e análise de dados.

Keywords: Internet of Things, agricultura, LoRa, agricultura de precisão, alterações climáticas

Abstract

Nowadays, the conditions of climate change, prolonged drought and decreasing of crops field due to the exponential increase of the population, agriculture needs a more sustainable approach. Solutions that meet sustainability are associated with precision agriculture.

In this context, the objective assessment of soil conditions and mapping using geographic information systems is of the utmost importance. So, the measurement of the concentration of nutrients in the soil such as nitrogen, phosphorus or potassium as well as moisture and temperature are essential for controlling the level of wat stress and nutrient stress that allows a great production efficiency. Internet of Things solutions that are represented by sensor networks for monitoring soil characteristics with or without contact are characterized by LoRa, Wi-Fi type communication protocols and NB-IoT. The sensor data are useful for the development of very useful predictive models for the availability of nutrients and the optimization of water management, being considered also the most important actuators in the optimal administration of water and nutrients. Therefore, this project proposal arises, which aims to develop an IoT ecosystem for precision agriculture.

In this project, three major parts of the IoT system were carried out: the base of the project, which consists of obtaining data from sensors and subsequently sending them to the server, a server that receives, processes and stores the data in a database and finally the website, for data visualization and analysis.

Keywords: Internet of Things, agriculture, LoRa, precision agriculture, climate change

Contents

Index of figures

Index of tables

Acronyms

IoT – Internet of Things

PA – Precision Agriculture

GPS – Global Positioning Systems

DSRM – Design Science Research Method

ML – Machine Learning

LPWAN – Low-Power Wide-Area Networks

MQTT – Message Queuing Telemetry Transport

IP – Internet Protocol

LoRaWAN – Long Range Wide-Area Network

NPK – nitrogen, phosphorus, and potassium

DC – Direct Current

TDR – Time-domain reflectometry

RFID – Radio Frequency identification

LoRa – Long Range

NB-IoT – Narrow Band-IoT

BPSK – Binary phase-shift keying

V – Voltage

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

IDE – Integrated Development Environment

HTPP – Hypertext Transfer Protocol

TTN – The Things Networ

Chapter 1 – Introduction

1.1. Motivation and Context

With the increasing evolution of technology, the Internet of Things (IoT) is one of the most important technologies in our generation. The rapid development of IoT technologies created tsunamis almost in every industry across the world and particularly in agriculture. These massive changes are shaking the existing agriculture methods and creating new wave of opportunities [1]. IoT can collect, sense, store, connect and communicate data to various components in a network. This technology brings very helpful uses and can help resolve various challenges, threats, and problems for the world, giving automatic sensing and control, for effectively decrease the use of natural resources.

Climate change is one of the biggest challenge and threat to the human health and agriculture in recent years. It is defined as significant changes in the average values of weather elements like precipitation and temperature [2]. Agriculture is one of the rare industries where the technology has not been accepted in the large scale, one of the reasons behind that is the economic condition of most of the farmers. However, the agriculture sector is not only influenced by climate change, but also with the exponential increase in population. This increases not only the demand for agricultural product, but also the need for urban expansion around cities, which leads to a decrease in agricultural fields and, consequently, to a decrease in agricultural production. Such increases can be resolved exclusively with the growth of productivity and efficiency of crops. For this, the wastage of available resources must be reduced, and this is where the concept of precision agriculture arises [3].

Precision agriculture (PA) is one of the most important applications of the IoT as well as one of the developments in the field of agriculture in the last two decades, from a perspective of resource and nutrient use, efficiency, and ecological impact. It mainly tries to manage – through measurement, analysis, and appropriate action – the variability of soil and crop parameters, with the aim of optimizing profit, sustainability, and environmental protection. PA uses various traditional and emerging technologies such as Global Position Systems (GPS) and remote sensing. It has been used in the context of large plantations in the United States of America, using GPS, GIS, and soil sensor networks, and not in small projects, such as for farmers with small properties, especially in developing countries [4].

In this thesis, an IoT ecosystem for precision agriculture will be proposed. The contribution of our proposal is to promote a more sustainable way to conserve the waste of natural resources and to help farmers, by giving them a low-cost system and the option to control and view data, like graphic representation and real-time data sended from the sensors, in a website.

1.2. Research Question

The research questions that motivate the thesis to be made are:

- What is the benefit of IoT for precision agriculture?
- What are the different types of communication protocols for this project?
- Is it possible to reduce the waste of water?
- Would farmers learn or even use this new technology and how would we teach it?
- What technologies and programming languages are used to develop the software component, and what are their advantages in this context?
- How does the IoT ecosystem provide adaptability to different soil types, crops, and environmental conditions?

1.3. Goals

The main goal of this work is the development of an IoT ecosystem to measure soil parameters, like humidity, temperature, nutrients concentration, and soil conductivity to control the water and nutrient concentration. These data are useful for developing predictive models for nutrient availably and optimization of water management.

To achieve this goal, three subgoals need to be achieve:

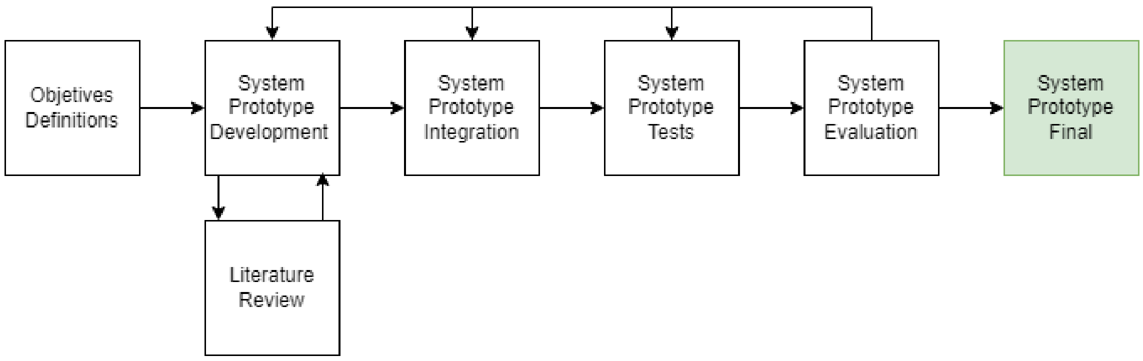
- Develop an IoT ecosystem that includes measurement nodes of soil characteristics, and computing and communication platforms using Arduino board in cojunction with LoRa modules, for real-time data collection, processing and transmission.
- Develop a management software component including geographic information, that presents the nodes location, data recording and graphical representation of the evolution of soil parameters: a website for desktops and mobiles to control, read and graphical representation.
- Develop a server-level data analysis module.

1.4. Research and Planning Method

The research method used in this thesis was based in the Design Science Research Method (DSRM) [5] to divide the bigger practical problem into smaller sub problems. This method was chosen, because it allows us to evaluate the system prototype and cycling back to earlier steps, if necessary, to adjust and improve, until we reach the best possible prototype.

Other alternative for DSRM is the research method Systems Development Research Methodology (SDRM) that includes a five-step research process, but in comparison with DSRM, SDRM is more linear than DSRM, where we focus on research issues by starting with a research question, and in DSRM we define the problem and focus on getting the best possible solution for it, by cycling back to earlier stages.

In figure 1 is represented the main steps that will constitute the research process to my dissertation practical work.



Objetives Definitions
System Prototype Development
System Prototype Integration
System Prototype Tests
System Prototype Evaluation
System Prototype Final
Literature Review

Figure 1: Design Science Research method illustration

1. Objectives Definitions: In this step of defining objectives, will be defined and establish which objectives are to be achieved by writing this dissertation. A set of technologies, processes, tools, and methodologies inherent to IoT and precision agriculture are studied. This step serves as guideline for research, to which the project must obey.
2. System Prototype Development: In this step the different segments and components that will be part of the project are developed to check with the established objectives. The objectives are:
   - Design of the system architecture.
   - Individual sensor and actuator testing, to later bring them all together.
   - Creation and structuring of a database for remote storage of relevant data from the sensors.
   - Development of the application for mobile devices intended for farmers, serving support to see information and control actuators.
3. Literature Review: Main stage of the research, collection and analysis of information content related to the topic and the work to be done in the dissertation. Will be helpful while developing the system prototype, as well for writing the dissertation report.
4. System Prototype Integration: The different components developed in the step before must be integrated to create a prototype of the system.
5. System Prototype Tests: Tests and adjustments are performed to obtain a final prototype.

6. System Prototype Evaluation: The final prototype of the project must be tested and evaluated.

However, we cannot reach a prototype 100% compatible with the user by doing these steps only one time. So, second, fourth and fifth steps are cyclical stages, and if at the end of the sixth step the final prototype is in accordance with the objectives defined in the first step, the project is finalized. Otherwise, it is necessary to analyse the problems or improvements to be made and return to the step 3 to combat the problems occurring throughout the process.

1.5. Dissertation structure

The dissertation is composed by 5 sections:

- Chapter 1 (Introduction) – Where the dissertation is introduced.
- Chapter 2 (State of Art) – Where the investigation and knowledge about what are the best practises are, and reviewing some of the work was done and how can we improve and make adjusts in our project.
- Chapter 3 (Hardware System) – Where all the hardware is described and its implementation.
- Chapter 4 (Software Systems) – Where all the developed server scripts, embedded platform scripts, databases, API and additional software are described and a brief explanation of its implementation.
- Chapter 5 (Results and evaluation) – Where the tests results are represented.
- Chapter 6 (Conclusions and future work) – Where final thoughts of the development of the system are described and possible improvements or additions that can be made for future iterations.

1.6. Planning

For the dissertation to go in a smooth way, a table was developed as form of organization and generalized tasks that need to be completed for the work to be finalized. The figure 2 shows that organization of tasks.

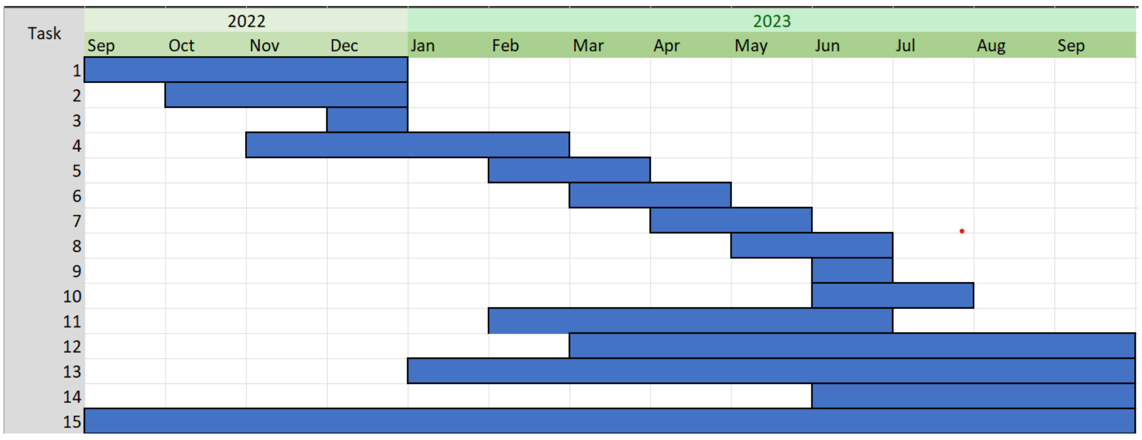


Figure 2: Task planning representation

1. Research of articles, papers, and exploration of the theme: This initial task started in final-September until mid-December, where I start to do some research relevant to the work.

2. Research of material to be used: A important task to know what material can be used to each monitoring. Material such as sensors, for data collection, and gateways for communication between node and server.

3. Research of databases: A initial database with crop related data, such as humidity, temperature and soil conditions, like NPK concentration and soil humidity, is needed to build a prediction model, so we can predict what crop is the best for a specific soil.

4. Writing initial report: In the beginning of February, an initial report must be done and to present, the same, to the juris.

5. System design based on system requirements: Firstly, after starting with the development of the ecosystem, we need to struturize the system and

averiguate what type of sensors and actuator will be used in the system. And then, test them individually, to check their functionalities.

6. Test sensors and actuators in a system: After checking the system design and test sensors and actuators individually, the next step is to test them in the system.
7. Module GPS: a subgoal from the thesis and a very important, because in this moment all the requirement data is being collected to make our database.
8. Software design and implementation: another subgoal and where we finish the development of hardware and software.
9. Writing of article: A task that is a requirement of this dissertation.
10. Writing a paper for conference: Another requirement for the thesis.
11. Process otimization: This taks will be continuous from the start of testing sensor and actuactors to the final system prototype. A very important task to optimize and to get the best effenciency possible.
12. Collecting and analyse data: Started collecting data after getting all sensors and actuators together, to get a vast database, and analyse the same.
13. Writing and development of the dissertation: This task is a process that must be continuous from February until the work is done, and this must be done until September.
14. Meeting with Smart Farm Colab and with Instituto Superior de Agronomia da Universidade de Lisboa: One goal is to have the ecosystem prepared in June, to meet with both organizations.
15. Meeting with Supervisors: This task will be continuous and from the beginning to the end of the dissertation. Important task, because without help from advisors, it would be a difficult work.

Chapter 2 – State of Art

2.1. Articles review

In this section of the state-of-the-art, we will summarize previous work that is closely related to the dissertation, in this case we will review 13 case-studies, where some similarities and differences between the previous work and our work, to then analyse improvements and adjustments.

In article [6] a series of calculations were made to optimize the layout area of 4 sensor nodes, to minimize the number of sensors use and the cost of the system.

A study done in India aimed to help small farmers with the development of a small kit with four devices such as an alarm, a humidity sensor, pest detection sensor, and a water control. This, as mentioned in article [7], to overcome some difficulties, such as the frequency of destruction of the fields due to the invasion of animals, the scarcity of water and pests. This study was a low-cost prototype that indeed helps farmers, however it doesn’t have features like npk and temperature sensor, measurements that are fundamental to crop growth.

The use of machine learning (ML) and IoT knowledge brings an advantage to help automatization traditional agriculture. In article [8], they propose training a logistic regression based model using Tensorflow that takes the condition of the soil, temperature, moisture and the soil that is to be grown as input, and predicts whether the conditions are optimal for a crop. This system gives farmers the possibility to control all the devices present on the field with the help of Internet. The control of actuators, with the help of a relay module used as a switch to control the farm equipment, using electromagnet, is done through a mobile application. The use of cloud was adopted, which was used like a database, to store data, and then be used for prediction. The article proposed the implementation of computer vision to identify the type of soil and cultivation in an image, and thus to identify pests and diseases.

In article [9], it was proposed to build a model with the objective of providing intelligent solutions for the agriculture community. It was divided into three parts, in the first part soil health is tested and weather conditions are predicted, then maintenance through factors such as soil humidity and temperature, atmospheric pressure and weather stations. Ultimately, the farmer has the choice of selling to distributors or changing the cycle and selling his crops directly to customers via a platform.

IoT systems involve the interconnection of various physical devices, sensors, and machines to exchange data and perform tasks. While wireless communication plays a crucial role in enabling IoT connectivity and flexibility, it introduces certain limitations compared to traditional wired connections. In article [10], the objective was to overcome this problem. The system of this article consists of nodes, gateways, servers, databases, and smartphones. To evaluate how IoT technology can overcome distance and place constraints of wired communication, they compare wireless communication such as Bluetooth and Low-Power Wide-Area Network (LPWAN) in this case LoRa communication, and a wired communication like RS485. The data from the nodes is send to a gateway, and the communication, from the gateway, is done through MQTT, as it is easy to build more gateways or servers, and the connection with advisors is possible only by IP address. The server captures data in real time through each communication network and records it in the database or transmits the command desired by the controller.

Are LPWAN technologies suitable for remote communication? According to article [11], these technologies are suitable for remote communication and among all, Long Range Wide-Area Network (LoRaWAN) is the most suitable for IoT system, expects lower power consumption, long battery life, highly mobile end devices with very high range coverage.

In article [12], a system is used to collect temperature, soil and level of precipitations and a DC Motor controlled by NodeMCU. This system is very useful for farmers, as they not only save water, but also the irrigation is automatic and controlled by the level of parameters like temperature, humidity, and rainfall.

Another article that proposes a smart watering system to enable real-time plant monitoring is article [13]. It is used the ESP32 microcontroller due to its compatibility with FreeRTOS and Arduino IDE. Sensors collect data and transmit to the microcontroller to determine if watering is required. However, the data collected from sensors are only the temperature and humidity. They propose in future research, the addition of another types of sensors, such as NPK, rain, and PH sensors.

A significant contributor to providing healthy, protein-rich products to humans at low cost is poultry farming. This has several associated problems, and one of them is diseases, which causes considerable losses for farmers. Article [14] proposed the installation of thermographic cameras, which can detect early infections in birds, as well as monitor floor temperature, air movement and the efficiency of the cooling system. However, this solution comes with a high installation cost.

In precision agriculture, forecasting weather conditions is essential to ensure the best possible efficiency. In article [15], an online forecasting service was considered, where data are collected from meteorological stations in real time. The effectiveness of this service is designed for forecasting frost and pests. The system warns farmers to protect their plantations from possible situations.

In the article [16] different types of drones were compared, the different benefits and tasks they bring in precision agriculture. However, vulnerability to cyber-attacks, such as false information and the capture of falsified data, means that this solution is still not completely secure.

After summarising some articles, we can see that our work as a lot of similarities with previous work, however, our work will make some changes to improve the problem, by merging some articles and try to continue their work.

2.2. Concepts Review

After reviewing and summarize several articles, in this section will be provided background information and context for the dissertation being conducted.

2.2.1 IoT architecture

As mentioned in the introduction of the dissertation, IoT technology has a wide variety of applications and is growing fast. Depending upon the different application areas of IoT, it works according to the designed/development. But it has not a standard defined architecture, this change for the application it requires. However, there is a basic process flow on which IoT is built.

IoT systems has different types of architecture, like three-layer, four-layer, etc. We will focus on the four-layer architecture that is designed to be flexible, scalable, and resilient to changes, and it allows for easy integration of new devices, protocols, and services [17]. The four-layer architecture is composed of sensing layer, network layer, data processing layer, and application layer, represented in figure 3.

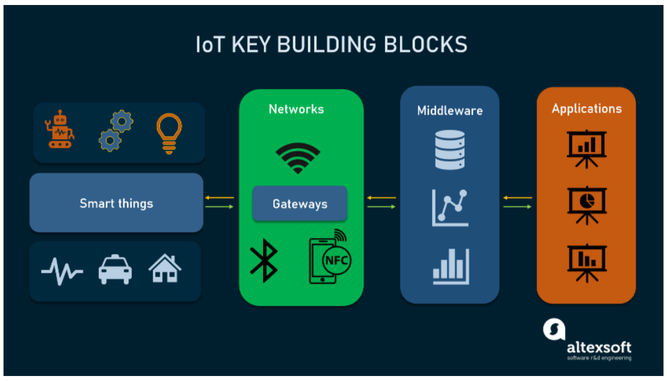


Figure 3: IoT four-layer architecture [18]

2.2.1.1 Sensing Layer

Sensing layer or Perception layer is the first layer of the IoT architecture. This layer encompasses all the devices capable of capturing, processing, and communicating data through the Internet. Layer of the smart devices or IoT devices, like sensors and actuators [19].

A sensor is a device that detects and responds to some type of input from the physical environment. There are various of types of input, like light, heat, motion, moisture, temperature, pressure, and a lot more. The output is generally a signal that is converted to a human-readable display at the sensor location or transmitted electronically over a network for reading or further processing. They play a pivotal role in the IoT. They make possible to create a ecosystem for collecting and processing data to monitored, managed and controlled more easily and efficiently [20]. In our work will be used a variety of sensors, like:

- Temperature sensors: Device, typically, a thermocouple or resistance temperature detector, that provides temperature measurements through an electrical signal. The working of a temperature meter depends upon the voltage across the diode. The two main types of temperatures sensors are: Contact and Non-Contact, DS18B20, and DHT22, respectively [21].

- Humidity sensors: Electronic devices that measures the humidity in its environment and converts into an electrical signal. Vary widely in size and functionality. Can be divided into two groups, as each one uses a different method to calculate humidity: relative humidity, calculated by comparing the live humidity reading at a given temperature to the maximum amount of humidity for air, and absolute humidity, is measured without reference to temperature [22].
- Light sensors: A type of photodetector that detects light. Common types of light sensors are photodiodes, photoresistors, phototransistors and photovoltaic light sensors. Photodiodes convert light into an electrical current. Photoresistors are passive devices that decrease resistance in proportion to the amount of light received. Phototransistors switch or amplify signals similarly to regular transistors, with the current applied to the terminals being created from exposure to light. Photovoltaic convert light into electricity in a process know as energy harvesting. These sensors work by the photoelectric effect [23].
- PH sensor: Capable of measuring alkalinity and acidity in water and other solutions. Can ensure the safety and quality of products and processes that occur in wastewater. There are many different types of pH sensors, combination pH sensors, laboratory pH sensors, process pH sensors and differential pH sensors. Reliable pH sensors are extremely important for plant optimization [24].
- Time-domain reflectometry (TDR) sensor: Non-destructive method to determine the water content of soils and other porous media. Electronic device that works on radar based on transmitting signals into the medium and collecting reflected signals. TDR determines dielectric constant and consequently permittivity and water content of soil, via wave propagation transmitted by two parallel embedded metal probes [25].

- Camera: For health of plants, there are two types of cameras that can be used: multispectral and modified cameras. Multispectral cameras are designed for fidelity to be as accurate as possible when measuring the reflectance of objects, can come with sun irradiance sensors. Modified cameras are normal cameras but by carefully changing the filters, it is possible to get wide band infrared measurements [26].

- Nitrogen (N), Phosphorus (P) and Potassium (K) sensors: NPK sensors are suitable for detecting the content of nitrogen, phosphorus, and potassium in the soil. Helps in determining the fertility of the soil. The knowledge of the soil composition can help us learn about nutritional deficiency, and so we can evaluate what type of crop can be use in a specific soil or we can stabilise the condition of the soil for a specific crop [27].

- Sensors for pest, diseases, and insects: For pest and insects, farmers can use various sensor from simple to complex work. Low-power image sensors are a wireless autonomous monitoring system based on a low-cost image sensor. These sensors periodically capture images and can send it remotely. Acoustic Sensors which work by monitoring the noise level of the insect's pests. These sensors can be useful by utilizing a threshold to determinate when a noise level from an insect, an information is sent to the computer. Crop diseases can significantly reduce the yield, endangering the production and health of crops. Farmers can use modern farm measures including direct and indirect disease identification method. Thermography sensors measure the differences in surface temperature of the plant leaves and canopy, captures infrared radiation emitted from the plant surface. In figure 4, we can see an image capture by these sensors. They can even detect the changes due to the disease before it even appears. Another method is the fluorescence disease method, that measures the chlorophyll fluorescence on the leaves and measures the incident light and the change in fluorescence parameters. By doing so, can be detected the pathogen presence. Finally, a non-optical sensor uses to determine volatile chemical compounds release by infected plants, this sensor is used in the gas chromatography disease detection method [28].

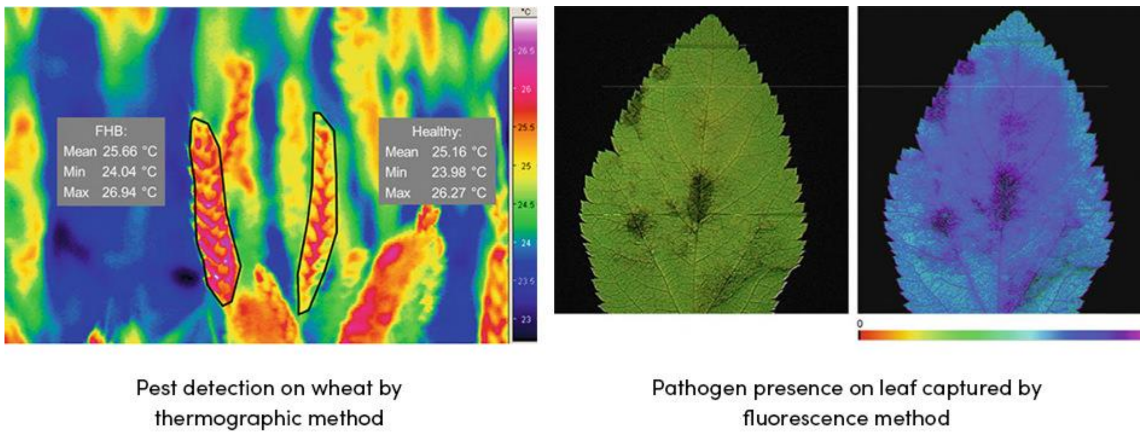

FHB:
Mean 25.66 °C
Min 24.04 °C
Max 26.94 °C
Healthy:
Mean 25.16 °C
Min 23.98 °C
Max 26.27 °C
Pest detection on wheat by thermographic method
Pathogen presence on leaf captured by fluorescence method

Figure 4: Image from a thermocography sensor [30]

- GPS sensors: Consists of surface mount chip which processes signals from GPS satellites using a small rectangular antenna, often mounted on the top of the GPS ship. For this type of sensor, its needed four satellites to determinate a position on the earth in three-dimensional space. Each one of them maintain precise time and a pseudo random number generator. This number is used to distinguish signals and calculate receiver's distance to each of these satellites by comparing arrival time. In figure 5 we can see an example [29].

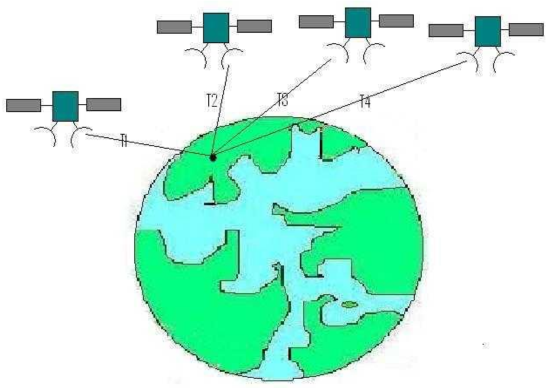


Figure 5: GPS sensor working [31]

Actuators are devices that convert an electrical signal into a physical action and can be used to control various farming operations. Like sensors, they are playing a pivotal role in the IoT. The three main types of actuators are [30]:

- Pneumatic (air pressure): Use compressed air or pressurized gas to create a controlled movement.

- Hydraulic (fluid pressure): Use fluid pressure to facilitate mechanical movement. Hydraulic power is controlled by the amount of fluid in a cylinder. Pressure is created by increasing fluid and lessened by decreasing fluid.
- Electric: Convert energy from an electrical power source into mechanical energy.

Actuators	Advantages	Disadvantages
Pneumatic	Versatile; Customizable; Cost-effective; Safe.	Compressor must run continuously.
Hydraulic	Very high force and power.	Volatile nature; Requires highly trained mechanics.
Electric	Easy to maintain; High level of precision.	Less cost effective than pneumatic; Strict working environment; No fail-safe position if power is lost.

Table 1 : Advantages and disadvantages of the three main types of actuators [32]

In our work, we will focus on the hydraulic ones, because they will be important to control the humidity values. These actuators are water pumps.

2.2.1.2 Network Layer

Network layer is also known as transmission layer. Acts like a bridge between sensing layer and support layer. It carries and transmits the information collected from the physical objects through sensors. This layer can be either wireless or wire based. Responsible as well for connecting the smart things, network devices and networks to each other [31].

IoT protocols are modes of communication that protect and ensure optimum security to the data being exchanged between connected devices. Devices are typically connected to the Internet via an Internet Protocol network. However, devices such as Bluetooth and Radio Frequency identification (RFID), can be connected locally. Although, connecting through IP are comparatively complex, required increased memory and power. On the other hand, locally connected devices demand comparatively less power and memory but have a range limitation [32]. So, protocols can be long range or shor range.

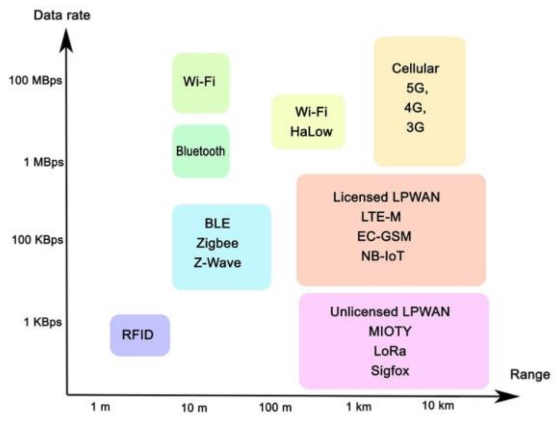


Figure 6: IoT communication protocols [33]

One of them is LoRa. The term LoRa stands for long range. It is a wireless radio frequency technology introduced by a company called Semtech. This LoRa technology can be used to transmit bidirectional information to long distance without consuming much power. This property can be used by remote sensors which have to transmit its data by just operating on a small battery. It can achieve a distance from 15 kilometres up to 20 kilometres and can work on battery for years. However, to have this long distance, this compromises the bandwidth, and this protocol operates on very low bandwidth, with a maximum of 5500 bits per second, which means that it will be able to send only small amount of data. Cannot send audio or video and works great only for transmitting less information, like sensors.

LoRaWAN protocol is a Low Power Wide Area Networking communication protocol that function on LoRa. Just like LoRa [34]:

- Long range communication, but up to 10 kilometres in line of sight.
- Long battery duration of up to 10 years.
- Low cost for devices and maintenance.
- License-free radio spectrum but region-specific regulations apply.
- Low power but has a limited payload size of 51 bytes to 241 bytes depending on the data rate.

Just like LoRa, this protocol is an LPWAN. The technology can be used for application requires low power consumption, long-distance communication, and long battery life. The advantage of this protocol is that has good coverage capacity and has a distance coverage of around 10 kilometres [35].

Wi-Fi uses a star network topology, and the access point can be used as a gateway to the Internet. Each access point can connect to 250 devices. Operates 2.4G Hertz, but the distance coverage is about 50 meters, unlikely NB-IoT and LoRa [36].

Bluetooth enables sending video and audio, one characteristic that LoRa can’t give. Is a short-range wireless technology that uses short-wavelength, ultrahigh-frequency radio waves. It has most commonly been used for audio streaming, but it has also become a significant enabler of wireless and connected devices. This one is a go-to for both personal are networks and IoT deployments [37].

SigFox is a long range cellular wireless communication that offers custom solutions primarily for low-throughput IoT, by availing its end-to-end IoT connectivity services. The end devices use binary phase-shift keying (BPSK) modulation to connect to the base stations. This protocol offer low power consumption ensures that remote devices have long battery life or maintenance. Enables long distance communication, uses a cellular style to enable remote nodes to use internet to communicate with base stations [38].

2.2.1.3 Data Processing Layer

The data processing layer or processing layer is the brain of the IoT ecosystem. Data is analysed, pre-processed, and store here, where it is accessed by software applications that both monitor and manage the data, as well as prepare further actions. In this layer, we will also have the databases [39]. In our work, to predict certain parameters, we will need a testing database in the beginning, and while producing the work, we will manage to get new data and build our database.

2.2.1.4 Application Layer

The application layer involves decoding IoT data and compiling them into summaries that are easy for humans to understand, such as graphs and tables. Programs for device control and monitoring, as well as process control software, this are typical examples of IoT applications [40].

In our work, we will build a mobile application that can control, visualize, and represent data in map-level. The data will be about the crops, weather, and GPS.

2.2.2 Precision agriculture

PA helps to improve the livelihood of the farmers by automating and optimizing all feasible agricultural parameters to enhance the agricultural productivity and cultivation. IoT sensors help to measure soil quality, weather conditions, moisture level, and finally optimize these parameters to increase the yield. This application consists of four components: weather monitoring, soil conditions monitoring, disease monitoring and irrigation monitoring.

2.2.2.1 Weather monitoring

Nowadays, weather forecasting is uncertain and inaccurate. Weather monitoring is utilized to monitor the constantly changing climatic and weather conditions over-regulated areas in agriculture. Parameters like temperature, humidity, wind, and air pressure are collected using sensors, passing it throught the microcontroller and to be used to representate or even controlled based on the level of the parameters. This data will be mapped and used to predict the next weather parameters to improve the agricultural growth [41].

2.2.2.2 Soil contents monitoring

This is one of the most demanding practices in agriculture field. Soil condition can be defined as the capacity of a soil to function, to sustain biological productivity, maintain environmental health, and promote plant, animal, and human health. The soil characteristics vary from place to place, and different characteristics are needed for diferentes crops. Growing rice is not the same as growing tomatoes. So, parameters like NPK, humidity, temperature, and pH need to be monitorized to optimize the production and health of crops [42].

2.2.2.3 Diseases monitoring

Image based monitoring, this practice can be autonomous or manual. This brings farmers the opportunity to see an image and diagone either if it is a disease or not. And autonomously with the help of ML techniques.

2.2.2.4 Irrigation Monitoring

The possibility to farmers control water irrigation or even an autonomous water system irrigation, with the help of sensors that collect weather conditions and soil conditions, can help them reduce desnecessary water waste and irrigation cost [43].

Chapter 3 – Hardware

3.1 Hardware Overview

The architecture of the system includes the sensing part is connected to Arduino Uno that presents a LoRa shield, and the NPK sensor is connected to The Things Gateway, with the help of TTN. After collecting necessary measurements, the data are sent via LoRa to the dragino gateway and via LoRaWAN to The Things Gateway, for later, communication with the server and database, this is explained in the software section. The use of both gateways will be explained later aswell.

The system architecture is represented in figure 7:

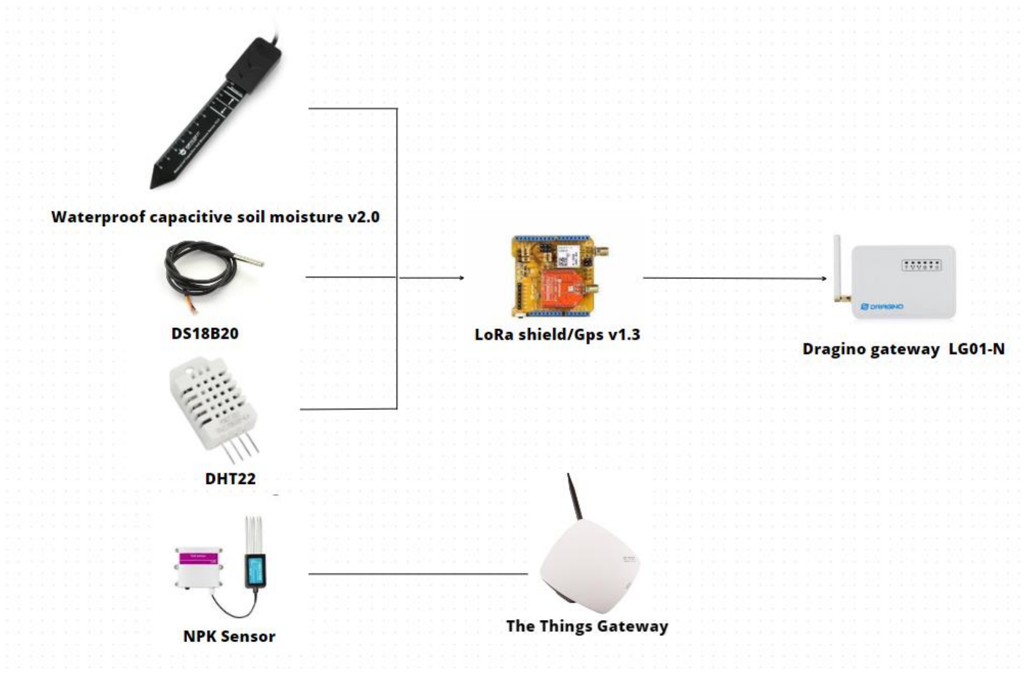


Figure 7: Hardware system architecture

The schematic of the implemented architecture is presented in figure 8:

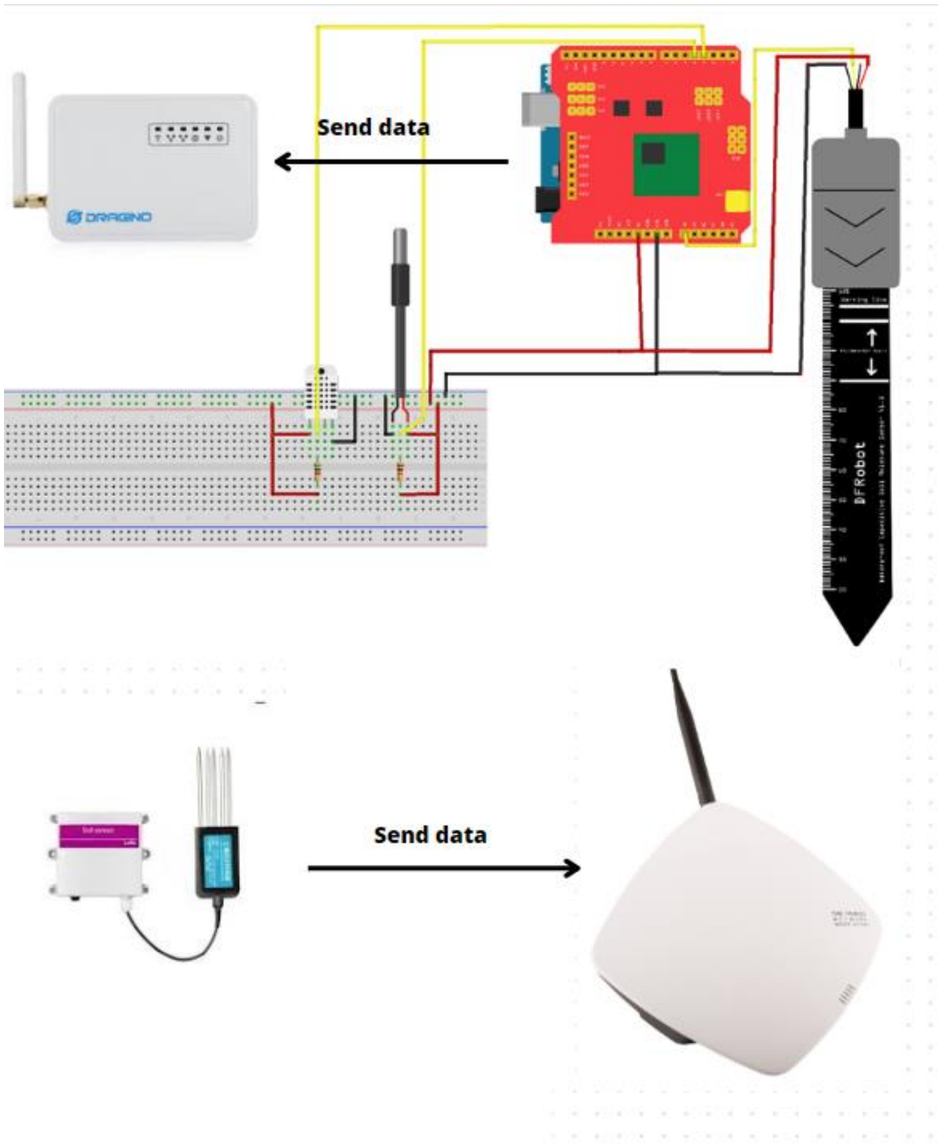


Figure 8: System communication between sensors and gateways

We can see in figure 8 that it was used a LoRa shield with Arduino Uno and the pins used are: two digital pins (3 and 4), one analog pin (0), ground and 5V power. The capacitive soil moisture sensor is connected to the analog pin, to the 5V pin and to the ground. The DHT22 sensor is connected to the ground and power pins, however it needs a 5K ohms resistor, that helps to maintain the data line at a know voltage level when the sensor is not actively driving it low, and this will be connected to digital pin 3. Finally, we have the ds18b20 sensor, that has a 10K ohms resistor between data and power, connected to power and ground pin and the data to digital pin 4. This will be all connected to the dragino gateway.

The other gateway is the The Things Gateway and receives the npk sensor.

3.2 Waterproof capacitive soil moisture v2.0

The soil moisture sensor has got four wires (Vin, output and two grounds), but only three are used for output and power (Vin, output, and ground). It is powered by the Arduino Uno and LoRa shield with 5V, and the output is connected to the same. The output voltage range is from 0 to 3V, which means, if it is used an Arduino UNO, the digital range is from 0 to 660.

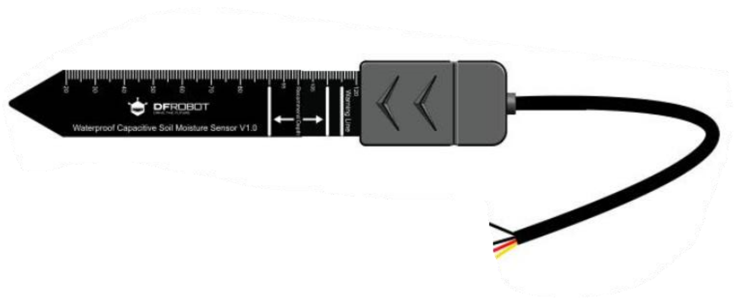


Figure 9: Waterproof capacitive soil moisture v2.0 [44]

3.3 Waterproof Digital Thermometer DS18B2

Temperature sensor that provides reading from -55ºC to 125ºC with a +/-0.5ºC accuracy between -10°C e +85°C. Uses the 1-Wire protocol for data communication which means that uses only 1 wire to exchange data with the Arduino Uno, and using the microcontroller power, so we don't need an external battery.

Figure 10: Waterproof Digital Thermometer DS18B20 [45]

3.4 Temperature-humidity sensor DHT22

A basic sensor, that uses capacitive humidity sensor and a thermistor to measure the surrounding air. Has four pins, and we will only use three pins (Vin, output, and ground), and additionally a resistor of 5K ohms, to prevent voltage fluctuactions for signal stability. Temperature values goes from -40ºC to +80ºC and humidity from 0% to 100%.

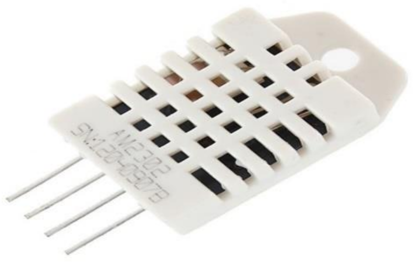
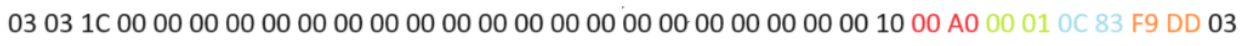


Figure 11: Temperature-humidity sensor DHT22

3.5 NPK sensor

LoRaWan Soil NPK sensor powered by a long-life battery. NPK values are measure in mg/kg and its range goes from 0 to approximately 1999. Works only with LoRaWAN communication between frequencys 915MHz to 930MHz. It collects data every 30 minutes, however for testing purposes, we turn on, wait a few seconds to send the data via LoRa and turn of, so we can force it to send data everytime we want. It uploads data as the figure 12 shows, where the hexadecimal colored red is the nitrogen concentration, the green, phosphorus concentration, the blue, potassium concentration and finally in orange the battery data. Later, we will demonstrate how we convert the data from hexadecimal to understandable numbers.



```
03 03 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00 A0 00 01 0C 83 F9 DD 03
```

Figure 12 : Data upload from sensor

Figure 13: NPK sensor

3.6 The Things Gateway

The Things Gateway comes to help the dragino gateway gaps, where the npk sensor cannot be connected to the dragino, duo to be outdated to The Things Network. This gateway enables devices to connect to the Internet, with an easy to connect process and with a capacity to serve a lot of nodes. It operates at 868MHz in the Europe.

Figure 14: The Things Gateway [46]

3.7 Dragino LoRa/GPS shield

The LoRa/GPS shield as the same functionality as the LoRa shield, with an additional feature, that he can collect GPS data, however, we didn't have the other shield (dragino yuno) or another microcontroller to use this functionality. However, the GPS from the shield is not being used in our project, this will be presented in the Arduino, as a shield, and used to send the data via LoRa to the dragino gateway.

Figure 15: Dragino LoRa/GPS shield [49]

3.8 Dragino Gateway LG01-N

LG01-N can be used to provide a low cost IoT wireless solution, allow us to send data and reach extremely long ranges at low data-rates. Let us use LoRa bridge wireless to an IP network via WiFi, Ethernet, Or 3G/4G cellular via optional LTE module. It can support multiple working mode such as MQTT, TCP/IP client mode to fit different requirements for IoT connection. However, this dragino has some limitations and can't be used with TTN, because the version we have, dragino LG01-N, is designed for private LoRa protocol, and not recommended for LoRaWan use.

Figure 16: Dragino Gateway LG01-N [50]

3.9 Arduino UNO

The Arduino UNO is based on the ATMega328P microcontroller. It is part of the Arduino platform, an open-source hardware and software ecosystem designed for creating simple projects. Has a total of 14 digital input/ouput pins. Power and programmed through a USB connection. Programmed using the Arduino IDE, which uses a simplify version of C/C++ programming language and has a vast of community of users and libraries to help make it easy of use.

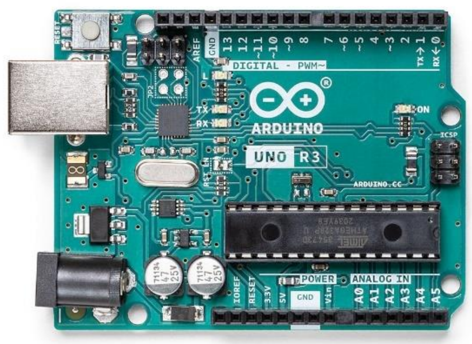


Figure 17: Arduino UNO [51]

Chapter 4 – Software

4.1 Software Overview

This chapter presents the system software describing the backend and frontend functionality within the Arduino, Dragino, remote server, databases, and websites.

The followed figure is a representation of the software system architecture:

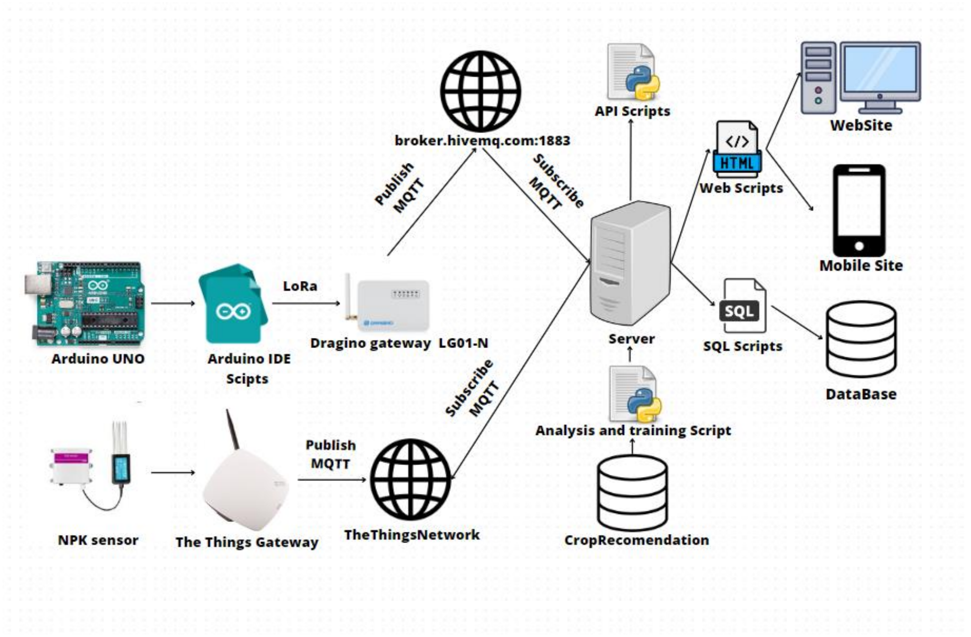


Figure 18: Software system architecture

In figure 18, on the Arduino UNO level, it uses a script for collecting the data from the sensors, some conversations for better understanding and for sending that data, via LoRa, to the dragino gateway. At the same time, we have the npk sensor sending data to The Things Gateway and some decoding made. Then, both gateway are configurated to publish the data received from the Arduino and npk sensor, using the MQTT communication protocol, in a free public MQTT broker from HiveMQ (free broker: broker.hivemq.com with the port: 1883) and a MQTT integration on TTN. The server will request the data by subscribing to the specific topic, for later to be use in the websites and database. Initially, the database CropRecomendation is used for better understanding, how the data will be analysed and what model is the best for our implementation. After choosing the best model, it will be used for optional prediction on the website. The website and the mobile site are programmed with html, css and javascript, and has three functionalities: project presentation, prediction and dashboard for data visualization and analysis. APIs are used so the websites and database fetch for the data and use it accordingly. Finally at the same time, the data is saved in the database created by us, and this data will be used in the websites to visualization and analysis.

4.2 Programming Languages

The Arduino UNO, websites, APIs, and databases have few scipts running and software to allow the system to work accordingly. The used software platforms were Python, HTML, CSS, Javascript, Arduino IDE, SQLite and Jupyter.

4.2.1 Python

Python is one of the most used programming languages, not only is a widely used high-level programming language for general programming, is also easy to learn and emphasizes on code readability which makes it visually appealing, and it can be programmed in a few lines of code. This language was selected due to having a lot of possibilities and solutions for the project, such as APIs requests, fetching data and database analysis. All of the functions mention, from APIs to communication between gateway and server, can be found in annex A.

4.2.2 HTML

HTML is the standart markup language for Web pages. Just like Python, is easy to learn and it is the foundation of most web content providing the structure and layout for displaying information on the Internet. This language combining with CSS and javascript make a structurized, styled, and functional website. The dashboard built is founded in annex C, using some sections for better organization and some cards for a better visualization.

4.2.3 CSS

CSS, or Cascading Style Sheets, is a stylesheet language used for describing the presentation and formatting of HTML documents. CSS allows developers to control how the visual appearance of the website will look like, by changing the styles of HTML documents. The CSS used in this project can be founded in annex D, this to improve the design of the website.

4.2.4 Javascript

Javascript, like python, is one of the most popular programming languages, easy to use and focuses on web development. It is a client-side scripting language, meaning that it runs in the web browser of the user, enabling dynamic and interactive behavior on websites. This programming language was fundamental, by not just creating a visualization dashboard to check and analyze sensor data, but also to present the project, the objectives and sensors used. The scripts using javascript can be founded in annex B and will help the html and css to have a responsive website.

4.2.5 Arduino IDE

The Arduino IDE is a software application used for programming and developing applications for Arduino microcontrollers. It is an open-source hardware and software platform, easily to use, by the vast libraries and community users. Used for combining the sensors with the server, by collecting and sending the data, through a simple process.

4.2.6 SQLite

SQlite is a lightweight, serveless, and open-source relational database, this means it does not require a separate server process and runs as a python library. Commonly used in embedded systems, mobile applications and small to medium sized web applications, just like what we needed. The database is simple, not having a relational aspect, and used for data visualization and analysis.

4.2.7 Jupyter

Jupyter notebook is an open-source web application that allows to create and share documents that contain live code, equations, visualizations, and narrative text. One of the most popular choices among data scientist and engineers for data analytics and machine learning. IPython notebook is the combination of Python and Jupyter, it is an interactive computation environment, which we can combine code execution, rich text, mathematics, plots and rich media. It was used for initial data analysis of the database and for training the best model for the project. The jupyther notebook was implemented to a Python file for better understanding and visualization and can be founded in annex E.

4.3 Database analysis and model training

To start our software development, we first searched for a database related to our project, for analysis and model training, to see what the best factors are for crops. It was found a few databases that contain the essential characteristics for crop health and growth, such as, npk, temperature and humidity. The Crop Recommendation Dataset was choosed, that has 2200 records that contains augmenting datasets of rainfall, climate and fertilizer, this is humidity, temperature and NPK concentration, data available on India [47]. It also includes 22 different crops, such as apples, bananas, mangos, mothbeans and coffee. We used IPython to analyse the data and for model creation, testing and evaluating what has the more accuracy for our project. For better understanding of this database, we analyze the data contained, by checking the npk values for each crop.

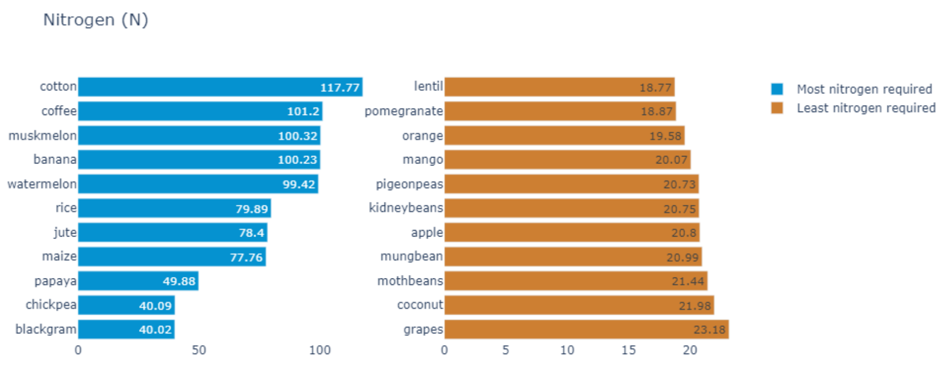


Figure 19: Nitrogen concentration

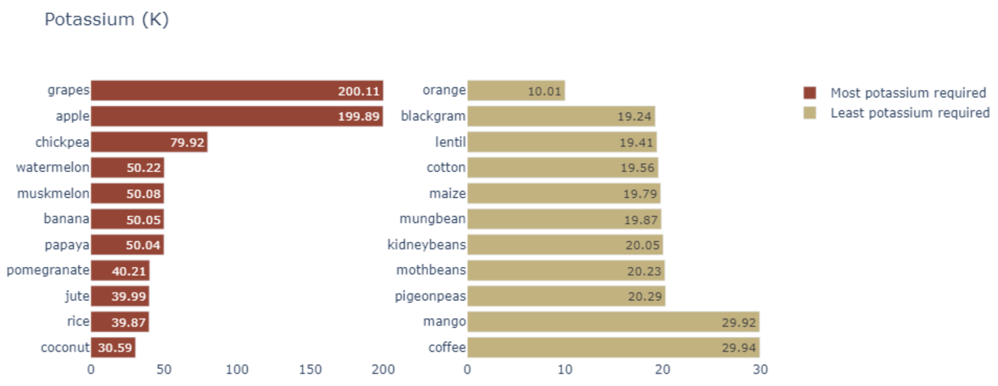


Figure 20: Potassium concentration

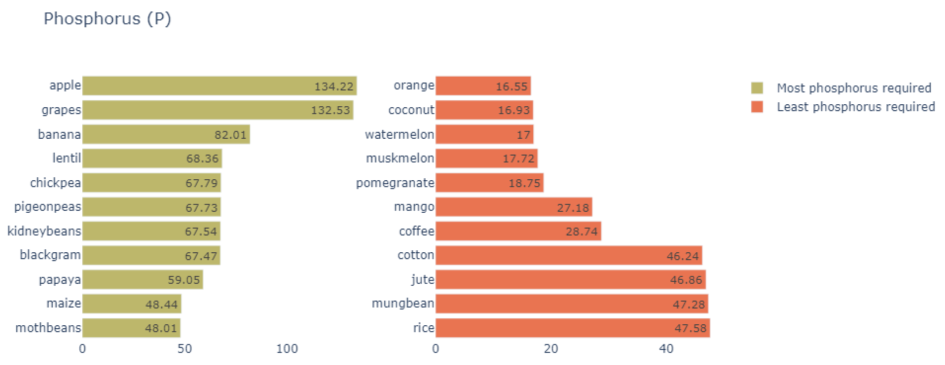


Figure 21: Phosphorus concentration

In figure 19, 20 and 21, we have the value of NPK for every crop. For example, for apple, it needs 20.8 mg/kg nitrogen, 199.89 mg/kg potassium and 134.22 mg/kg phosphorus concentration. We can calculate the percentage of each concentration in the next equation, where X is the choice chemical.

$$\%\ de\ X = \frac{X}{N+P+K} * 100\ \% \qquad (1)$$

And in this case, for apple, the concentration is approximately 5,9 % nitrogen, 56,3 % potassium and 37,8 % phosphorus.

The next step is to train various models, and check who brings a better performance for our characteristics. Some values from the dataset are not valuable for our project, like rainfall and ph, so the features chosen for the train model are npk, climate temperature and humidity.

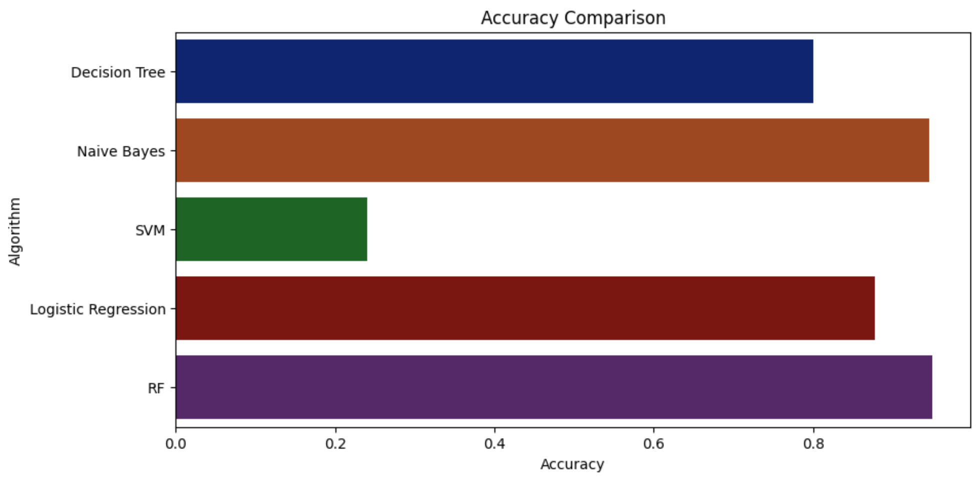


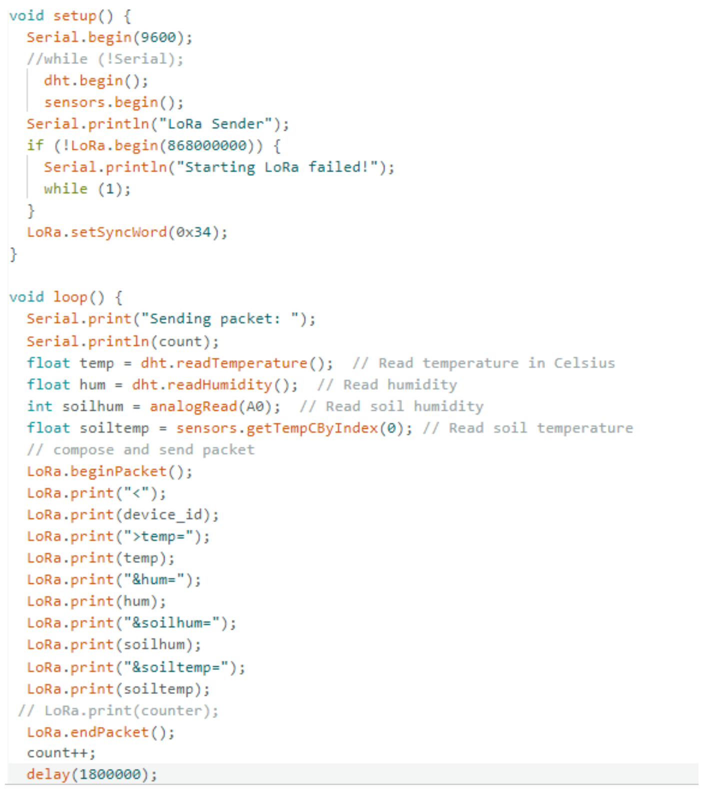
Figure 22: Models accuracy

To get the results in figure 22 we used sklearn library, that has one function to train each mode. We train the models with 80 for train and 20% for test of the dataset, and Naïve Bayes and Random Forest are the best models for the chosen features, however Random Forest has 95 % accuracy and Naive Bayes has 94,5 %. So, it was chosen the Random Forest algorithm to predict what crop is the best for a certain soil.

4.4 Data collection and communication

4.4.1 Collect and send sensor data via LoRa

In this part, we collect the data from the sensors and send the data via LoRa to the dragino gateway.



```
void setup() {
  Serial.begin(9600);
  //while (!Serial);
    dht.begin();
    sensors.begin();
  Serial.println("LoRa Sender");
  if (!LoRa.begin(868000000)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  LoRa.setSyncWord(0x34);
}

void loop() {
  Serial.print("Sending packet: ");
  Serial.println(count);
  float temp = dht.readTemperature();  // Read temperature in Celsius
  float hum = dht.readHumidity();  // Read humidity
  int soilhum = analogRead(A0);  // Read soil humidity
  float soiltemp = sensors.getTempCByIndex(0); // Read soil temperature
  // compose and send packet
  LoRa.beginPacket();
  LoRa.print("<");
  LoRa.print(device_id);
  LoRa.print(">temp=");
  LoRa.print(temp);
  LoRa.print("&hum=");
  LoRa.print(hum);
  LoRa.print("&soilhum=");
  LoRa.print(soilhum);
  LoRa.print("&soiltemp=");
  LoRa.print(soiltemp);
 // LoRa.print(counter);
  LoRa.endPacket();
  count++;
  delay(1800000);
```

Figure 23: Collect and send data via LoRa

As we can see in figure 23, the script starts by looking for the gateway specific frequency, then data is collected. To collect data from DHT22, it is used the dht library and initialialize the dhttype as DHT22. From ds18b20, one wire and dallas temperature libraries are fundamental to get the data from the sensor, because this sensor works as a one-wire temperature sensor, and finally the capacitive soil moisture, it reads analog voltage and transforms into digital, as said in chapter 4. After all the data is collected from the sensor it is send, with an interval of 30 minutes, to the dragino gateway, with the help of the LoRa library.

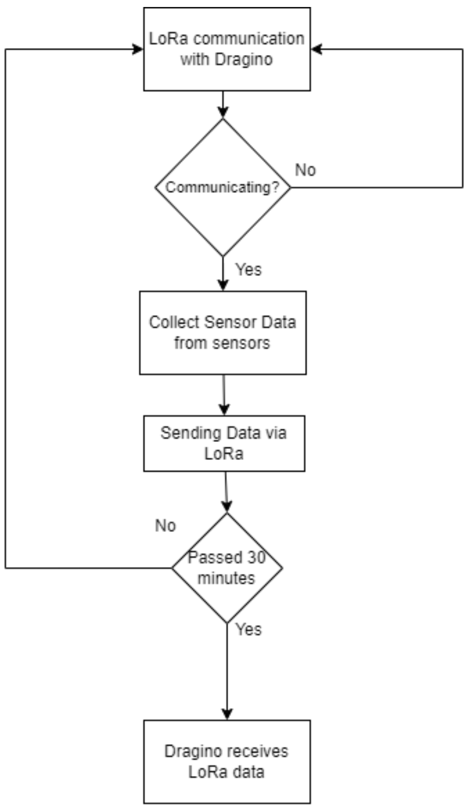
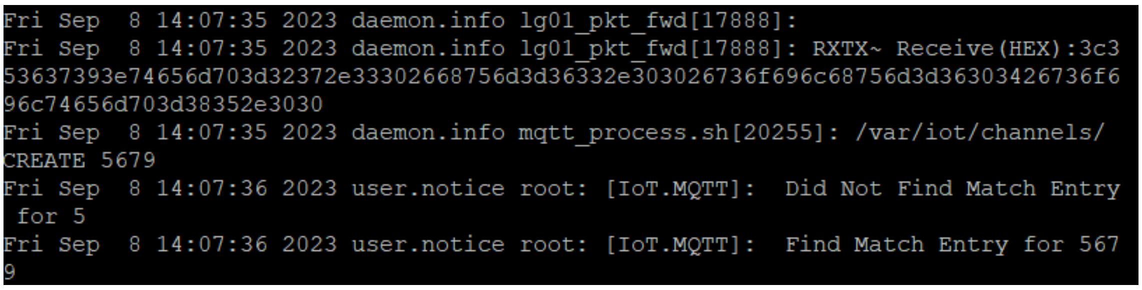

LoRa communication with Dragino
Communicating?
No
Yes
Collect Sensor Data from sensors
Sending Data via LoRa
No
Passed 30 minutes
Yes
Dragino receives LoRa data

Figure 24: Flowchart arduino script

As the figure 24 shows, first we will try establishing a connection between the dragino gateway and the Arduino UNO + LoRa/GPS shield. After, the connection is established, we start collecting data from the sensors, in this case, ds18b20, moisture sensor and dht22, and the data s send via LoRa to the dragino. After 30 minutes and if the communication between the dragino and Arduino is still alive, we get another collection of data from the sensors.

The data is then received by the dragino gateway, as figure 24 shows, in channel 567, that is device_id variable in the Arduino script. After receiving the data, it will then make a MQTT publish.



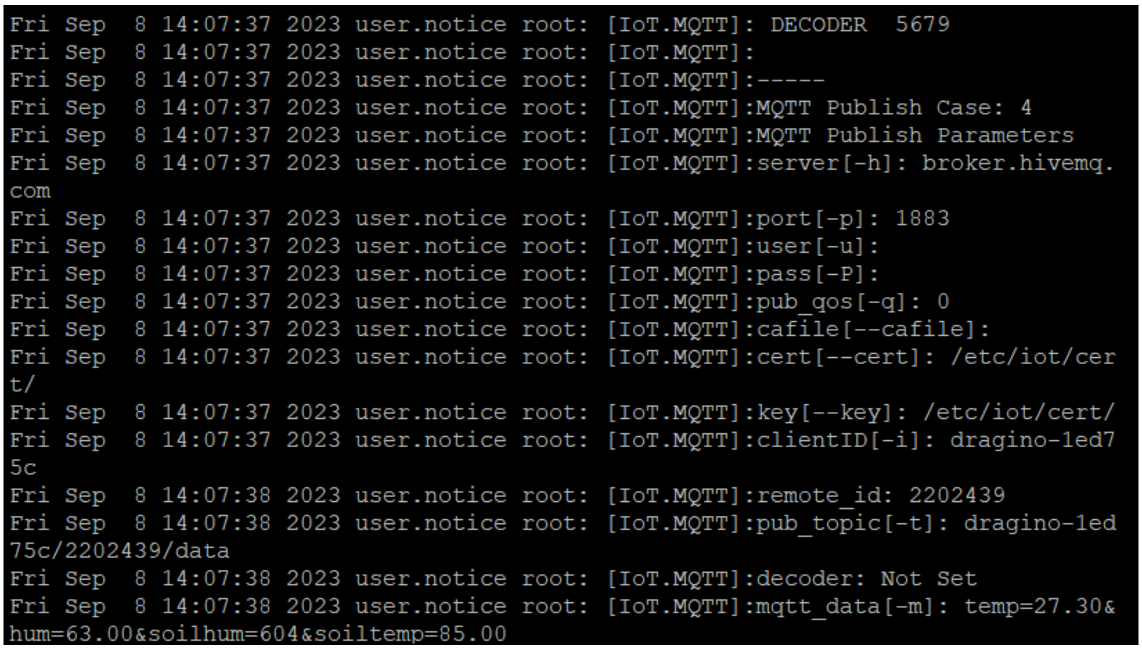
```
Fri Sep  8 14:07:35 2023 daemon.info lg01_pkt_fwd[17888]:
Fri Sep  8 14:07:35 2023 daemon.info lg01_pkt_fwd[17888]: RXTX~ Receive(HEX):3c3
53637393e74656d703d32372e33302668756d3d36332e303026736f696c68756d3d36303426736f6
96c74656d703d38352e3030
Fri Sep  8 14:07:35 2023 daemon.info mqtt_process.sh[20255]: /var/iot/channels/
CREATE 5679
Fri Sep  8 14:07:36 2023 user.notice root: [IoT.MQTT]:  Did Not Find Match Entry
 for 5
Fri Sep  8 14:07:36 2023 user.notice root: [IoT.MQTT]:  Find Match Entry for 567
9
```

Figure 25: Data received in dragino gateway

4.4.2 MQTT protocol

In this part of the project, it was giving a lot of options to where to use LoRaWan or MQTT, but unfortunately, the dragino gateway given, was outdated with several softwares, like ThingSpeak and The Things Network, because the dragino LG01-N is designed for private LoRa protocol, and not recommended for LoRaWan use. So, the unic solution is via MQTT. And again, we couldn't neither use ThingSpeak and The Things Network, however it will be implemented another gateway for collecting data from the npk sensor, that uses the LoRaWan communication, and The Things Network will be used. And it could not be possible to create a broker, therefore it was decided to use a public MQTT broker, like HiveMQ. The gateway was configurated to publish data in the server "broker.hivemq.com", with the port "1883" and with the topic "dragino-1ed75c/2202439/data", this would contain the data send from the Arduino script.

An example of a MQTT publish can be seen in figure 25. Later in this chapter, it will be explained when the data will be subscribed.

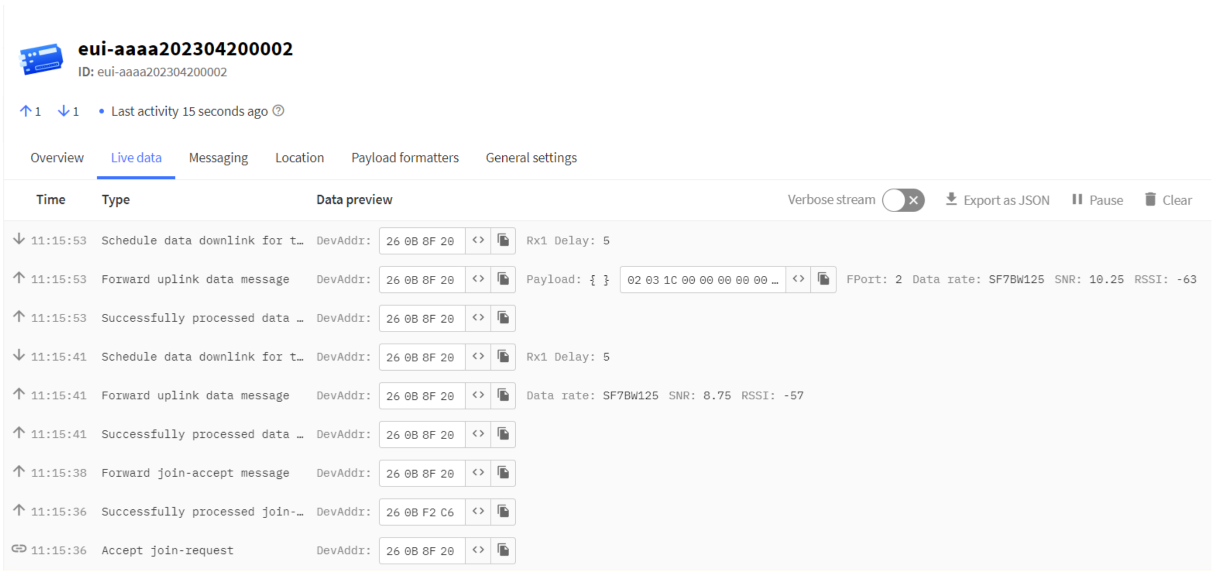


```
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]: DECODER  5679
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:-----
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:MQTT Publish Case: 4
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:MQTT Publish Parameters
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:server[-h]: broker.hivemq.
com
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:port[-p]: 1883
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:user[-u]:
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:pass[-P]:
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:pub_qos[-q]: 0
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:cafile[--cafile]:
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:cert[--cert]: /etc/iot/cer
t/
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:key[--key]: /etc/iot/cert/
Fri Sep  8 14:07:37 2023 user.notice root: [IoT.MQTT]:clientID[-i]: dragino-1ed7
5c
Fri Sep  8 14:07:38 2023 user.notice root: [IoT.MQTT]:remote_id: 2202439
Fri Sep  8 14:07:38 2023 user.notice root: [IoT.MQTT]:pub_topic[-t]: dragino-1ed
75c/2202439/data
Fri Sep  8 14:07:38 2023 user.notice root: [IoT.MQTT]:decoder: Not Set
Fri Sep  8 14:07:38 2023 user.notice root: [IoT.MQTT]:mqtt_data[-m]: temp=27.30&
hum=63.00&soilhum=604&soiltemp=85.00
```

Figure 26: MQTT publish data

4.4.3 The Things Network

As said in this chapter, the dragino gateway could not connect to The Things Network, because this is used for private LoRa protocols, and our npk sensor needs to communicate via LoRaWan, so we can collect data from it. This can be done using The Things Gateway, and in TTN we can create an application for our end device, in this case the npk sensor. After registering the end device, it is time to check what is the data receive from it.



Time	Type	Data preview
↓ 11:15:53	Schedule data downlink for t…	DevAddr: 26 0B 8F 20 Rx1 Delay: 5
↑ 11:15:53	Forward uplink data message	DevAddr: 26 0B 8F 20 Payload: { } 02 03 1C 00 00 00 00 00 … FPort: 2 Data rate: SF7BW125 SNR: 10.25 RSSI: -63
↑ 11:15:53	Successfully processed data …	DevAddr: 26 0B 8F 20
↓ 11:15:41	Schedule data downlink for t…	DevAddr: 26 0B 8F 20 Rx1 Delay: 5
↑ 11:15:41	Forward uplink data message	DevAddr: 26 0B 8F 20 Data rate: SF7BW125 SNR: 8.75 RSSI: -57
↑ 11:15:41	Successfully processed data …	DevAddr: 26 0B 8F 20
↑ 11:15:38	Forward join-accept message	DevAddr: 26 0B 8F 20
↑ 11:15:36	Successfully processed join-…	DevAddr: 26 0B F2 C6
11:15:36	Accept join-request	DevAddr: 26 0B 8F 20

Figure 27: Messages from npk sensor

As we can see in figure 27, the sensor sends 3 messages. A join message and two uplink messages. Every message will then be sent to the server via MQTT, to later we get the useful data for our project.

4.5 Server-Side Software

The server is responsible for receive and deliver all the data to the website, for either mobile or desktop, using Python scripts, and data visualization, using Javascript, CSS and HTML scripts.

4.5.1 Website design

The website is composed of three parts: the apresentation of the project, dashboard to visualize data, and a prediction modal, to predict based on the model and characteristics of the soil and climate.

In this sub-section of the chapter, we will make a brief introducing to the website design, to later be explain how some features were made. As is show in figure 28, this modal appears when it is click in the navigation bar to predict, and in figure 29, we have the user login modal, to get to the user dashboard, for data visualization and analysis.

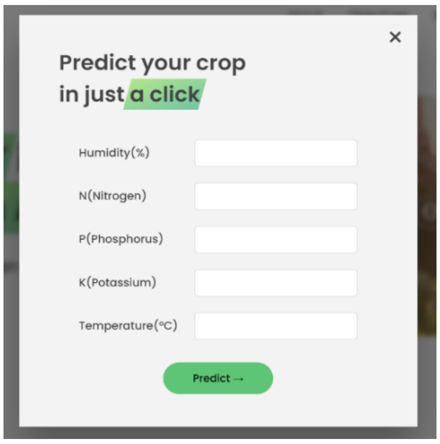


Figure 28: Predict Modal

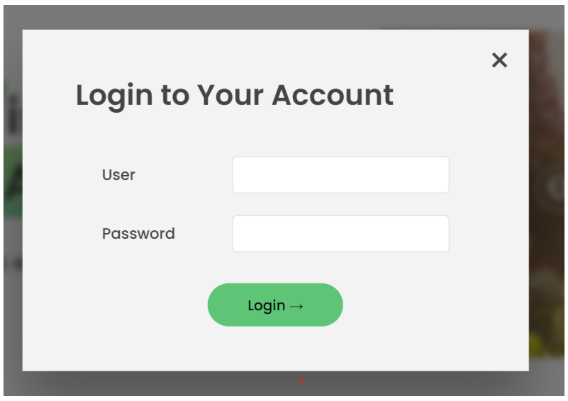
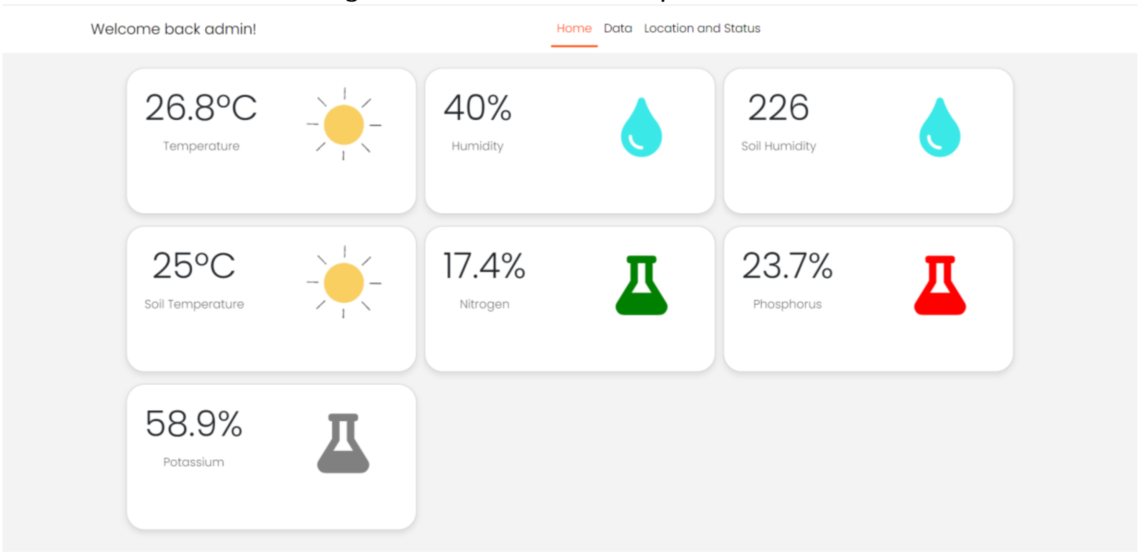


Figure 29: Login Modal

The user dashboard is divided by three parts : real-time sensor data, graphics visualization and crop status.

In real-time sensor, as figure 30 shows, the data in that section is the last 30 minute capture by the sensors.

Figure 30: Real-time data from dashbo


Welcome back admin!

Home Data Location and Status

26.8ºC Temperature

40% Humidity

226 Soil Humidity

25ºC Soil Temperature

17.4% Nitrogen

23.7% Phosphorus

58.9% Potassium

ard

In figure 31, is the data visualization and analysis section, where we can interpretate, avaluate and discuss some values. This data comes from the SQLite database that we are creating, so it is all the data that we collect every day.

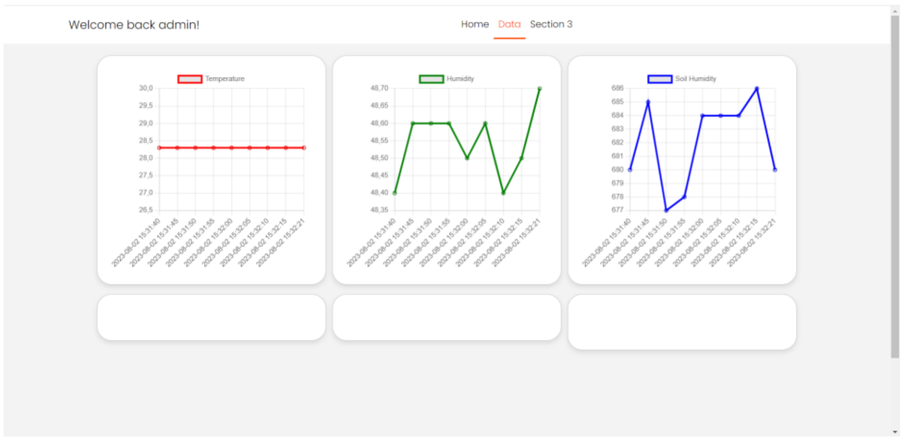


Figure 31: Data visualization from dashboard

And finally, to introduce the whole dashboard, in the last section we have the status dashboard, where we can check the crop health and location.

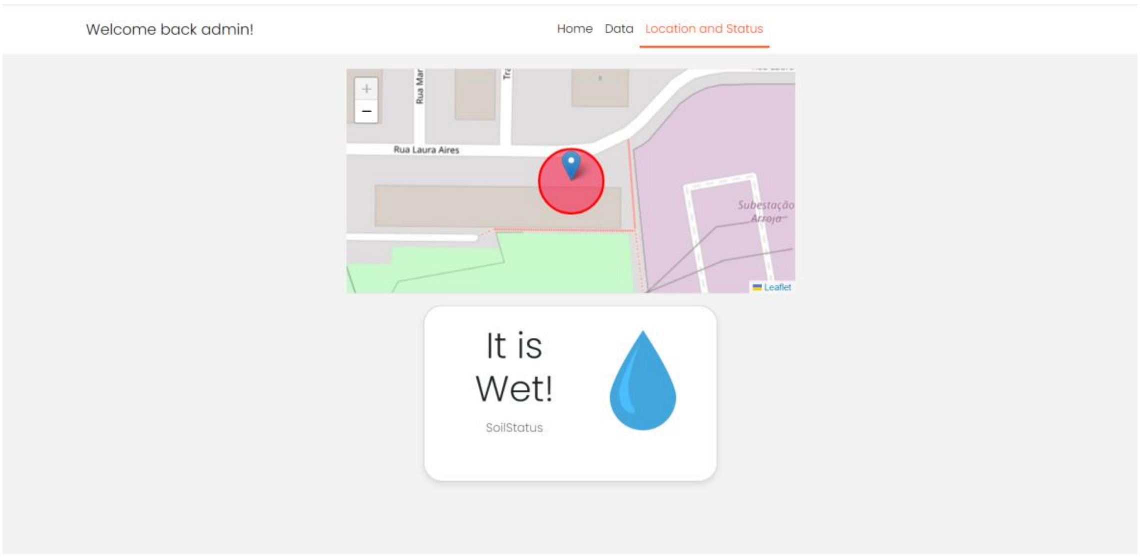
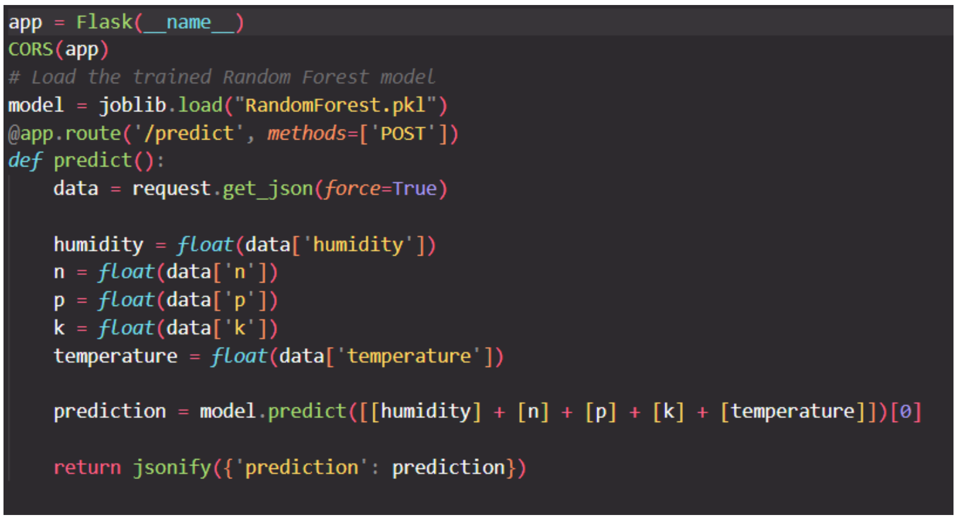


Figure 32: Crop status and location

4.5.2 Crop Prediction

After all the analysis and choice of the best model for prediction, we needed to find a way to communicate between the model and the input data.

After inputing data in the modal of the figure 26, we send that same data via a Post HTTP request on route "/predict", and in the body a json with the data, this to send the data from the server to the website. We then predict, using the Random Forest model trained, the crop for the data, and we send that prediction to the server, to answer the user request. This process is demonstrated in figure 33.

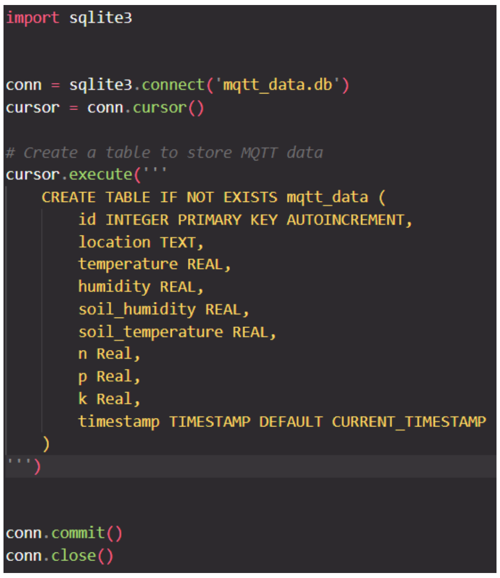


```
app = Flask(__name__)
CORS(app)
# Load the trained Random Forest model
model = joblib.load("RandomForest.pkl")
@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)

    humidity = float(data['humidity'])
    n = float(data['n'])
    p = float(data['p'])
    k = float(data['k'])
    temperature = float(data['temperature'])

    prediction = model.predict([[humidity] + [n] + [p] + [k] + [temperature]])[0]

    return jsonify({'prediction': prediction})
```

Figure 33: Script for HTTP request prediction

4.5.3 Database creation

This is a simple step, where we find the best data to store and create the SQLite database. The data stored in the database is all the data received from the sensors, location and, for graphic visualization and analysis, the timestamp of each data capture, this database creation can be seen in figure 34.



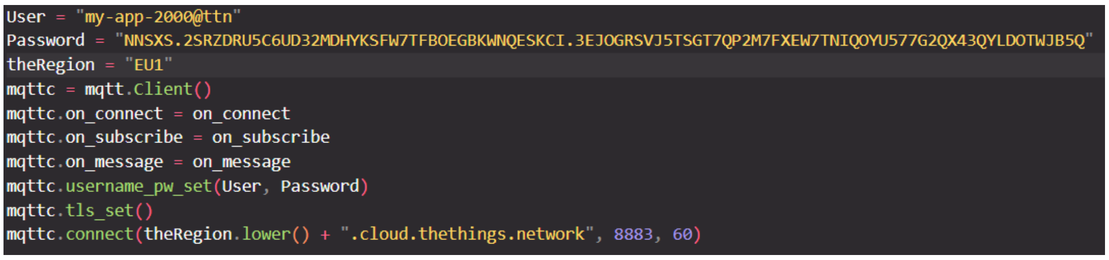
```python
import sqlite3

conn = sqlite3.connect('mqtt_data.db')
cursor = conn.cursor()

# Create a table to store MQTT data
cursor.execute('''
    CREATE TABLE IF NOT EXISTS mqtt_data (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        location TEXT,
        temperature REAL,
        humidity REAL,
        soil_humidity REAL,
        soil_temperature REAL,
        n Real,
        p Real,
        k Real,
        timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    )
''')


conn.commit()
conn.close()
```

Figure 34: Database creation

4.5.4 MQTT protocol

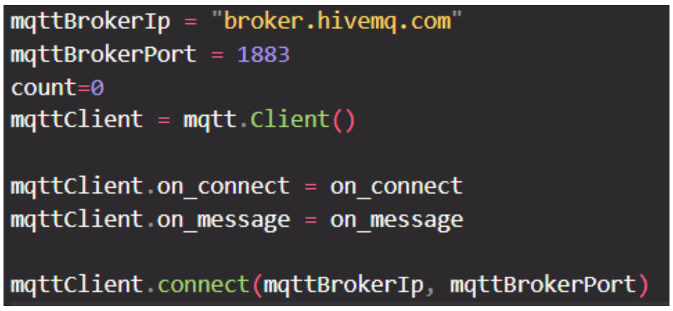
To get the data into the server, we used MQTT protocol for either the dragino gateway and The Things gateway, because both have integrated the protocol and it is easy of use, however one of them has a broker defined and can be easily accessed and the other needs the HiveMQ broker, as said in the previous chapter. So, our scripts will have two diferents subscriptions to different brokers.

Each of them has different connection to the MQTT client, one with the topic and the other via username and password. In figure 35, we can see that to connect to the MQTT client from TTN, it will be used the application name and a generated password, and in figure 36 the MQTT client will be connected to the HiveMQ public broker.



```
User = "my-app-2000@ttn"
Password = "NNSXS.2SRZDRU5C6UD32MDHYKSFW7TFBOEGBKWNQESKCI.3EJOGRSVJ5TSGT7QP2M7FXEW7TNIQOYU577G2QX43QYLDOTWJB5Q"
theRegion = "EU1"
mqttc = mqtt.Client()
mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe
mqttc.on_message = on_message
mqttc.username_pw_set(User, Password)
mqttc.tls_set()
mqttc.connect(theRegion.lower() + ".cloud.thethings.network", 8883, 60)
```

Figure 35: MQTT protocol for TTN



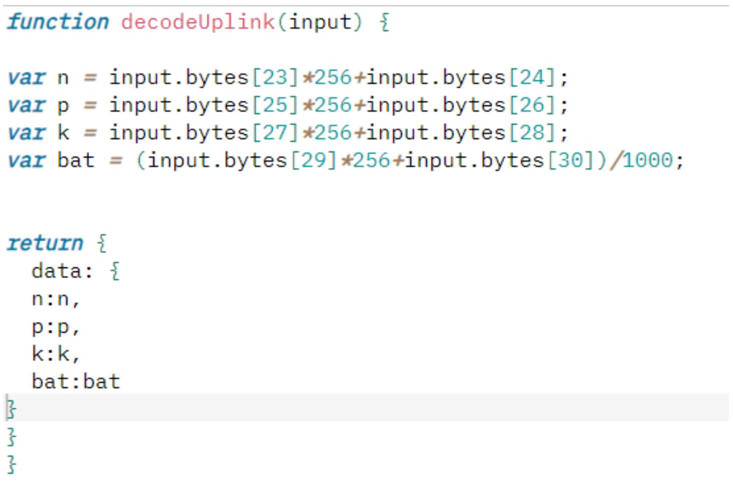
```
mqttBrokerIp = "broker.hivemq.com"
mqttBrokerPort = 1883
count=0
mqttClient = mqtt.Client()

mqttClient.on_connect = on_connect
mqttClient.on_message = on_message

mqttClient.connect(mqttBrokerIp, mqttBrokerPort)
```

Figure 36: MQTT protocol for dragino

And for each client, different subscriptions are made. For TTN, we only have one device connect to it, so we subscribe for that device. And for dragino, we will be subscribing to a certain topic, where all the data is sent, and in this case to the topic “dragino-1ed75c/2202439/data”, as explained in the previous chapter.

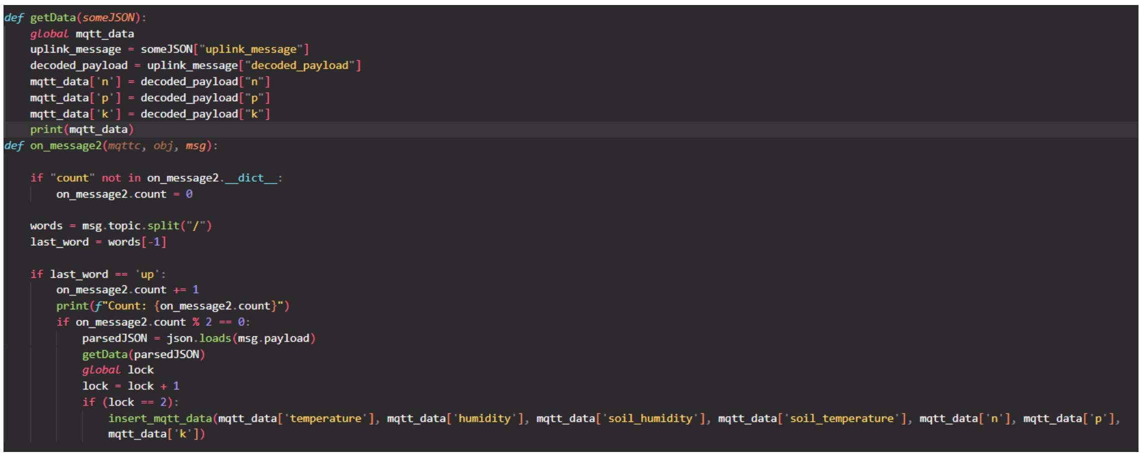
Then the data received needs to be processed to be storage in our database and analysis on the website. In figure 38, it is showing the process of receiving the data from TTN, where when subscribes data, it will come three messages and only in the last one, in this case, the second uplink message, we get the payload from it, and it is decodified already, because in the TTN we had a function to decode and split the data we needed, like figure 37 shows. As seen in figure 26, the data came from the sensors is divided by a “&”, for in the process of subscribing to the topic, we can divide the data easily, as show in figure 39.

Both have a variable called “lock”, this is an important factor in the process of gathering and inserting data from all sensors, because we have two clients running at the same time, and both aren’t synchronized to receive the data at the same time. So, this variable was created, to wait for both data to be collected, and to insert all data into the database at the same time, instead of creating two databases for each client.



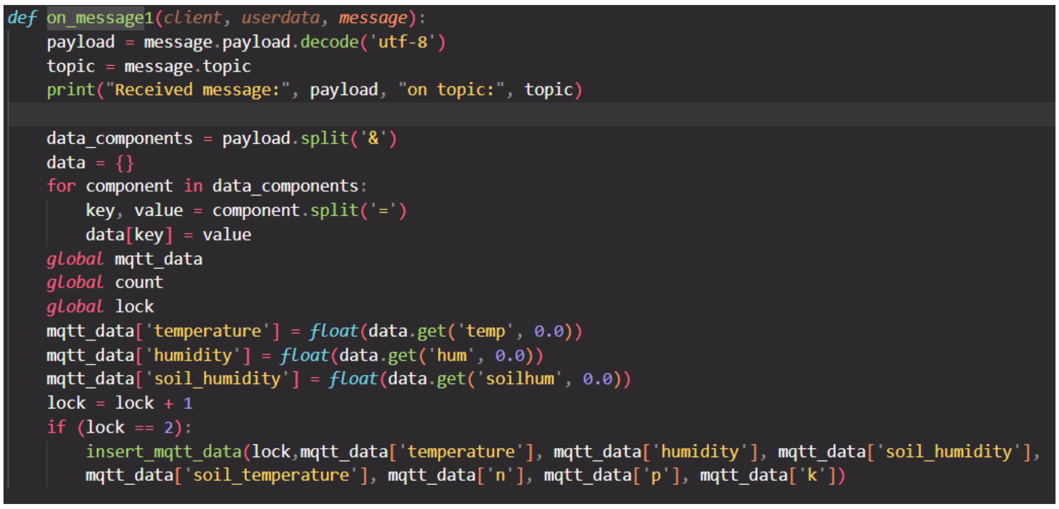
```javascript
function decodeUplink(input) {

var n = input.bytes[23]*256+input.bytes[24];
var p = input.bytes[25]*256+input.bytes[26];
var k = input.bytes[27]*256+input.bytes[28];
var bat = (input.bytes[29]*256+input.bytes[30])/1000;


return {
  data: {
  n:n,
  p:p,
  k:k,
  bat:bat
}
}
}
```

Figure 37: Decoder for npk sensor



```python
def getData(someJSON):
    global mqtt_data
    uplink_message = someJSON["uplink_message"]
    decoded_payload = uplink_message["decoded_payload"]
    mqtt_data['n'] = decoded_payload["n"]
    mqtt_data['p'] = decoded_payload["p"]
    mqtt_data['k'] = decoded_payload["k"]
    print(mqtt_data)
def on_message2(mqttc, obj, msg):

    if "count" not in on_message2.__dict__:
        on_message2.count = 0

    words = msg.topic.split("/")
    last_word = words[-1]

    if last_word == 'up':
        on_message2.count += 1
        print(f"Count: {on_message2.count}")
        if on_message2.count % 2 == 0:
            parsedJSON = json.loads(msg.payload)
            getData(parsedJSON)
            global lock
            lock = lock + 1
            if (lock == 2):
                insert_mqtt_data(mqtt_data['temperature'], mqtt_data['humidity'], mqtt_data['soil_humidity'], mqtt_data['soil_temperature'], mqtt_data['n'], mqtt_data['p'],
                mqtt_data['k'])
```

Figure 38: Data received and conversion from TTN

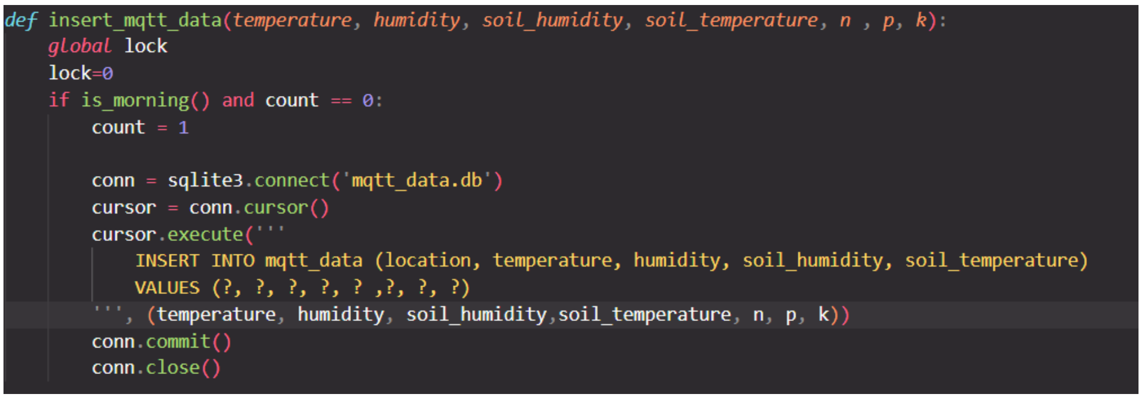


```python
def on_message1(client, userdata, message):
    payload = message.payload.decode('utf-8')
    topic = message.topic
    print("Received message:", payload, "on topic:", topic)

    data_components = payload.split('&')
    data = {}
    for component in data_components:
        key, value = component.split('=')
        data[key] = value
    global mqtt_data
    global count
    global lock
    mqtt_data['temperature'] = float(data.get('temp', 0.0))
    mqtt_data['humidity'] = float(data.get('hum', 0.0))
    mqtt_data['soil_humidity'] = float(data.get('soilhum', 0.0))
    lock = lock + 1
    if (lock == 2):
        insert_mqtt_data(lock,mqtt_data['temperature'], mqtt_data['humidity'], mqtt_data['soil_humidity'],
        mqtt_data['soil_temperature'], mqtt_data['n'], mqtt_data['p'], mqtt_data['k'])
```

Figure 39: Data received and conversion from dragino

4.5.5 Database insertion

After we waited for data coming from dragino and The Things Gateway, it is time to insert it into our database. In the database creation, the timestamp is already being inserted automatically, but the rest of the data is not. It was created a function, to receive all data left to complete the database. As shown in figure 40, the function is a simple process, and when we enter this function our "lock" variable is set to 0, for the next data collection. It only records data in the morning between 9 and 10 am.

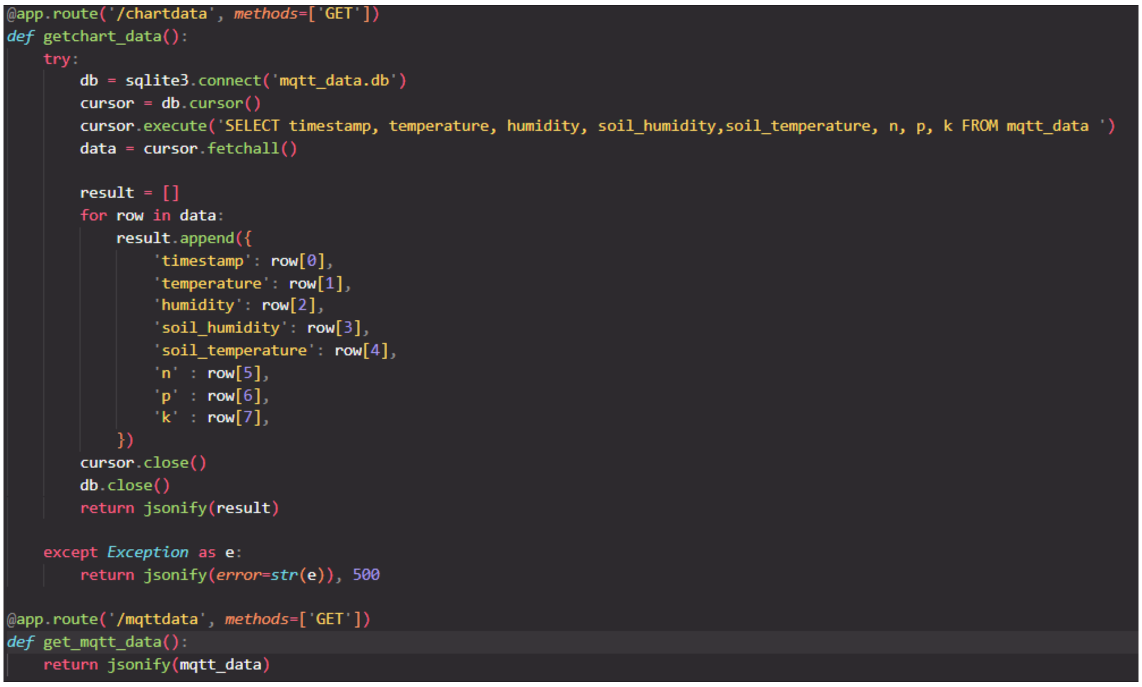


```
def insert_mqtt_data(temperature, humidity, soil_humidity, soil_temperature, n , p, k):
    global lock
    lock=0
    if is_morning() and count == 0:
        count = 1

        conn = sqlite3.connect('mqtt_data.db')
        cursor = conn.cursor()
        cursor.execute('''
            INSERT INTO mqtt_data (location, temperature, humidity, soil_humidity, soil_temperature)
            VALUES (?, ?, ?, ?, ? ,?, ?, ?)
        ''', (temperature, humidity, soil_humidity,soil_temperature, n, p, k))
        conn.commit()
        conn.close()
```

Figure 40: Function to insert data in database

4.5.6 Fetch data

After all the process of adquiring, process and storage of the data, we send that same data to our website. Every time we enter in the dashboard, we will fetch for "/chardata", for graphic visualization and "/mqttdata" for real-time visualization, as we can see in figure 41.



```
@app.route('/chartdata', methods=['GET'])
def getchart_data():
    try:
        db = sqlite3.connect('mqtt_data.db')
        cursor = db.cursor()
        cursor.execute('SELECT timestamp, temperature, humidity, soil_humidity,soil_temperature, n, p, k FROM mqtt_data ')
        data = cursor.fetchall()

        result = []
        for row in data:
            result.append({
                'timestamp': row[0],
                'temperature': row[1],
                'humidity': row[2],
                'soil_humidity': row[3],
                'soil_temperature': row[4],
                'n' : row[5],
                'p' : row[6],
                'k' : row[7],
            })
        cursor.close()
        db.close()
        return jsonify(result)

    except Exception as e:
        return jsonify(error=str(e)), 500

@app.route('/mqttdata', methods=['GET'])
def get_mqtt_data():
    return jsonify(mqtt_data)
```

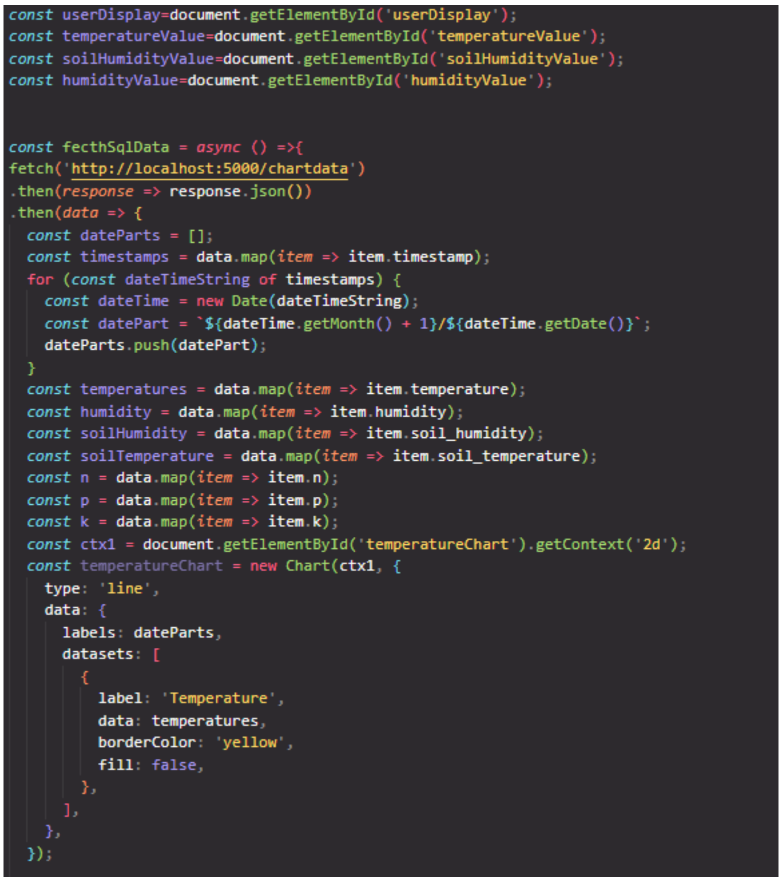
Figure 41: Fetch data to website

4.5.7 GPS Location

This part of the thesis we adapted so we didn’t use a sensor, because of the hardware choice, and the LoRa shield needs a dragino yuno shield, and so we use an API to get our current latitude and longitude of the network gateway. This API takes the network wifi connecting to the system and get its current location. So, the network will be our location in this case.

4.5.8 Website

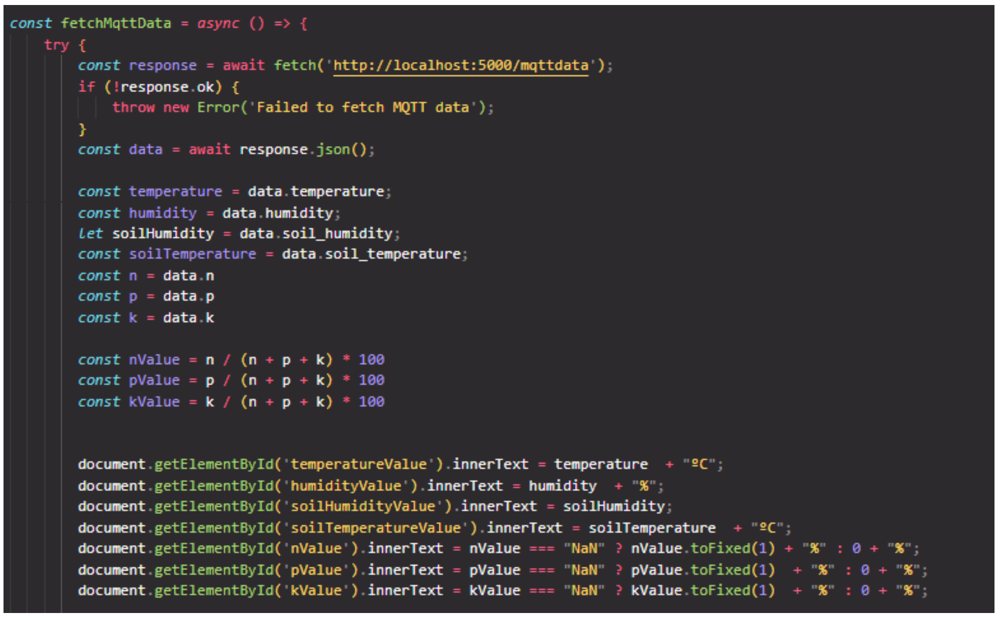
To the purpose of graphic visualization, the data sent from the gateways, a simple dashboard website was built. In figure 42, we await the get fetch from “/chardata” to get all the data from the sensors. The fetch will get all the data in the database and make a representation for each value. And then charts are created with the library from javascript, Chartjs, this is done for the rest of the values.



```
const userDisplay=document.getElementById('userDisplay');
const temperatureValue=document.getElementById('temperatureValue');
const soilHumidityValue=document.getElementById('soilHumidityValue');
const humidityValue=document.getElementById('humidityValue');


const fecthSqlData = async () =>{
fetch('http://localhost:5000/chartdata')
.then(response => response.json())
.then(data => {
  const dateParts = [];
  const timestamps = data.map(item => item.timestamp);
  for (const dateTimeString of timestamps) {
    const dateTime = new Date(dateTimeString);
    const datePart = `${dateTime.getMonth() + 1}/${dateTime.getDate()}`;
    dateParts.push(datePart);
  }
  const temperatures = data.map(item => item.temperature);
  const humidity = data.map(item => item.humidity);
  const soilHumidity = data.map(item => item.soil_humidity);
  const soilTemperature = data.map(item => item.soil_temperature);
  const n = data.map(item => item.n);
  const p = data.map(item => item.p);
  const k = data.map(item => item.k);
  const ctx1 = document.getElementById('temperatureChart').getContext('2d');
  const temperatureChart = new Chart(ctx1, {
    type: 'line',
    data: {
      labels: dateParts,
      datasets: [
        {
          label: 'Temperature',
          data: temperatures,
          borderColor: 'yellow',
          fill: false,
        },
      ],
    },
  });
```

Figure 42: Get data from sensor and build graphic

For the real-time values, we fetch the "/mqttdata" url, to get only the real-time values from the sensors and put in the specific element on the website html.



```
const fetchMqttData = async () => {
    try {
        const response = await fetch('http://localhost:5000/mqttdata');
        if (!response.ok) {
            throw new Error('Failed to fetch MQTT data');
        }
        const data = await response.json();

        const temperature = data.temperature;
        const humidity = data.humidity;
        let soilHumidity = data.soil_humidity;
        const soilTemperature = data.soil_temperature;
        const n = data.n
        const p = data.p
        const k = data.k

        const nValue = n / (n + p + k) * 100
        const pValue = p / (n + p + k) * 100
        const kValue = k / (n + p + k) * 100


        document.getElementById('temperatureValue').innerText = temperature  + "ºC";
        document.getElementById('humidityValue').innerText = humidity  + "%";
        document.getElementById('soilHumidityValue').innerText = soilHumidity;
        document.getElementById('soilTemperatureValue').innerText = soilTemperature  + "ºC";
        document.getElementById('nValue').innerText = nValue === "NaN" ? nValue.toFixed(1) + "%" : 0 + "%";
        document.getElementById('pValue').innerText = pValue === "NaN" ? pValue.toFixed(1)  + "%" : 0 + "%";
        document.getElementById('kValue').innerText = kValue === "NaN" ? kValue.toFixed(1)  + "%" : 0 + "%";
```

Figure 43: Real Time values

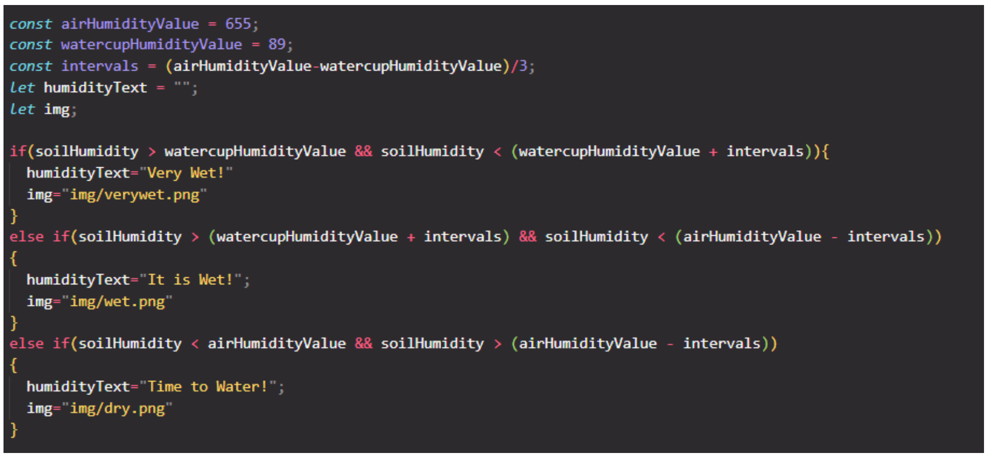
Chapter 5 – Results

The results are going to be tested in two different places, indoor and outdoor, and in the same amount of time, 15 days.

5.1 Indoor Test

To start collecting and visualising some data, we checked first our soil and climate conditions, for this we used a flower vase. The used fertilizer has a 10% nitrogen, 12% phosphorus and 18% potassium, meaning that the potassium will be always the decisive value for deciding what crop will we be using. The experimentation was conducted on my balcony, as illustrated by figure 45, during the summer season. It is imperative to note that this environmental setting had a substantial impact on the recorded temperature and humidity parameters. One crucial factor to consider is the absence of the antenna on The Things Gateway device due to its remarkable long-range capabilities, although it is illustrated in the figure, and the dragino gateway is not illustrated, but working behind the desktop. This omission significantly influenced the data collection and transmission aspects of our experiment. The range of the antenna will be tested in the open space.

The soil moisture sensor is capacitive, so the value range goes from 0 to 660 dimensionless value (5V DC, 10-bit ADC), that is calculated by the output voltage of the sensor, with a range of 0V to approximately 2.9V. We test the sensor to check the intervals of dry, very wet, and wet. The sensor, as shown in figure 44, needs two values, one value when it is exposed to the air, in this case we test the humidity of the air, to check when he has the lowest percentage of humidity and a second value, when he has the most amount of water, so we tested the soil with the most amount of water he could possibly have. We then divide the difference between these two values by three, and we get three intervals: “wet”, “dry” and “very wet”. Three if conditions were made for each interval, and this text will then be showed in the status tab in dashboard.



```
const airHumidityValue = 655;
const watercupHumidityValue = 89;
const intervals = (airHumidityValue-watercupHumidityValue)/3;
let humidityText = "";
let img;

if(soilHumidity > watercupHumidityValue && soilHumidity < (watercupHumidityValue + intervals)){
  humidityText="Very Wet!"
  img="img/verywet.png"
}
else if(soilHumidity > (watercupHumidityValue + intervals) && soilHumidity < (airHumidityValue - intervals))
{
  humidityText="It is Wet!";
  img="img/wet.png"
}
else if(soilHumidity < airHumidityValue && soilHumidity > (airHumidityValue - intervals))
{
  humidityText="Time to Water!";
  img="img/dry.png"
}
```

Figure 44: Soil humidity sensor data process

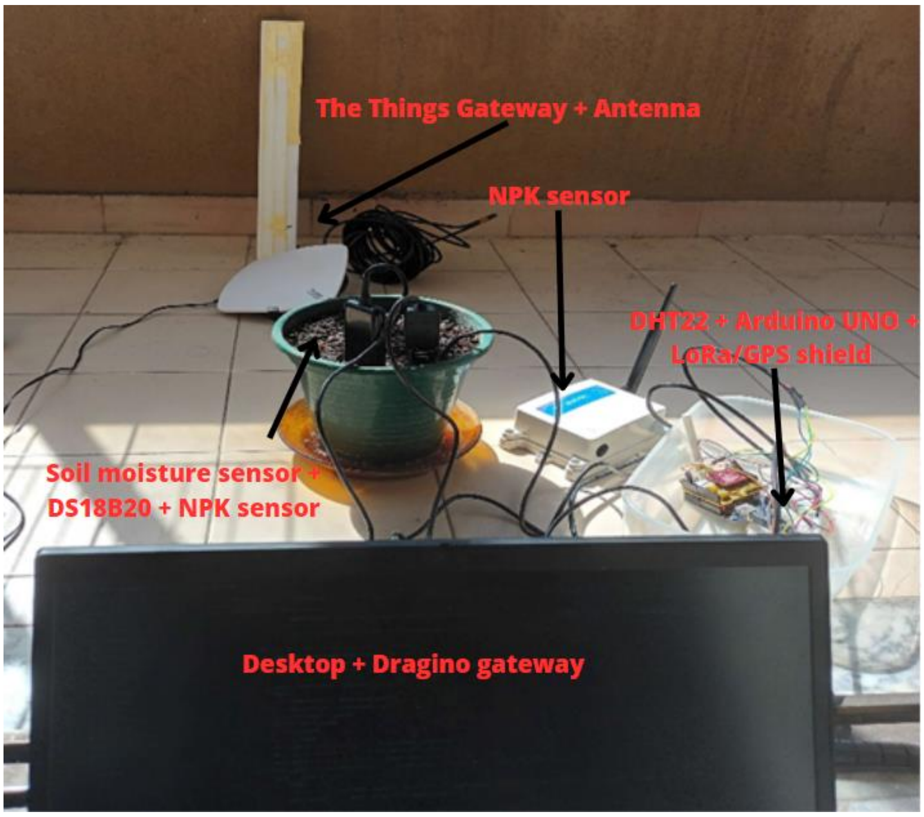

The Things Gateway + Antenna
NPK sensor
DHT22 + Arduino UNO + LoRa/GPS shield
Soil moisture sensor + DS18B20 + NPK sensor
Desktop + Dragino gateway

Figure 45: Experimental setup structure indoor test

As we can see in table 2, in the first day we had a dry soil with zero concentration of any kind of nutrient, so we added a cup of water to first check how the value would oscilate. In the second day, we can see that a cup of water in the vase made the value go from 640 to 531, and this went to the limit of the wet interval, so we had to add another cup of water and use fertilizer to test it. In the third day, we checked the values, and we finally have all data necessary for the prediction, and the result of it is grapes. This mean that we must try to regularize the values from water, temperature and npk concentration for a continuous healthy growth of grapes. In terms of soil humidity, we have the intervals programmed, so the website will always tell us the required soil moisture. The next days we checked, in the prediction modal, if the values still got grapes as the best crop to our soil, and we get it everyday in the 15 days. We can confirm that the values of npk concentration doesn't decrease that much in the 15 days, and only in day 10 we add 20 grams of fertilizer.

After these 15 days, we will start the outdoor test, to then compare both tables.

Table 2 : Indoor test data

Day	Temperature ºC	Humidity %	SoilTemperature ºC	SoilHumidity	N Mg/kg	P Mg/kg	K Mg/kg
1	23.0	49.1	26.5	640	0	0	0
2	21.8	49.3	25.2	531	0	0	0
3	21.5	47.5	24.8	363	370	506	1255
4	19.0	49.9	22.9	310	370	506	1255
5	21.9	50.1	25.7	435	370	506	1255
6	22.3	47.9	25.8	499	343	491	1192
7	23.1	48.0	26.1	566	305	481	1168
8	22.4	48.5	26.0	400	283	476	1156
9	20.8	49.8	24.6	440	268	466	1133
10	20.1	47.8	23.7	484	257	461	1121
11	21.2	49.1	25.0	511	334	490	1189
12	22.6	50.2	26.4	549	317	480	1165
13	26.1	48.4	29.6	450	294	475	1153
14	27.0	49.3	30.1	505	270	465	1129
15	21.9	51.1	25.2	533	262	451	1095

5.2 Outdoor Test

For this test, we went to Benavente and like indoor test, we will start by verifying how are the soil conditions and temperature and humidity of our climate outdoor. First, as we said in the other test, we checked the range of our antenna of The Things Gateway. To do this, we turn on the npk sensor, in every step we take, to check if the gateway receives data. After 60 meters of trying to communicate with the gateway, we finally didn't get any kinda of response from the sensor, so the max range is approximately 60 meters. The ecosystem outdoor is shown in figure 46, while the The Things Gateway and dragino are at the necessary range to get the data.

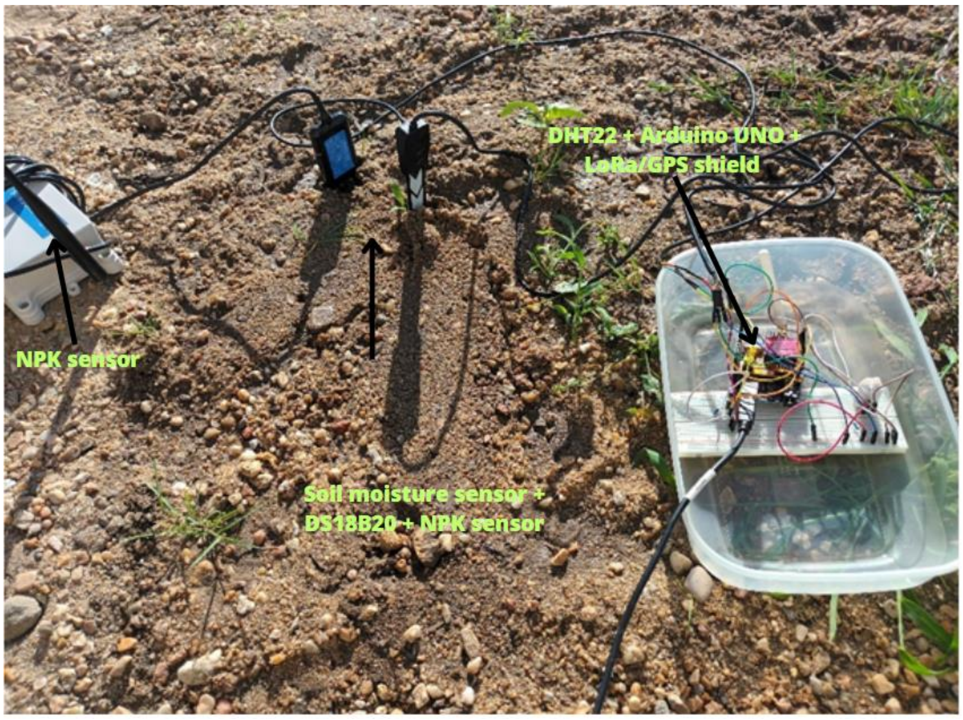

DHT22 + Arduino UNO + LoRa/GPS shield
NPK sensor
Soil moisture sensor + DS18B20 + NPK sensor

Figure 46: Outdoor experimental setup

We take the same steps, like the indoor test. First, we checked the starting values, make actions according to the values, and if we already have all the values from the first day, we predict the crop and regularize it.

The soil moisture sensor was measured to check the intervals in outdoor environment. We again get the data from the air and the value of the earh with the most water possible and just like in the indoor test and in figure 44, we will get another three intervals for each soil humidity condition: “wet”, “very wet” and “dry”.

In the first day, we got already npk concentration values, so since day one we will regularize the values according to the prediction made by our model. In the prediction modal, we inserted the data collected, and we got, as result, mango. The fertilizer we had to the indoor test can be used in this aswell, because the highest concentration is still potassium, and the less concentration is nitrogen, however we can’t add to much of fertilizer, to not exceed the concentration value for mango growth. The values are a bit low, so we were a little bit more careful adding the fertilizer. In the same day we added a bit of fertilizer to check what values we got the next day.

As we can see in table 3, the amount of fertilizer added in the first day (8 grams), was a good value, because the values didn’t increase that much, and we can use this amount in the next days, if needed. The soil humidity doesn’t oscilate a lot, in the first days, because we had the help of the rain to water the earth, and so we just need to check if it didn’t pass the “wet” interval. The rest of the days, were easy to maintain, just adding the amount that we added in day one, so the values don’t go to low, and adding more water when the interval pass to the “dry” interval.

In comparison with the indoor test, the outdoor test was the easiest one, by several number of factors, like rain, having already a npk concentration, and not being inside on a hot place, like my balcony, so the values are not deceived. However, the amount of time and not being able to full growth a crop, doesn’t help to make better comparison between each test.

Table 3 : Outdoor test data

Day	Temperature (ºC)	Humidity (%)	SoilTemperature (ºC)	SoilHumidity (Dimensioneless)	N (mg/kg)	P (mg/kg)	K (mg/kg)
1	30.7	55.6	33.6	599	12	16	42
2	26.4	52.3	29.6	574	16	27	60
3	22.9	53.7	26.1	534	15	25	58
4	23.2	54.1	26.5	455	12	22	53
5	25.5	51.5	28.6	393	22	36	75
6	25.8	50.8	29.2	366	18	30	67
7	23.6	53.3	26.3	322	16	24	59
8	24.2	51.6	27.5	293	13	19	47
9	27.3	52.1	30.1	250	20	32	53
10	30.1	50.1	33.4	283	17	24	55
11	30.5	48.9	33.8	301	15	20	48
12	29.4	49.6	32.9	342	13	18	43
13	30.0	51.2	33.1	396	21	36	76
14	34.2	53.2	37.1	521	18	27	61
15	34.8	56.3	37.4	350	15	20	49

5.3 Indoor vs Outdoor Test

As figure 47 and figure 48 shows, both temperature and soil temperature are the same, and only the humidity and npk concentration can be changed with fertilizer and water. We can see that in indoor, figure 47, we put a lot of fertilizer in a vase, and so it doesn’t oscilate a lot, as we can see in the graphics. While the outdoor, figure 48 the values oscilate a lot as the concentration in the soil is minimum and difficult to regulate, however the values and prediction still say that the best crop is mango.

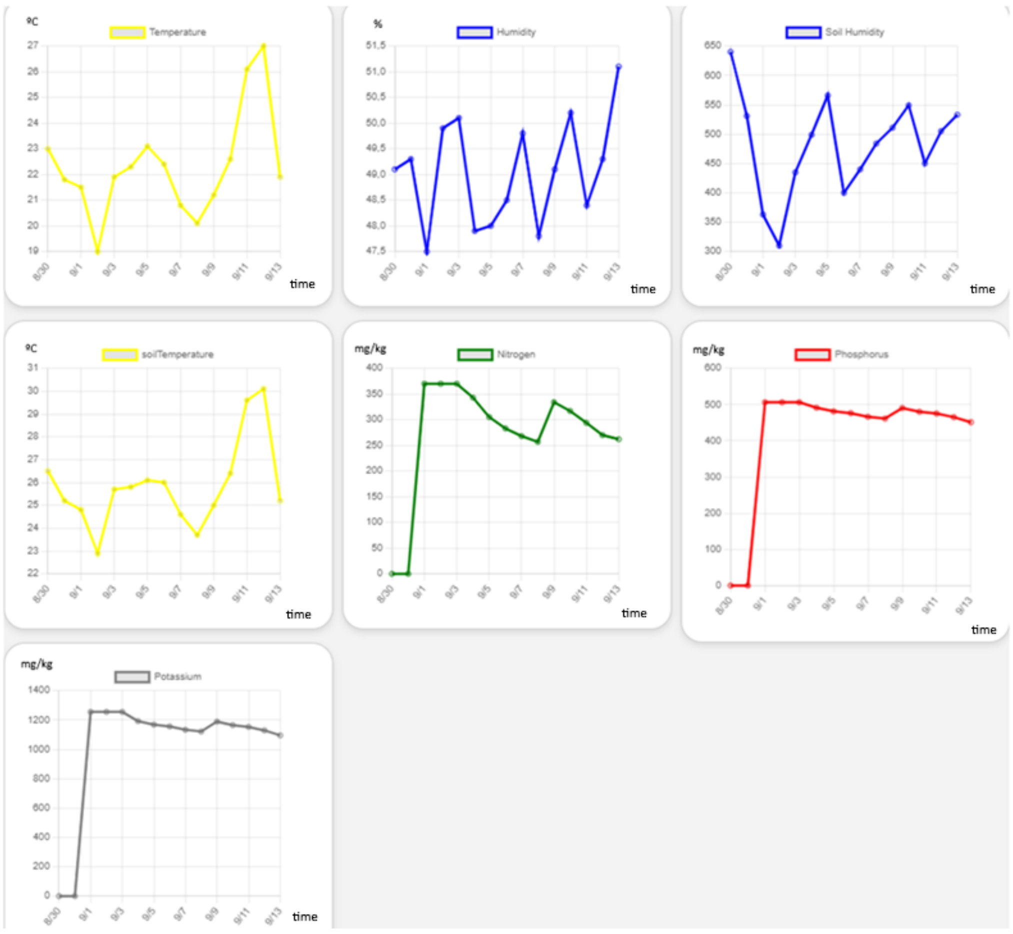


Figure 47: Indoor data visualization

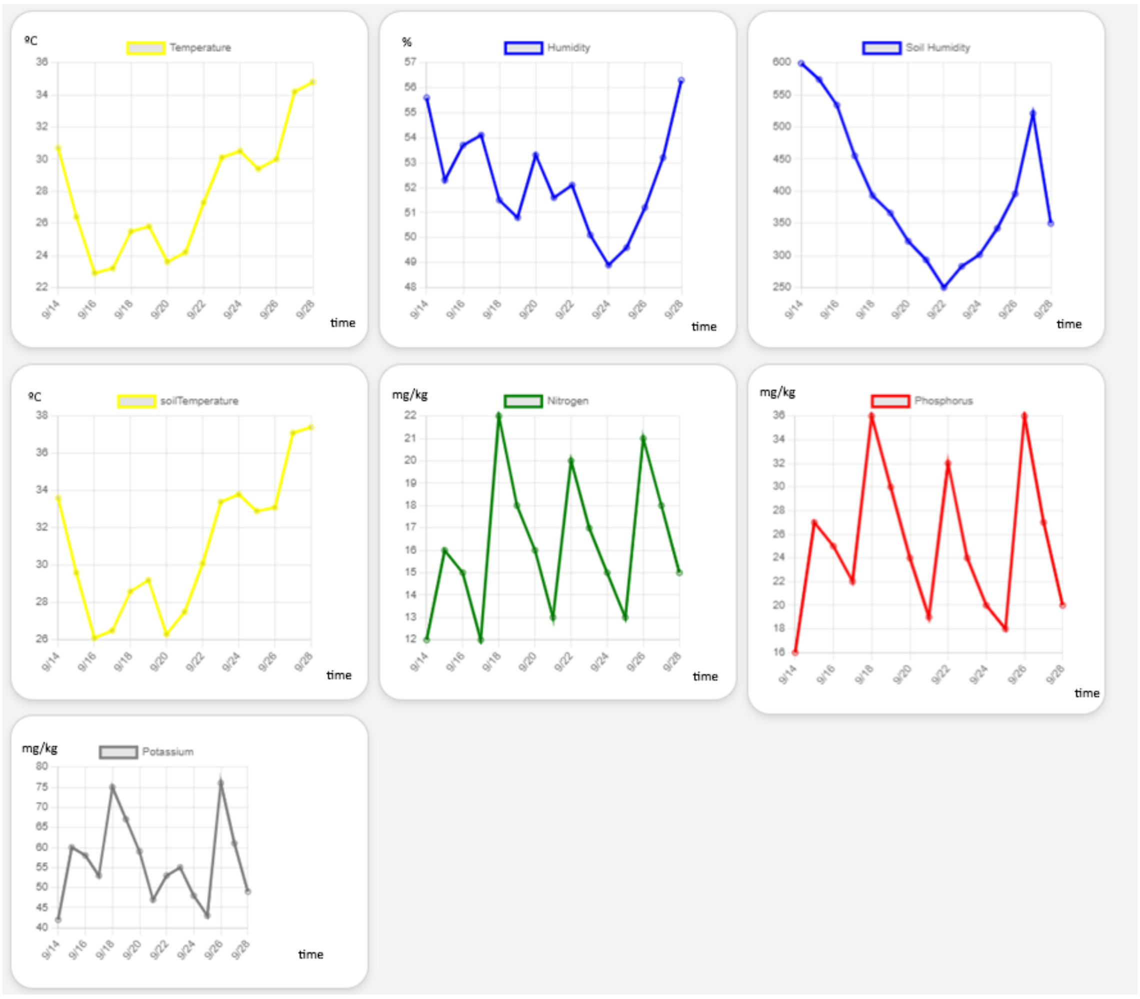


Figure 48: Outdoor data visualization

And in figure 49, we can see that one doesn't have urbanization (Benavente) and other it is urban (Odivelas).

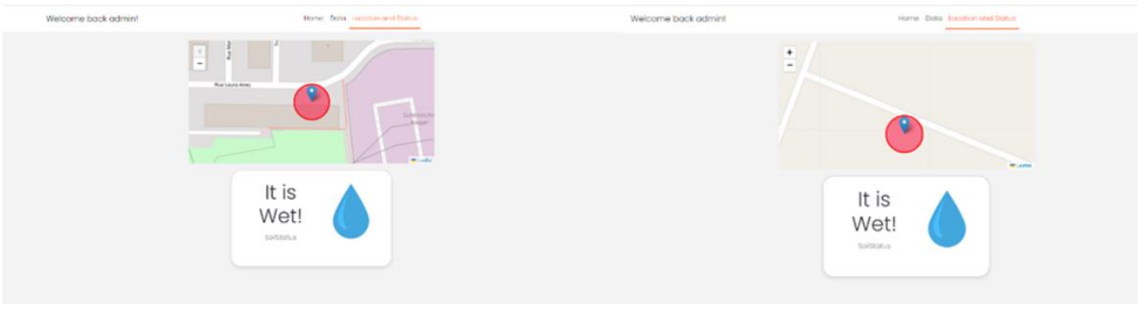


Figure 49: Benavente e Odivelas

Chapter 6 – Conclusions and Future Work

6.1 Conclusions

The goal of this dissertation was to build an IoT system with capabilities on soil moisture and nutrient concentration measurements and with the general objective to make a crop growth, in this case, to maintain the ideal values for it to growth. Therefore, the following developed solution is presented:

- Ecosystem of sensors that send data via LoRa or LoRaWan to the specific hardware, either dragino or The Things Gateway, as the core of the project for collecting data, this because of the limitations of dragino and the need of a gateway that uses LoRaWAN communication for NPK sensor,
- Server that subscribes both MQTT protocol from each gateway to then storage in SQLite database and send data to the website,
- Mobile and desktop website, for both, project apresentation and data visualization.

As described above, we have the three big platforms working: the core system, (Arduino UNO + sensor node + dragino gateway) and (NPK sensor + The Things Gateway), the cloud server with the python scripts for storage data in SQLite database and the mobile and desktop website with the presentation of the project and for data visualization.

The planning made in the beginning of the Thesis, we just didn't do the paper for the conference, didn't implement actuators, and the data collection was a bit later of what was planned because of several difficulties with the hardware, such as connection between gateway and sensor and malfunctioning. Although all of this wasn't planned, the project had great results and has a big core to an entrepreneur project involving technology transfer, because we could maintain the values, although being put manually the water and nutrients and not automatically, and because it has a good system for every crop growth.

6.2 Future Work

The developed ecosystem is of great importance in helping farmers, however there are still many improvements for this project to be complete and ready to be adopted from a business perspective:

- Automatization by using actuators like water pumps,
- Extending the database for better prediction with help of big data analytics,
- Improving the dashboard of the website, by giving the opportunity to farmers control the actuators, if necessary,
- Disease detection, installation of a real-time camera to check crops for diseases and therefore creating a database for then use for prediction,
- Security practices, although some were considered, for data encryption and website security,
- GPS sensor working with our core.

References

[1] M. R. M. Kassim, 'IoT Applications in Smart Agriculture: Issues and Challenges', in *2020 IEEE Conference on Open Systems (ICOS)*, Nov. 2020, pp. 19–24. doi: 10.1109/ICOS50156.2020.9293672.

[2] A. Mohanty, 'Impacts of climate change on human health and agriculture in recent years', in *2021 IEEE Region 10 Symposium (TENSYMP)*, Aug. 2021, pp. 1–4. doi: 10.1109/TENSYMP52854.2021.9550876.

[3] M. Dholu and K. A. Ghodinde, 'Internet of Things (IoT) for Precision Agriculture Application', in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, May 2018, pp. 339–342. doi: 10.1109/ICOEI.2018.8553720.

[4] S. Babu, 'A software model for precision agriculture for small and marginal farmers', in *2013 IEEE Global Humanitarian Technology Conference: South Asia Satellite (GHTC-SAS)*, Aug. 2013, pp. 352–355. doi: 10.1109/GHTC-SAS.2013.6629944.

[5] Alan Hevner and Samir Chatterjee, *Design science research in information systems*. 2010.

[6] J. Baek and M. W. Kanampiu, 'A Strategic Sensor Placement for a Smart Farm Water Sprinkler System: A Computational Model', in *2022 24th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2022, pp. 53–57. doi: 10.23919/ICACT53585.2022.9728828.

[7] S. Jaisankar, P. Nalini, and K. K. Rubigha, 'A Study on IoT based Low-Cost Smart Kit for Coconut Farm Management', in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Oct. 2020, pp. 161–165. doi: 10.1109/I-SMAC49090.2020.9243486.

[8] 'Affordable Smart Farming Using IoT and Machine Learning | IEEE Conference Publication | IEEE Xplore'. Accessed: Jan. 27, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/8663044

[9] H. P. Thakor and S. Iyer, 'Development and Analysis of Smart Digi-farming Robust Model for Production Optimization in Agriculture', in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, Mar. 2019, pp. 461–465.

[10] C. Yoon, M. Huh, S.-G. Kang, J. Park, and C. Lee, 'Implement smart farm with IoT technology', in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2018, pp. 749–752. doi: 10.23919/ICACT.2018.8323908.

[11] N. Islam, B. Ray, and F. Pasandideh, 'IoT Based Smart Farming: Are the LPWAN Technologies Suitable for Remote Communication?', in *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*, Aug. 2020, pp. 270–276. doi: 10.1109/SmartIoT49966.2020.00048.

[12] M. S. D. Abhiram, J. Kuppili, and N. A. Manga, 'Smart Farming System using IoT for Efficient Crop Growth', in *2020 IEEE International Students' Conference on*

Electrical,Electronics and Computer Science (SCEECS), Feb. 2020, pp. 1–4. doi: 10.1109/SCEECS48394.2020.147.

[13] T. S. Gunawan, N. N. Kamarudin, M. Kartiwi, and M. R. Effendi, 'Automatic Watering System for Smart Agriculture using ESP32 Platform', in *2022 IEEE 8th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA),* Sep. 2022, pp. 185–189. doi: 10.1109/ICSIMA55652.2022.9928950.

[14] V. Goyal, A. Yadav, and R. Mukherjee, 'Thermo graphic Camera-Based E-IoT Enabled Architecture for Smart Poultry Farm', in *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT),* Dec. 2021, pp. 772–775. doi: 10.1109/ICEECCOT52851.2021.9708062.

[15] J. Park, A. Moon, E. Lee, and S. Kim, 'Understanding IoT climate Data based Predictive Model for Outdoor Smart Farm', in *2021 International Conference on Information and Communication Technology Convergence (ICTC),* Oct. 2021, pp. 1892–1894. doi: 10.1109/ICTC52510.2021.9620971.

[16] Priyadharshini. SP and P. Balamurugan, 'Unmanned Aerial Vehicle in the Smart Farming Systems: Types, Applications and Cyber-Security Threats', in *2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES),* Jul. 2022, pp. 1–9. doi: 10.1109/ICSES55317.2022.9914070.

[17] 'Architecture of Internet of Things (IoT)', GeeksforGeeks. Accessed: Jan. 27, 2023. [Online]. Available: https://www.geeksforgeeks.org/architecture-of-internet-of-things-iot/

[18] 'Internet of Things (IoT) Architecture: Key Layers and Components | AltexSoft'. Accessed: Jan. 27, 2023. [Online]. Available: https://www.altexsoft.com/blog/iot-architecture-layers-components/

[19] C. Gregersen, 'A Complete Rundown of the IoT Stack and Layers', Nabto. Accessed: Jan. 27, 2023. [Online]. Available: https://www.nabto.com/guide-to-iot-stack-layers/

[20] 'What Are Sensors and How Do They Work?', WhatIs.com. Accessed: Jan. 27, 2023. [Online]. Available: https://www.techtarget.com/whatis/definition/sensor

[21] 'Temperature Sensors: Types, How It Works, & Applications', Encardio Rite. Accessed: Jan. 27, 2023. [Online]. Available: https://www.encardio.com/blog/temperature-sensor-probe-types-how-it-works-applications

[22] D. Jost, 'What is a Humidity Sensor?', Fierce Electronics. Accessed: Jan. 28, 2023. [Online]. Available: https://www.fierceelectronics.com/sensors/what-a-humidity-sensor

[23] 'Definition of Light Sensor | Analog Devices'. Accessed: Jan. 28, 2023. [Online]. Available: https://www.analog.com/en/design-center/glossary/light-sensor.html

[24] Apure, 'What is pH sensor & How does it work? - Apure'. Accessed: Jan. 28, 2023. [Online]. Available: https://apureinstrument.com/blogs/about-ph-sensor/

[25] ‘Time Domain Reflectometry - an overview | ScienceDirect Topics’. Accessed: Jan. 28, 2023. [Online]. Available: https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/time-domain-reflectometry

[26] ‘Which cameras can be used for precision agriculture’, Support. Accessed: Jan. 28, 2023. [Online]. Available: https://support.pix4d.com/hc/en-us/articles/202559059-Which-cameras-can-be-used-for-precision-agriculture

[27] Admin, ‘Measure Soil Nutrient using Arduino & Soil NPK Sensor’, How To Electronics. Accessed: Jan. 28, 2023. [Online]. Available: https://how2electronics.com/measure-soil-nutrient-using-arduino-soil-npk-sensor/

[28] T. Folnović, ‘Farm Revolution - Sensors for Crop Pest Detection’, AGRIVI. Accessed: Jan. 28, 2023. [Online]. Available: https://www.agrivi.com/blog/farm-revolution-sensors-for-crop-pest-detection/

[29] ‘GPS Sensor Function | GPS Sensor Working’. Accessed: Jan. 28, 2023. [Online]. Available: https://www.rfwireless-world.com/Terminology/GPS-Sensor-function-and-working.html

[30] rdemko, ‘Understanding the 3 Types of Mechanical Actuators: Pneumatic, Hydraulic and Electric’, SDC Automation. Accessed: Jan. 29, 2023. [Online]. Available: https://sdcautomation.com/understanding-the-3-types-of-mechanical-actuators-pneumatic-hydraulic-and-electric/

[31] M. Burhan, R. A. Rehman, B. Khan, and B.-S. Kim, ‘IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey’, *Sensors*, vol. 18, no. 9, p. 2796, Aug. 2018, doi: 10.3390/s18092796.

[32] ‘IoT Protocols and Standards in 2022: A Comprehensive Guide’. Accessed: Jan. 29, 2023. [Online]. Available: https://www.kellton.com/kellton-tech-blog/your-complete-guide-to-iot-protocols-and-Standards-2022

[33] ‘A Comprehensive Guide to IoT Protocols | IoT Glossary’. Accessed: Feb. 01, 2023. [Online]. Available: https://www.emnify.com/iot-glossary/guide-iot-protocols

[34] ‘What is LoRaWAN? - AWS IoT Core’. Accessed: Feb. 01, 2023. [Online]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/connect-iot-lorawan-what-is-lorawan.html

[35] ‘Different Types of Wireless Communication Protocols in IOT’. Accessed: Feb. 01, 2023. [Online]. Available: https://iotdesignpro.com/articles/different-types-of-wireless-communication-protocols-for-iot

[36] ‘IoT Wireless Network Protocols’, Farnell. Accessed: Feb. 01, 2023. [Online]. Available: https://pt.farnell.com/iot-wireless-network-protocols

[37] ‘Top 12 most commonly used IoT protocols and standards | TechTarget’, IoT Agenda. Accessed: Feb. 01, 2023. [Online]. Available: https://www.techtarget.com/iotagenda/tip/Top-12-most-commonly-used-IoT-protocols-and-standards

[38] ‘An Introduction to Sigfox Technology – Basics, Architecture and Security Features’. Accessed: Feb. 01, 2023. [Online]. Available:

https://circuitdigest.com/article/what-is-sigfox-basics-architecture-and-security-features

[39] 'IoT Architecture - Detailed Explanation', InterviewBit. Accessed: Feb. 02, 2023. [Online]. Available: https://www.interviewbit.com/blog/iot-architecture/

[40] A. Simmons, 'Internet of Things (IoT) Architecture: Layers Explained', Dgtl Infra. Accessed: Feb. 02, 2023. [Online]. Available: https://dgtlinfra.com/internet-of-things-iot-architecture/

[41] A. Business, 'Real-Time Weather Monitoring System Using IoT', Airtel B2B Blog. Accessed: Feb. 03, 2023. [Online]. Available: https://www.airtel.in/blog/business/iot-use-cases-in-real-time-weather-monitoring-system/

[42] 'Soil Condition | Department of Natural Resources and Environment Tasmania'. Accessed: Feb. 03, 2023. [Online]. Available: https://nre.tas.gov.au/agriculture/land-management-and-soils/land-and-soil-resource-assessment/soil-condition

[43] 'IoT solutions for irrigation monitoring'. Accessed: Feb. 06, 2023. [Online]. Available: https://www.paessler.com/iot/irrigation-monitoring

[44] 'Waterproof_Capacitive_Soil_Moisture_Sensor_SKU_SEN0308-DFRobot'. Accessed: Aug. 27, 2023. [Online]. Available: https://wiki.dfrobot.com/Waterproof_Capacitive_Soil_Moisture_Sensor_SKU_SEN0308

[45] 'SHELLY DS18B20 Sensor de temperatura'. Accessed: Aug. 27, 2023. [Online]. Available: https://mauser.pt/catalog/product_info.php?products_id=096-8756

[46] 'The Things Gateway', The Things Network. Accessed: Sep. 18, 2023. [Online]. Available: https://www.thethingsnetwork.org/docs/gateways/gateway/

[47] 'Crop Recommendation Dataset'. Accessed: Feb. 02, 2023. [Online]. Available: https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset

Annexs

Annex A

```
from flask import Flask, jsonify, request, render_template
from flask_cors import CORS
import joblib
import paho.mqtt.client as mqtt
import sqlite3
from datetime import datetime, time
import json
#apikey ttn =
NNSXS.YO7KBNS5TGJDXTKW3TDBV6F7TSH66BPBNSFXUBQ.JOVSI6UXBHFVHVSHMQI5247MUKO3U2W

app = Flask(__name__)
CORS(app)

model = joblib.load("RandomForest.pkl")


mqtt_data = {
    'temperature': 0.0,
    'humidity': 0.0,
    'soil_humidity': 0.0,
    'soil_temperature': 0.0,
    'n': 0.0,
    'p': 0.0,
    'k': 0.0,
}
count=0
lock=0



def is_morning():
    now = datetime.now()
    if (now.time() < time(10, 0) and now.time() < time(9, 0)):
        return True
    else:
        return False

def on_connect2(mqttc, obj, flags, rc):
    print("\nConnect: rc = " + str(rc))

def on_connect1(client, userdata, flags, rc):
    if rc == 0:
        mqttClient.subscribe("dragino-1ed75c/2202439/data")
    else:
        print("Connection failed with error code:", rc)

def on_message1(client, userdata, message):
    payload = message.payload.decode('utf-8')
    topic = message.topic
    print("Received message:", payload, "on topic:", topic)

    data_components = payload.split('&')
    data = {}
    for component in data_components:
        key, value = component.split('=')
        data[key] = value
    global mqtt_data
    global count
    global lock
    mqtt_data['temperature'] = float(data.get('temp', 0.0))
    mqtt_data['humidity'] = float(data.get('hum', 0.0))
    mqtt_data['soil_humidity'] = float(data.get('soilhum', 0.0))
    mqtt_data['soil_temperature'] = float(data.get('soiltemp', 0.0))
    lock = lock + 1
    if (lock == 2):
        insert_mqtt_data(mqtt_data['temperature'], mqtt_data['humidity'],
mqtt_data['soil_humidity'], mqtt_data['soil_temperature'], mqtt_data['n'],
mqtt_data['p'], mqtt_data['k'])


def insert_mqtt_data(temperature, humidity, soil_humidity,
soil_temperature, n , p, k):
    print("hello")
    global lock
    global count
    lock=0
    if is_morning() ==False :
        count = 1

        conn = sqlite3.connect('mqtt_data.db')
        cursor = conn.cursor()
        cursor.execute('''
            INSERT INTO mqtt_data (temperature, humidity, soil_humidity,
soil_temperature, n, p, k)
            VALUES (?, ?, ?, ?, ? ,?, ?)
        ''', (temperature, humidity, soil_humidity,soil_temperature, n, p,
k))
        conn.commit()
        conn.close()

def getData(someJSON):
    global mqtt_data
    uplink_message = someJSON["uplink_message"]
    decoded_payload = uplink_message["decoded_payload"]
    mqtt_data['n'] = decoded_payload["n"]
    mqtt_data['p'] = decoded_payload["p"]
    mqtt_data['k'] = decoded_payload["k"]
    print(mqtt_data)
def on_message2(mqttc, obj, msg):

    if "count" not in on_message2.__dict__:
        on_message2.count = 0

    words = msg.topic.split("/")
    last_word = words[-1]

    if last_word == 'up':
        on_message2.count += 1
        print(f"Count: {on_message2.count}")
        if on_message2.count % 2 == 0:
            parsedJSON = json.loads(msg.payload)
            getData(parsedJSON)
            global lock
            lock = lock + 1
            if (lock == 2):
                insert_mqtt_data(mqtt_data['temperature'],
mqtt_data['humidity'], mqtt_data['soil_humidity'],
mqtt_data['soil_temperature'], mqtt_data['n'], mqtt_data['p'],
mqtt_data['k'])

@app.route('/')
def index():

    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)

    humidity = float(data['humidity'])
    n = float(data['n'])
    p = float(data['p'])
    k = float(data['k'])
    temperature = float(data['temperature'])

    prediction = model.predict([[humidity] + [n] + [p] + [k] +
[temperature]])[0]

    return jsonify({'prediction': prediction})

@app.route('/chartdata', methods=['GET'])
def getchart_data():
    try:
        db = sqlite3.connect('mqtt_data.db')
        cursor = db.cursor()
        cursor.execute('SELECT timestamp, temperature, humidity,
soil_humidity,soil_temperature, n, p, k FROM mqtt_data ')
        data = cursor.fetchall()

        result = []
        for row in data:
            result.append({
                'timestamp': row[0],
                'temperature': row[1],
                'humidity': row[2],
                'soil_humidity': row[3],
                'soil_temperature': row[4],
                'n' : row[5],
                'p' : row[6],
                'k' : row[7],
            })
        cursor.close()
        db.close()
        return jsonify(result)

    except Exception as e:
        return jsonify(error=str(e)), 500

@app.route('/mqttdata', methods=['GET'])
def get_mqtt_data():
    return jsonify(mqtt_data)

User = "my-app-2000@ttn"
Password =
"NNSXS.2SRZDRU5C6UD32MDHYKSFW7TFBOEGBKWNQESKCI.3EJOGRSVJ5TSGT7QP2M7FXEW7TNIQC

theRegion = "EU1"
mqttc = mqtt.Client(client_id="1")
mqttc.on_connect = on_connect2
mqttc.on_message = on_message2
mqttc.username_pw_set(User, Password)
mqttc.tls_set()
mqttc.connect(theRegion.lower() + ".cloud.thethings.network", 8883, 60)
mqttc.subscribe("#", 0)
mqttc.loop_start()

mqttBrokerIp = "broker.hivemq.com"
mqttBrokerPort = 1883
mqttClient = mqtt.Client(client_id="2")

mqttClient.on_connect = on_connect1
mqttClient.on_message = on_message1

mqttClient.connect(mqttBrokerIp, mqttBrokerPort)

mqttClient.loop_start()

if __name__ == '__main__':
    app.run()
```

Annex B

```
const userDisplay=document.getElementById('userDisplay');
const
temperatureValue=document.getElementById('temperatureValue');
const
soilHumidityValue=document.getElementById('soilHumidityValue');
const humidityValue=document.getElementById('humidityValue');


const fecthSqlData = async () =>{
fetch('http://localhost:5000/chartdata')
.then(response => response.json())
.then(data => {
  const dateParts = [];
  const timestamps = data.map(item => item.timestamp);
  for (const dateTimeString of timestamps) {
    const dateTime = new Date(dateTimeString);
    const datePart = `${dateTime.getMonth() +
1}/${dateTime.getDate()}`;
    dateParts.push(datePart);
  }
  const temperatures = data.map(item => item.temperature);
  const humidity = data.map(item => item.humidity);
  const soilHumidity = data.map(item => item.soil_humidity);
  const soilTemperature = data.map(item =>
item.soil_temperature);
  const n = data.map(item => item.n);
  const p = data.map(item => item.p);
  const k = data.map(item => item.k);
  const ctx1 =
document.getElementById('temperatureChart').getContext('2d');
  const temperatureChart = new Chart(ctx1, {
    type: 'line',
    data: {
      labels: dateParts,
      datasets: [
        {
          label: 'Temperature',
          data: temperatures,
          borderColor: 'yellow',
          fill: false,
        },
      ],
    },
  });
  const ctx2 =
document.getElementById('humidityChart').getContext('2d');
  const humidityChart = new Chart(ctx2, {
    type: 'line',
    data: {
      labels: dateParts,
      datasets: [
        {
          label: 'Humidity',
          data: humidity,
          borderColor: 'blue',
          fill: false,
        },
      ],
    },
  });

  const ctx3 =
document.getElementById('soilHumidityChart').getContext('2d');
  const soilHumidityChart = new Chart(ctx3, {
    type: 'line',
    data: {
      labels: dateParts,
      datasets: [
        {
          label: 'Soil Humidity',
          data: soilHumidity,
          borderColor: 'blue',
          fill: false,
        },
      ],
    },
  });
  const ctx4 =
document.getElementById('soilTemperatureChart').getContext('2d');

  const soilTemperatureChart = new Chart(ctx4, {
    type: 'line',
    data: {
      labels: dateParts,
      datasets: [
        {
          label: 'soilTemperature',
          data: soilTemperature,
          borderColor: 'yellow',
            fill: false,
          },
        ],
      },
    });
    const ctx5 =
document.getElementById('nChart').getContext('2d');
    const nChart = new Chart(ctx5, {
      type: 'line',
      data: {
        labels: dateParts,
        datasets: [
          {
            label: 'Nitrogen',
            data: n,
            borderColor: 'green',
            fill: false,
          },
        ],
      },
    });
    const ctx6 =
document.getElementById('pChart').getContext('2d');
    const pChart = new Chart(ctx6, {
      type: 'line',
      data: {
        labels: dateParts,
        datasets: [
          {
            label: 'Phosphorus',
            data: p,
            borderColor: 'red',
            fill: false,
          },
        ],
      },
    });
    const ctx7 =
document.getElementById('kChart').getContext('2d');
    const kChart = new Chart(ctx7, {
      type: 'line',
      data: {
        labels: dateParts,
        datasets: [
          {
```

```
          label: 'Potassium',
          data: k,
          borderColor: 'gray',
          fill: false,
        },
      ],
    },
  });
})
.catch(error => console.error('Error fetching chart data:',
error));
}

const fetchMqttData = async () => {
    try {
        const response = await
fetch('http://localhost:5000/mqttdata');
        if (!response.ok) {
            throw new Error('Failed to fetch MQTT data');
        }
        const data = await response.json();

        const temperature = data.temperature;
        const humidity = data.humidity;
        let soilHumidity = data.soil_humidity;
        const soilTemperature = data.soil_temperature;
        const n = data.n
        const p = data.p
        const k = data.k

        const nValue = n / (n + p + k) * 100
        const pValue = p / (n + p + k) * 100
        const kValue = k / (n + p + k) * 100


        document.getElementById('temperatureValue').innerText =
temperature  + "ยบC";
        document.getElementById('humidityValue').innerText =
humidity  + "%";
        document.getElementById('soilHumidityValue').innerText
= soilHumidity;

document.getElementById('soilTemperatureValue').innerText =
soilTemperature  + "ยบC";
        document.getElementById('nValue').innerText = nValue
```

```
=== "NaN" ? nValue.toFixed(1) + "%" : 0 + "%";
        document.getElementById('pValue').innerText = pValue
=== "NaN" ? pValue.toFixed(1)  + "%" : 0 + "%";
        document.getElementById('kValue').innerText = kValue
=== "NaN" ? kValue.toFixed(1)  + "%" : 0 + "%";


        const airHumidityValue = 655;
        const watercupHumidityValue = 89;
        const intervals =
(airHumidityValue-watercupHumidityValue)/3;
        let humidityText = "";
        let img;

        if(soilHumidity > watercupHumidityValue && soilHumidity
< (watercupHumidityValue + intervals)){
          humidityText="Very Wet!"
          img="img/verywet.png"
        }
        else if(soilHumidity > (watercupHumidityValue +
intervals) && soilHumidity < (airHumidityValue - intervals))
        {
          humidityText="It is Wet!";
          img="img/wet.png"
        }
        else if(soilHumidity < airHumidityValue && soilHumidity
> (airHumidityValue - intervals))
        {
          humidityText="Time to Water!";
          img="img/dry.png"
        }


        document.getElementById('temperatureValue').innerText =
temperature  + "ยบC";
        document.getElementById('humidityValue').innerText =
humidity  + "%";
        document.getElementById('soilHumidityValue').innerText
= soilHumidity;

document.getElementById('soilTemperatureValue').innerText =
soilTemperature  + "ยบC";
        document.getElementById('nValue').innerText = nValue
=== "NaN" ? nValue.toFixed(1) + "%" : 0 + "%";
        document.getElementById('pValue').innerText = pValue
```

```
=== "NaN" ? pValue.toFixed(1)  + "%" : 0 + "%";
        document.getElementById('kValue').innerText = kValue
=== "NaN" ? kValue.toFixed(1)  + "%" : 0 + "%";
        document.getElementById('humidityText').innerText =
humidityText;
        document.getElementById('humidityImg').src = img;




    } catch (error) {
        console.error(error);
    }
};

const loggedInUser=localStorage.getItem('user');

userDisplay.textContent+=loggedInUser + '!';
fetchMqttData();
fecthSqlData();

document.addEventListener("DOMContentLoaded", function() {
  const contentContainer = document.getElementById("content-
container");
  const navLinks = document.querySelectorAll(".navbar .navbar-
nav .nav-link");

  navLinks.forEach(link => {
    link.addEventListener("click", function(event) {
      event.preventDefault();
      const targetSection = this.dataset.target;
      const selectedSection =
document.getElementById(targetSection);

      document.querySelectorAll("section").forEach(section => {
        section.style.display = "none";
      });

      selectedSection.style.display = "block";
    });
  });
});

$(document).ready(function() {
  function showSection(section) {
    $(".nav-link").removeClass("active");
    $('[data-target="' + section + '"]').addClass("active");
    $("section").hide();
    $("#" + section).show();
  }

  showSection("section1");

  $(".nav-link").click(function(event) {
    event.preventDefault();
    var target = $(this).data("target");
    showSection(target);
  });
});

setTimeout(function () {
  window.dispatchEvent(new Event('resize'));
}, 3000);


if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(function(position) {
    const lat = 38.917729;
    const lon = -8.675921;
    var map = L.map('map').setView([lat, lon], 17);


L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 20,
    }).addTo(map);
    var marker = L.marker([lat, lon]).addTo(map);

    var circle = L.circle([lat, lon], {
      color: 'red',
      fillColor: '#f03',
      fillOpacity: 0.5,
      radius: 10
    }).addTo(map);
  });
} else {
  console.log("Geolocation is not supported by this browser.");
}
```

Annex C

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link rel="shortcut icon" type="image/png" href="img/icon.png" />

    <link
      href="https://fonts.googleapis.com/css?
family=Poppins:300,400,500,600&display=swap"
      rel="stylesheet"
    />
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css
rel="stylesheet" integrity="sha384-
4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9"
crossorigin="anonymous">
    <link rel="stylesheet" href="dashboard.css" />
    <title>Dashboard</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js
integrity="sha384-
xorO1P97wJ2iCw9zlwUvK6y8S08i1I8m/yhznEnm2srmjq6LHJsRvtn3p69KdOIv"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.min.js" integrity="sha384-
ejm14v8qOq4E8y0OgVXWVc1+5ayJ1txxL2T4fQa5zmz0K82GB9zB1NWCg2o1l75O"
crossorigin="anonymous"></script>
    <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
    integrity="sha256-p4NxAoJBhIIN+hmNHrzRCf9tD/miZyoHS5obTRR9BMY="
    crossorigin=""/>
    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"
    integrity="sha256-20nQCchB9co0qIjJZRGuk2/Z9VM+kNiyxNV1lvTlZBo="
    crossorigin=""></script>

    <script defer src="dashboard.js"></script>
  </head>
  <body>
    <div class="overflow-hidden">
      <nav class="navbar navbar-expand-lg navbar-dark bg-white">
        <div class="container">
          <!-- Left-aligned text in the menu -->
          <span id="userDisplay" class="navbar-brand">Welcome back </span>

          <!-- Centered menu items -->
          <div class="navbar-nav mx-auto ">
```

```
              <a class="nav-link" href="#" data-target="section1">Home</a>
              <a class="nav-link" href="#" data-target="section2">Data</a>
              <a class="nav-link" href="#" data-target="section3">Location
and Status</a>
              <!-- Add more links as needed -->
            </div>

            <!-- Right-aligned content -->
            <div class="navbar-nav ml-auto">
              <!-- Add any right-aligned content here -->
            </div>
          </div>
        </nav>
    <section id="section1">
      <div class="card-container">
        <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
          <div class="card-body p-4">

            <div id="demo1" class="carousel slide" data-ride="carousel">
              <!-- Carousel inner -->
              <div class="carousel-inner">
                <div class="carousel-item active">
                  <div class="d-flex justify-content-between mb-4 pb-2">
                    <div>
                      <h2 id="temperatureValue" class="display-2 display-
size"><strong></strong></h2>
                      <p class="text-muted mb-0">Temperature</p>
                    </div>
                    <div>
                      <img src="https://mdbcdn.b-cdn.net/img/Photos/new-
templates/bootstrap-weather/ilu3.webp"
                        width="150px">
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
        <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
          <div class="card-body p-4">

            <div id="demo1" class="carousel slide" data-ride="carousel">
              <!-- Carousel inner -->
              <div class="carousel-inner">
                <div class="carousel-item active">
                  <div class="d-flex justify-content-between mb-4 pb-2">
                    <div>
                      <h2 id="humidityValue" class="display-2 display-size">
<strong></strong></h2>
                      <p class="text-muted mb-0">Humidity</p>
                    </div>
                    <div>
                      <i class="fas fa-tint fa-5x mt-3" style="color:
```

```
#3be8e8;width:150px !important ;height:100px !important"></i>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
      <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
        <div class="card-body p-4">

          <div id="demo1" class="carousel slide" data-ride="carousel">

            <!-- Carousel inner -->
            <div class="carousel-inner">
              <div class="carousel-item active">
                <div class="d-flex justify-content-between mb-4 pb-2">
                  <div>
                    <h2 id="soilHumidityValue" class="display-2 display-
size"><strong></strong></h2>
                    <p class="text-muted mb-0">Soil Humidity</p>
                  </div>
                  <div>
                    <i class="fas fa-tint fa-5x mt-3" style="color:
#3be8e8;width:150px !important ;height:100px !important"></i>
                  </div>
                </div>
              </div>
            </div>
          </div>

        </div>
      </div>
      <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
        <div class="card-body p-4">

          <div id="demo1" class="carousel slide" data-ride="carousel">
            <!-- Carousel inner -->
            <div class="carousel-inner">
              <div class="carousel-item active">
                <div class="d-flex justify-content-between mb-4 pb-2">
                  <div>
                    <h2 id="soilTemperatureValue" class="display-2 display-
size"><strong></strong></h2>
                    <p class="text-muted mb-0">Soil Temperature</p>
                  </div>
                  <div>
                    <img src="https://mdbcdn.b-cdn.net/img/Photos/new-
templates/bootstrap-weather/ilu3.webp"
                      width="150px">
                  </div>
                </div>
              </div>
            </div>
          </div>

        </div>
      </div>
      <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
        <div class="card-body p-4">

          <div id="demo1" class="carousel slide" data-ride="carousel">
            <!-- Carousel inner -->
            <div class="carousel-inner">
              <div class="carousel-item active">
                <div class="d-flex justify-content-between mb-4 pb-2">
                  <div>
                    <h2 id="nValue" class="display-2 display-size"><strong>
</strong></h2>
                    <p class="text-muted mb-0">Nitrogen</p>
                  </div>
                  <div>
                    <i class="fas fa-flask fa-5x mt-3" style="color:
green;width:150px !important ;height:100px !important"></i>
                  </div>
                </div>
              </div>
            </div>
          </div>

        </div>
      </div>
      <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
        <div class="card-body p-4">

          <div id="demo1" class="carousel slide" data-ride="carousel">
            <!-- Carousel inner -->
            <div class="carousel-inner">
              <div class="carousel-item active">
                <div class="d-flex justify-content-between mb-4 pb-2">
                  <div>
                    <h2 id="pValue" class="display-2 display-size"><strong>
</strong></h2>
                    <p class="text-muted mb-0">Phosphorus</p>
                  </div>
                  <div>
                    <i class="fas fa-flask fa-5x mt-3" style="color:
red;width:150px !important ;height:100px !important"></i>
                  </div>
                </div>
              </div>
            </div>
          </div>

        </div>
      </div>
      <div class="card gradient-custom" style="border-radius: 25px;">
        <div class="card-body p-4">
```

```
          <div id="demo1" class="carousel slide" data-ride="carousel">
            <!-- Carousel inner -->
            <div class="carousel-inner">
              <div class="carousel-item active">
                <div class="d-flex justify-content-between mb-4 pb-2">
                  <div>
                    <h2 id="kValue" class="display-2 display-size display-
size"><strong></strong></h2>
                    <p class="text-muted mb-0">Potassium</p>
                  </div>
                  <div>
                    <i class="fas fa-flask fa-5x mt-3" style="color:
gray;width:150px !important ;height:100px !important"></i>
                  </div>
                </div>
              </div>
            </div>
          </div>

        </div>
      </div>
    </div>
  </section>

  <!-- Section 2 content -->
  <section id="section2" style="display:none;">
    <div class="card-container">
    <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
      <div class="card-body p-4">
        <canvas id="temperatureChart" width="10" height="10"></canvas>
      </div>
    </div>
    <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
      <div class="card-body p-4">
        <canvas id="humidityChart" width="10" height="10"></canvas>
      </div>
    </div>
    <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
      <div class="card-body p-4">
        <canvas id="soilHumidityChart" width="10" height="10"></canvas>
      </div>
    </div>
    <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
      <div class="card-body p-4">
        <canvas id="soilTemperatureChart" width="10" height="10"></canvas>
      </div>
    </div>
    <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
      <div class="card-body p-4">
        <canvas id="nChart" width="10" height="10"></canvas>
      </div>
    </div>
    <div class="card gradient-custom" style="border-radius: 25px;">
```

```
      <div class="card-body p-4">
        <canvas id="pChart" width="10" height="10"></canvas>
      </div>
    </div>
    <div class="card gradient-custom" style="border-radius: 25px;">
      <div class="card-body p-4">
        <canvas id="kChart" width="10" height="10"></canvas>
      </div>
    </div>
    <div class="flex" style="height: 300px; width: 400px;" >
      <canvas id="temperatureChart" width="10" height="10"></canvas>
      <canvas id="humidityChart" width="10" height="10"></canvas>
      <canvas id="soilHumidityChart" width="10" height="10"></canvas>
    </div>
  </div>
  </section>

  <!-- Section 3 content -->
  <section id="section3" style="display:none;">
    <div id="map"></div>
    <div class="card-container justify-content-center mt-3">
      <div class="card mb-3 gradient-custom" style="border-radius: 25px;">
      <div class="card-body p-4">

        <div id="demo1" class="carousel slide" data-ride="carousel">
          <!-- Carousel inner -->
          <div class="carousel-inner">
            <div class="carousel-item active">
              <div class="d-flex justify-content-between mb-4 pb-2">
                <div>
                  <h2 id="humidityText" class="display-2 display-size">
<strong></strong></h2>
                  <p class="text-muted mb-0">SoilStatus</p>
                </div>
                <div>
                  <img id="humidityImg" src="https://mdbcdn.b-
cdn.net/img/Photos/new-templates/bootstrap-weather/ilu3.webp"
                    width="150px">
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
      </div>
    </div>

  </section>
</div>
  </body>
</html>
```

Annex D

```
body {
     font-family: 'Poppins', sans-serif;
     font-weight: 300;
     color: #444;
     line-height: 1.9;
     background-color: #f3f3f3;
   }
.navbar .navbar-brand{
     color: black; /* Text color */
}
#map {
   width: 600px;
   height: 300px;
   margin: 0 auto;
}
.nav-link{
     color: black;
}
.navbar-dark .navbar-brand .nav-link{
     color: orange;
}
.navbar-dark .navbar-nav .nav-link:hover {
     border-bottom: 3px solid orangered;
     color: orangered;
}
.navbar-nav .nav-link.active{
     border-bottom: 3px solid orangered;
     color: orangered;
}

#section1 {
     display: flex;
     justify-content: center;
```

```
      flex-wrap: wrap;
      align-items: flex-start;
      min-height: 90vh;
      padding-top: 20px;
      overflow-y: hidden;
    }
    #section2 {
      display: flex;
      justify-content: center;
      flex-wrap: wrap;
      align-items: flex-start;
      min-height: 100vh;
      padding-top: 20px;
    }
    #section3 {
      display: flex;
      justify-content: center;
      flex-wrap: wrap;
      align-items: flex-start;
      min-height: 100vh;
      padding-top: 20px;
    }

    .card-container {
      display: flex;
      justify-content: space-between;
      max-width: 1200px;
      margin: 0 auto;
      flex-wrap: wrap;
    }
    .display-size{
      font-size: 3rem !important;
    }
```

```css
.card {
  border: 1px solid #ccc;
  border-radius: 25px;
  width: calc(16.666% - -193px);
  text-align: center;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
```

Annex E

```
import pandas as pd
import numpy as np
import random
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import json
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots

from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn import tree
import warnings
warnings.filterwarnings('ignore')

df=pd.read_csv("Crop_recommendation.csv")
json_data = df.to_json(orient='records', indent=2)

# Write JSON to a file
with open('data.json', 'w') as jsonfile:
    jsonfile.write(json_data)

print("Number of various crops: ", len(df['label'].unique()))
print("List of crops: ", df['label'].unique())

crop_summary = pd.pivot_table(df,index=['label'],aggfunc='mean')
crop_summary.head()

crop_summary_N = crop_summary.sort_values(by='N', ascending=False)

fig = make_subplots(rows=1, cols=2)

top = {
    'y' : crop_summary_N['N'][0:11].sort_values().index,
    'x' : crop_summary_N['N'][0:11].sort_values()
}

last = {
    'y' : crop_summary_N['N'][-11:].index,
    'x' : crop_summary_N['N'][-11:]
}

fig.add_trace(
    go.Bar(top,
           name="Most nitrogen required",
           marker_color='#0592D0',
           orientation='h',
           text=top['x']),

    row=1, col=1
)

fig.add_trace(
    go.Bar(last,
            name="Least nitrogen required",
            marker_color='#Cd7f32',
            orientation='h',
           text=last['x']),
    row=1, col=2
)
fig.update_layout(title_text="Nitrogen (N)",
                  plot_bgcolor='white')

fig.show()

crop_summary_P = crop_summary.sort_values(by='P', ascending=False)

fig = make_subplots(rows=1, cols=2)

top = {
    'y' : crop_summary_P['P'][0:11].sort_values().index,
    'x' : crop_summary_P['P'][0:11].sort_values()
}

last = {
    'y' : crop_summary_P['P'][-11:].index,
    'x' : crop_summary_P['P'][-11:]
}

fig.add_trace(
    go.Bar(top,
            name="Most phosphorus required",
            marker_color='#Bdb76b',
            orientation='h',
           text=top['x']),

    row=1, col=1
)

fig.add_trace(
    go.Bar(last,
            name="Least phosphorus required",
            marker_color='#E97451',
            orientation='h',
           text=last['x']),
    row=1, col=2
```

```
)
fig.update_traces(texttemplate='%{text}', textposition='inside')
fig.update_layout(title_text="Phosphorus (P)",
                  plot_bgcolor='white')
fig.show()

crop_summary_K = crop_summary.sort_values(by='K', ascending=False)

fig = make_subplots(rows=1, cols=2)

top = {
    'y' : crop_summary_K['K'][0:11].sort_values().index,
    'x' : crop_summary_K['K'][0:11].sort_values()
}

last = {
    'y' : crop_summary_K['K'][-11:].index,
    'x' : crop_summary_K['K'][-11:]
}

fig.add_trace(
    go.Bar(top,
           name="Most potassium required",
           marker_color='#954535',
           orientation='h',
          text=top['x']),

    row=1, col=1
)

fig.add_trace(
    go.Bar(last,
           name="Least potassium required",
           marker_color='#C2b280',
           orientation='h',
          text=last['x']),
    row=1, col=2
)
fig.update_traces(texttemplate='%{text}', textposition='inside')
fig.update_layout(title_text="Potassium (K)",
                  plot_bgcolor='white')
fig.show()

features = df[['N', 'P','K','temperature', 'humidity']]
target = df['label']
labels = df['label']

acc = []
model = []
```

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest =
train_test_split(features,target,test_size = 0.2,random_state =2)

from sklearn.tree import DecisionTreeClassifier

DecisionTree =
DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=5)


DecisionTree.fit(Xtrain,Ytrain)

predicted_values = DecisionTree.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Decision Tree')
print("DecisionTrees's Accuracy is: ", x*100)

print(classification_report(Ytest,predicted_values))

from sklearn.model_selection import cross_val_score

score = cross_val_score(DecisionTree, features, target,cv=5)
score

import pickle
# Dump the trained Naive Bayes classifier with Pickle
DT_pkl_filename = 'DecisionTree.pkl'
# Open the file to save as pkl file
DT_Model_pkl = open(DT_pkl_filename, 'wb')
pickle.dump(DecisionTree, DT_Model_pkl)
# Close the pickle instances
DT_Model_pkl.close()

#Guassian Naive Bayes
from sklearn.naive_bayes import GaussianNB

NaiveBayes = GaussianNB()

NaiveBayes.fit(Xtrain,Ytrain)

predicted_values = NaiveBayes.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Naive Bayes')
print("Naive Bayes's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
# Cross validation score (NaiveBayes)
score = cross_val_score(NaiveBayes,features,target,cv=5)
score

import pickle
# Dump the trained Naive Bayes classifier with Pickle
NB_pkl_filename = 'NBClassifier.pkl'
# Open the file to save as pkl file
NB_Model_pkl = open(NB_pkl_filename, 'wb')
pickle.dump(NaiveBayes, NB_Model_pkl)
# Close the pickle instances
NB_Model_pkl.close()

#Support Vector Machine (SVM)
from sklearn.svm import SVC

SVM = SVC(gamma='auto')

SVM.fit(Xtrain,Ytrain)

predicted_values = SVM.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('SVM')
print("SVM's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))

# Cross validation score (SVM)
score = cross_val_score(SVM,features,target,cv=5)
score

#Logistic Regression
from sklearn.linear_model import LogisticRegression

LogReg = LogisticRegression(random_state=2)

LogReg.fit(Xtrain,Ytrain)

predicted_values = LogReg.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Logistic Regression')
print("Logistic Regression's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
```

```
# Cross validation score (Logistic Regression)
score = cross_val_score(LogReg,features,target,cv=5)
score

import pickle
# Dump the trained Naive Bayes classifier with Pickle
LR_pkl_filename = 'LogisticRegression.pkl'
# Open the file to save as pkl file
LR_Model_pkl = open(DT_pkl_filename, 'wb')
pickle.dump(LogReg, LR_Model_pkl)
# Close the pickle instances
LR_Model_pkl.close()

#Random Forest
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=20,
random_state=0).fit(Xtrain,Ytrain)

predicted_values = RF.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('RF')
print("RF's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))
# Cross validation score (Random Forest)
score = cross_val_score(RF,features,target,cv=5)
score

import pickle
import joblib
# Dump the trained Naive Bayes classifier with Pickle
RF_pkl_filename = 'RandomForest.pkl'
# Open the file to save as pkl file
RF_Model_pkl = open(RF_pkl_filename, 'wb')
pickle.dump(RF, RF_Model_pkl)
# Close the pickle instances
RF_Model_pkl.close()
joblib.dump(RF,'random_forest_model.joblib')

plt.figure(figsize=[10,5],dpi = 100)
plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
sns.barplot(x = acc,y = model,palette='dark')
```

```
data = np.array([[100,18, 20, 23.603016, 60.3]])
prediction = NaiveBayes.predict(data)
print(prediction)
```