FarmRisk

An intelligent manager for perimetral crop protection

DIEGO ACEITUNO
RENATO LUIGI BARRA
DENNIS TANDA

Architectures and Service Platforms: Final Project
February 2, 2025 v2.0

Contents

List of Figures

List of Tables

Glosary

5G Fifth Generation of mobile communications 8

BLE Bluetooth Low Energy 28

IoT Internet of Things 1, 4, 5, 9, 49

IR InfraRed 6

ISM Industrial, Scientific and Medical 5

LLN Low power lossy Networks 27

LoRaWAN Long Range Wide Area Network 5, 27, 28, 49

MCU MicroController Unit 9, 30, 49

MEC Mobile Edge Computing 1, 8, 49

MVVM Model-View-ViewModel 40

PIR Passive InfraRed 7, 22, 49

PPM Parts Per Million 33

RFID Radio Frequency Identification 6

RPC Remote Procedure Call 39

SaaS Software as a Service 8

UV UltraViolet 6

1 Introduction

In recent years, *Smart Farming* projects have been growing in popularity as a method to solve the challenges that the agriculture sector is facing in the last decades. This projects are based around the IoT (Internet of Things) domain, in which sensors and actuators are connected and coordinated to obtain data to represent the real world, this data can later be processed to make decisions that will make the farmers life easier.

One of the problems that *Smart Farming* deals with is the real-time protection of crops from different risks. This risks can go from natural risks such as wild fires, to wild animals trespassing fences to feed on the crops. IoT systems can also be applied to this problems, detecting in real-time these situations and obtaining all the information related to these events.

Considering that the topic requires fast-response times, new technologies such as MEC (Mobile Edge Computing) in combination with *Cloud Computing* can be applied to process the data as closest as possible to the main use case. This Edge solutions can aggregate all the data from the sensors and, with the help of algorithms for decision making, calculate the best possible response in the fastest way possible.

Using this technologies, this document presents the design of a system for the risks management in crops of Madrid, Spain.

These rural areas in the lasts decades have been declining in population. This creates two problems in relation to fire risks:

1. The depopulation of land creates areas with higher risk of fires[1], where intentional fires can be created and extend faster due to the heavy amount of fuel.
2. Farmers now need to be aware of these areas in case a fire appears.

In addition to the above mentioned, the risk of animal intrusion increases[2], as the perimetral limits of this crops is now uncontrolled.

To achieve the objective mentioned above, IoT technologies can be integrated with edge computing to create a fast-response system to protect farms against risks related to fires and wild animal intrusions.

2 System Description

The FarmRisk system can be divided considering the main two risks presented before:

- The risk of fire hazards in the vicinities of the farm need to be detected, in order to ensure save operations. To do this, fire and smoke sensors are used to detect early signs of fire. This data will be sent to the middleware and to a decision system that will infer the presence of fire. If any fire is detected, the farmer will be notified, indicating the proximity and the location of the detection.
- Secondly, if any signs of wild animal presence are detected near the limits of the farm, information will be sent to the edge platform. This information will be used to decide the possible actuator execution. If it is decided that an animal has been detected, it will be notified to the user and a smart speaker will be activated in the node. To detect this presence, sensors of presence and accelerometers are used, this allows for the coverage of bigger distances without the need of extra budget.

To integrate all the sensing of the farm's perimeter, an specific node integrates all the sensor's information and is in charge of sending the information to the communications gateway. This sensor will be placed every $50m$ in the perimeter.

This information will be forwarded to the edge computing platform, implemented using Thingsboard. When the data arrives, it is processed (while being logged) in order to determine possible actions or alarms that need to be executed.

The information will be accessible both from the edge computing platform and from and android mobile app.

All this situations and alarms are logged in a knowledge extraction database separated from the main edge platform. This information can be used to apply statistical learning over temporal data. This can be usedto create future solutions based on artificial intelligence.

Finally, the system will allow the possibility of two type of users: admin or worker. The worker will only be able to receive alerts and see the current status of the system, while the admin user will be able to command the network, stopping the alert system, responding to alerts, etc.

3 Project Scope

This project addresses multiple use cases on a farm, which will be detailed in the following chapters through UML diagrams and a Project Requirement Analysis. However, due to time and resource constraints, it wasn't feasible to implement the entire project with all its functionalities. Consequently, the project incorporates both real and simulated hardware.

A real embedded system was developed, equipped with PIR sensors for presence detection, accelerometers to monitor movements on the crop field fences, smoke sensors to detect will fires close to the system, and a GPS module to provide information about the device's current location.

To complement this, the system will simulate hardware responsible for fire detection. By using software simulations of fire detection sensors, combined with data on climatic conditions and physical soil factors, it will be possible to validate alerts generated in response to potential wildfire risks or active fire detections.

Due to the project's limitations and scope, the proof of concept will be conducted on a small scale, focusing on achieving the course objectives. These objectives will be addressed by utilizing both real and simulated hardware, where periodic events will be sent regarding sensed conditions, as well as sporadic events in the event of potential animal detection via the PIR sensor or accelerometers on the fences. Moreover, the project will include alarm systems to detect animal presence, imminent wildfire risks, or the occurrence of an active fire. This approach ensures the main goals of the course are met.

Finally, as an extra objective for this project, a mobile application was designed to enhance the usability of the system and to inform users as fast as possible in case any event appears.

4 Previous Works

Some of the topics covered in this project follow some existing research lines:

- Wireless sensor networks.
- Fire detection sensors.
- Animal intrusion detection.

In relation to this, there are already some examples of working projects that focus on protecting the crops from the external risks mentioned in earlier chapters. These projects achieve solutions using current technologies, which will be analyzed in this chapter to understand the *state of the art*.

The analysis of this projects results in a classification into three main types of solution design: IoT-based systems, solutions that incorporate AI, and finally hybrid systems.

4.1 Type of systems

IoT-based solutions

The Internet of Things has become an important tool for detecting risks. Sensors and embedded systems are used to collect real-time information and analyze the presence of wild animals or fires in the proximity of controlled areas.

These solutions architecture contain the next elements:

- A set of intercommunicated nodes that obtain information through and send data through a wireless network.
- A middleware layer to operate and integrate all this information for the user-level applications.
- End-User application layer.

AI-based solutions

Solutions using AI and machine learning are effective for animal classification and recognition, and this usage has been extended to detect behavior, like farm intrusions[3]. These systems are typically designed to process images collected by cameras and maybe additional information from sensors, identifying patterns that allow different species or even animals and humans to be distinguished.

For instance, algorithms such as YOLOv8 have proven highly effective at detecting animals in real-time using image analysis[4]. These systems divide images into grids, simultaneously predict bounding boxes, and assign probability to detected classes. These tools can also be used to process thermal images and detect fires.

The main drawback of these systems is the training stage with large pre-labeled data sets, needing a preparation of training data in order to achieve the results expected. This step is also very heavy in power consumption, and, depending on the objective, it may not be needed.

Hybrid solutions

This is the main paradigm being followed by the IoT industry in order to solve problems. These architectures integrate AI capabilities into the middleware to offer more complete and adaptable solutions.

An example of this kind of system is a project that combines PIR sensors and thermal cameras to collect data. This data is processed in the middleware by machine learning models to classify the type of animal detected, reducing the number of false positives and optimizing response, as specific measures can be applied depending on the identified species[5].

4.2 Wireless Sensor Networks

To monitor the presence of animals or fires, different devices that include sensors can be interconnected with each other to obtain information of the real world.

To transmit this data, different wireless technologies are used, and there can be more that one technology for any implementation, as it is device-dependent.

The technology selected depend on different parameters:

- Speed needed.
- Need for a license-free environment, for example, technologies like LoRaWAN (Long Range Wide Area Network) or `SigFox` use the ISM (Industrial, Scientific and Medical) band, while $4G$ and $5G$ technologies use non-free bands.
- Maximum number of nodes interconnected.
- Size of data sent and period of that data.
- Is low power consumption needed on the node?.
- Some technologies need GateWays, while others only need the buy of the end node.

Some of the mainly used technologies are Wi-Fi, mobile networks (LTE-M or NB-IoT), LoRaWAN or SigFox. Some of the key characteristics of this technologies can be seen in the next table[6][7][8].

Param	**LoRaWAN**	**SigFox**	**NB-IOT**	**Wi-Fi**
Range	$\leq$ 5km (Urban) $\leq$ 15km (Rural)	$\leq$ 10km (Urban) $\leq$ 50km (Rural)	$\leq$ 15 km	$\leq$ 40 m
Licensed band?	No	No	Yes	No
Data Rate	37.5 kbps (LoRa) 50 kbps (FSK)	100 bps (UL) 600 bps (DL)	$\leq$ 150 kbps	Several MBps
Current	$32mA$ $1\mu A$ (Sleep)	$27mA$ (Tx) $16\mu A$ (Sleep)	$120 - 300mA$ $5\mu A$ (Sleep)	

Table 1: Parameter differences between wireless network technologies

As seen, SigFox or LoRaWAN can be really useful for new projects that have less budget or need more range.

4.3 Sensors for crop protection systems

The sensors used for these kind of systems varies across the state of the art, but there are some specific type of technologies that are commonly used.

Fire detection sensors

Some of the most common sensors for prototyping are fire detection sensors, some types can be:

- **IR (InfraRed)**: react to the presence of flames analyzing parts of the spectrum.
- **UV (UltraViolet)**: these type of sensors respond to the instant a flame appears.
- **Other types**: such as photoelectrical sensors or ionization sensors.

The most used fire sensors are the IR sensors, mostly for their low price and fast reaction time. Usually the information of these sensors alone isn't able to detect a fire hazard without false positives, so the information is enhanced with information like temperature, humidity or even soil moisture. Finally, one important problem of this sensors is that the detection range is very low(can be around 2 meters) and can not be applied to open environments.

Gas presence sensors are also used for detecting fires. They contain a detection capsule composed of an electro-chemical that varies it's resistance if there are specific substances present.

These can prove to be useful to detect different types of fires, as they can detect the presence of:

- Smoke.
- Alcohol.
- Carbon monoxide.
- Hydrogen.
- Sulfuric Acid.
- Etc.

In the use case of this study, the most useful is the smoke detector.

Animal intrusion detection

There are different approaches for the detection. One key aspect is the animal is controlled, if this is the case, the system can be implemented with an approach based on RFID (Radio Frequency Identification). This line of research employs RFID technology to identify individual animals using implanted tags. When an animal with a tag approaches a protected area, the system detects it and generates deterrent measures, such as sounds or artificial fog.

This approach, while effective, requires knowing the animal and equipping them with sensors, which can limit their applicability to certain species.

But if the animal is wild, there is no possibility for sensors to be equipped to the animal. It is neither cost-effective nor scalable. To solve this, there are different technologies than

are applied: light-beam interruption sensors, radar, thermal radiation sensors and optical or thermal sensors. As it is required to cover big areas with these sensors, the power requirements of the platforms need to be considered, balancing this with the range covered. In this regard, the most used for detecting the presence of animals around the crop field are PIR (Passive InfraRed) sensors. These sensors are passive (the needed current order is μA, compared to the others that need to be constantly active) and use pyroelectric electronics to detect changes in radiation that could indicate the presence of a living animal.

These sensors are also good in terms of power consumption, some modern ones achieving less than $30\mu A$[9], but as the range depends on the application, decisions about the model are needed. The range and the angle that this sensors are able to use depend mainly on the lens, which are usually Fresnel lens.

Fresnel lens allow for light aggregation to one focal point without the need for concave of convex lens, thus, being easy to manufacture and to carry. For PIR sensors two types of lens are used, multisegmented or single-segmented Fresnel lens. The multisegmented usually can achieve a minimum detection range of $20m$, while the seconds can get up to $45m$[6].

As an example of their usage, there are systems that have PIR sensors to detect movements in the environment, as well as load cells to differentiate small animals from humans[10]. These nodes can communicate with each other and with central devices using Wi-Fi. In case of detection, these systems can activate acoustic alarms to scare away animals, while sending alerts to farmers.

Another example is a project to detect the intrusion of wild animal in roads[6]. This is done by putting nodes alongside the road, every $50m$. This is achieved studying the lens needed in that use case.

4.4 Active responses

Modern IoT systems not only detect animals or fires, but also implement active responses. In the early works for Smart Farming, the usual actuator would be automatic sound generation systems (to scare the wild animal), moving elements or water pumps (to stop the fire hazard). Nowadays the use of new technologies like mobile phones has expanded. For example, some projects include ultrasonic sensors and thermal cameras to identify animals of various species and emit specific sounds to repel them. In addition, the use of mobile applications is used to notify users about intrusions in real time, allowing the user to take the final decision about what the system should do.

The projects seen doesn't include functionalities like:

- Functionalities to control or commanding the network.
- Functionalities that allow for scalability in the system.
- Functionalities based around roles, to allow for more that one user with different attributes.

But this are not easy to implement, as they need bigger architectures and specific middleware developed. This also means that more computing power is needed in the system.

4.5 Edge computing

The new 5G (Fifth Generation of mobile communications) paradigm focuses on extending the capabilities of cloud computing to the border of the networks of the mobile operators. This allows for the creation of computer resources physically close to the end-user, this is called MEC. The general architecture of these solutions can be seen in Figure 1.

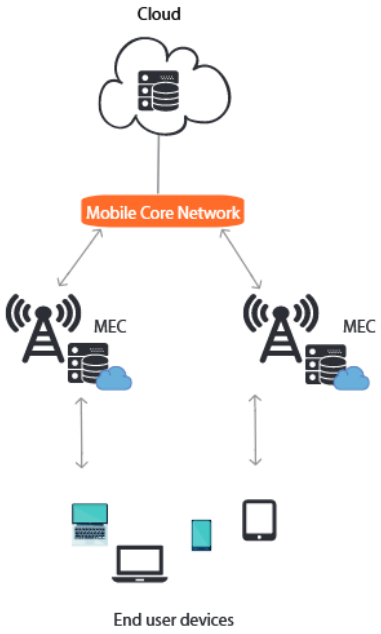


Figure 1: General architecture of edge computing[11]

There are already some companies offering SaaS (Software as a Service) solutions for edge computing, but there are open solutions that can be integrated in a edge node to create a middleware for the system. One of this technologies is the `Thingsboard` platform. This platform[12] allows for:

- Device management.
- Data aggregation.
- Data collection.
- Data processing.
- Visualization solutions or piping to other services.
- Intercommunication interfaces.
- Scalability.
- Integration with other useful technologies like `Kafka` for logs or `Node Red` for simulation of events.

5 Requirement Analysis

In this chapter, the requirement analysis is done to further divide the task for achieving the project. First, the general requirements for the project are presented and explained, followed by the identification of stakeholders to help understand the possible users and external factors that can affect the development of the solution. With this, the formal requirements are analyzed, and finally, quality attributed are specified for the validation of the system.

5.1 General Requirements of the Project

The minimum requirements that need to be achieved by the end of the project are:

- The deployment must have real and simulated devices working together.
- Sensors are actuators must be used. In this sense, some must be real.
- The intelligence must be distributed in some nodes of the system.
- There must be a possibility to command the IoT network.
- Knowledge extraction from the IoT network must be possible.
- Fusion of information is needed, both in the edge and from decision support systems.

5.2 Stakeholder Analysis

This project can affect several type of future users. This needs to be considered to stablish the scenarios the system will face.

- **Farmers**: The end users who will utilize the system to protect their crops and properties. The jobs these farmers carry out can differ, for example, one farmer can be the owner of the system and others can just be workers that also need to access the system.
- **Maintenance Personnel**: If any problem appears, these users are responsible for maintaining the system and ensuring it's proper functioning.
- **Equipment and Sensor Suppliers**: Companies that supply necessary components, such as MCU (MicroController Unit), smoke, flame or motion sensors.
- **Internet Service Providers**: These companies supply the resources for edge computing and internet connectivity.
- **Governmental and Regulatory Organizations**: Entities that ensure the system complies with agricultural and safety regulations.
- **Agricultural Consultants**: Professionals who advise farmers on integrating and using the system in their agricultural practices.
- **Energy Suppliers**: Companies that provide energy solutions, such as solar panels, to power the system.
- **Local Community**: Neighbors and other local stakeholders who could indirectly benefit from the system's implementation.
- **Investors and Financiers**: Entities or individuals who provide the necessary funds for the system's development and implementation.

5.3 Functional Requirements

ID	Functional Requirement	Description
FUN-FIRE-001	Smoke Detection	The system must detect smoke
FUN-FIRE-002	Temperature Change Detection	The system must detect sudden temperature changes
FUN-FIRE-003	Sensor Alert Transmission	Smoke and temperature sensors must send alerts to the central system
FUN-FIRE-004	Sensor Alert Transmission	Smoke and temperature sensors must send alerts to the central system
FUN-ANIMAL-001	Perimeter Animal Detection	The system must detect animals around crops using motion sensors
FUN-ANIMAL-002	Animal Detection Alert	Sensors must send alerts to the central system when animal movement is detected
FUN-ANIMAL-003	Automatic Animal Repellent Activation	The system must activate repellents automatically when animals are detected
FUN-ANIMAL-004	Manual Repellent Activation	The system must allow manual activation of repellents from a remote interface
FUN-ALERT-001	Real-time Notifications	The system must send real-time alerts to farmers through mobile or web apps
FUN-ALERT-002	Alert Details	Notifications must include details about the type of alert and exact location

Table 2: Functional Requirements 1

ID	Functional Requirement	Description
FUN-NET-001	Internet Connectivity	The system must connect to the internet for remote monitoring and control
FUN-NET-002	Communication Technology	The system must use Wi-Fi or cellular communication to ensure connectivity in remote areas
FUN-NET-003	Lora Connectivity	The node system must use Lora connection
FUN-UI-001	User Interface Design	The node system must provide a user-friendly interface accessible from mobile and desktop devices
FUN-UI-002	Real-time Data Visualization	The interface must display real-time data and allow manual device control
FUN-ENERGY-001	Autonomous Operation	The system must operate autonomously for prolonged periods
FUN-ENERGY-002	Renewable Energy Source	The system must utilize renewable energy, such as solar panels, to ensure continuous operation
FUN-ENERGY-003	Battery Duration	Send battery energy
FUN-ENERGY-004	Battery Alert	Send low battery alert
FUN-DATA-001	Sensor Data Storage	The system must store sensor data for analysis and reporting
FUN-DATA-002	Historical Data Visualization	Users must be able to view historical data in graphs and tables
FUN-USER-001	User Management	The system must manage multiple users
FUN-USER-002	Role-based access	The system must work with permissions

Table 3: Functional Requirements 2

5.4 Non-Functional Requirements

ID	Domain	Description
NFR-PF-001	Performance	The system must process and respond to sensor signals in real-time, with a maximum delay of 1 second for critical actions
NFR-SC-001	Scalability	The system must be scalable to support an increased number of sensors and actuators in the future without affecting performance
NFR-RE-001	Reliability	The system must have high availability, with downtime not exceeding 1% annually
NFR-RE-002	Reliability	Redundancy mechanisms must be included to ensure that alerts and critical actions do not fail
NFR-MAN-001	Maintainability	The system must be easy to maintain and update both in hardware and software
NFR-MAN-002	Maintainability	Manuals and detailed documentation must be provided to facilitate maintenance tasks
NFR-US-001	Usability	The user interface must be intuitive and easy to use with a minimal learning curve
NFR-US-002	Usability	It must be accessible from mobile devices, computers and be available in multiple languages if necessary
NFR-SEC-001	Security	The system must include secutrity measures to protect user data and privacy
NFR-SEC-001	Security	The system must include secutrity measures to protect user data and privacy

Table 4: Non-Functional Requirements 1

ID	Domain	Description
NFR-SEC-002	Security	Data encryption must be used for sensitive data transmission and authentication for user interface access
NFR-COM-001	Compatibility	The system must be compatible with various types of sensors and actuators available in the market
NFR-COM-002	Compatibility	It must support different communication protocols with the nodes
NFR-EE-001	Energy Efficiency	The system must minimize energy consumption, especially if renewable energy solutions such as solar panels are used
NFR-EE-002	Energy Efficiency	Components must enter low-power mode when not actively in use
NFR-POR-002	Portability	The system software must be portable and able to run on different platforms and devices without significant modifications
NFR-LC-001	Legality and Compliance	The system must comply international regulations and standards regarding agriculture, security, and privacy

Table 5: Non-Functional Requirements 2

5.5 Operation Domain Requirements

- Interaction with the farm workers while maintaining compatibility with agricultural practices.
- Helping in the promotion of sustainable practices and efficient energy use.
- Compliance with agricultural and privacy regulations.
- Optimization of the use of agricultural resources.

5.6 Quality attributes analysis and Impact analysis

- **Performance**:
  - **Response time**: The system must process and respond to alerts in less than 1 second.
  - **Processing capacity**: It must handle multiple simultaneous events without degrading performance.
- **Reliability**:
  - **Availability**: The system must be operational at least 99% of the time.
  - **Redundancy**: It must have backup mechanisms to ensure service continuity in case of failures.
- **Security**:
  - **Data protection**: All transmitted data must be encrypted.
  - **Controlled access**: Only authorized users should be able to access and control the system.
- **Scalability**:
  - **Sensor expansion**: It must allow the addition of new sensors and actuators without significant changes.
  - **User increase**: The system should support a growing number of users and connected devices.
- **Maintainability**:
  - **Ease of updating**: The system must allow software updates without interrupting its operation.
  - **Documentation**: There must be clear and detailed documentation to facilitate maintenance.
- **Usability**:
  - **User-friendly interface**: The interface must be intuitive and easy to use for people without technical knowledge.
  - **Accessibility**: It must be accessible from different devices (mobile and computers).

Based on the quality attributes presented above, each attribute has been classified according to its relevance and complexity within the system, as presented in Table 6.

Quality Attribute	**Relevance**	**Complexity**
Response Time	High	High
High Processing	High	Medium
Availability	High	High
Redundancy	High	Medium
Data Protection	Low	Low
Controlled Access	High	Low
Sensor Expansion	High	Medium
User Increase	Medium	High
Ease of Updating	High	High
Documentation	Medium	Medium
Friendly UI	High	Medium
Accessibility	High	Medium

Table 6: Prioritization of Quality Attributes Based on Relevance and Complexity

6 Architectural Views

This chapter aims to provide information on the general functionalities of the proposed idea as well as insights on the implementation methodologies and architectural designs that will be chosen. To achieve this representation of the idea, the main resource used will be several UML diagrams.

Firstly, the functionalities of the system will be defined through the scenario views, thereby the logical view which will serve as the implementation base of the system will be designed. With the two views, several other views such as the process views for some use cases of the system and deployment designs will be presented.

All this designing will be based on the Kruchten's 4+1 views model[13].

6.1 Scenarios view: Use case diagrams

Use case diagrams are used to represent all the possible scenarios that implement the functionalities specified.

For each subsystem use case analysis, a detailed description can be found with:

- Actors affected by the use-case system.
- The preconditions identified to achieve the functionalities described.
- The post-conditions.
- The basic flow of actions.

Animal detection system use cases

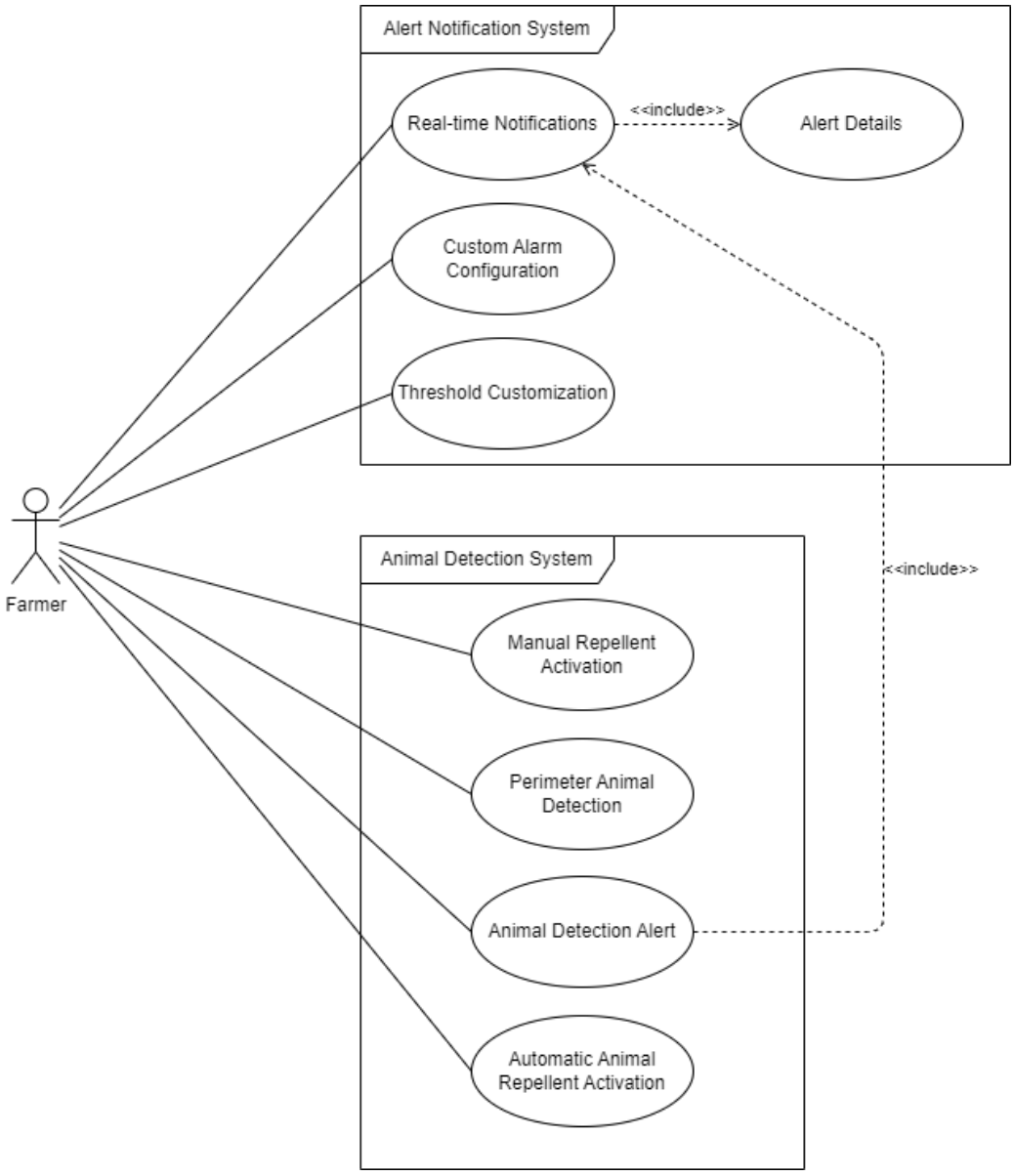


Figure 2: Animal detection system use cases diagram

- **Name**: Animal detection.
- **Actors**: Farmer.
- **Description**: Farmer can receive alerts of animal detection on the farm perimeter and manually or automatically configure the repulsion system.
- **Pre-conditions**:
  1. The animal detection system must be operational.
  2. The system must have the alert sending function enabled.
  3. The user must be logged into the system.
- **Basic flow of actions**:

1. The user login into the system.
2. The perimeter of the farm can be viewed on the map.
3. If an animal is detected in the perimeter, an alert is sent.
4. The user views the animal detection alert.
5. In the map the line the perimeter changes the color in reaction to the presence.

User management system use cases

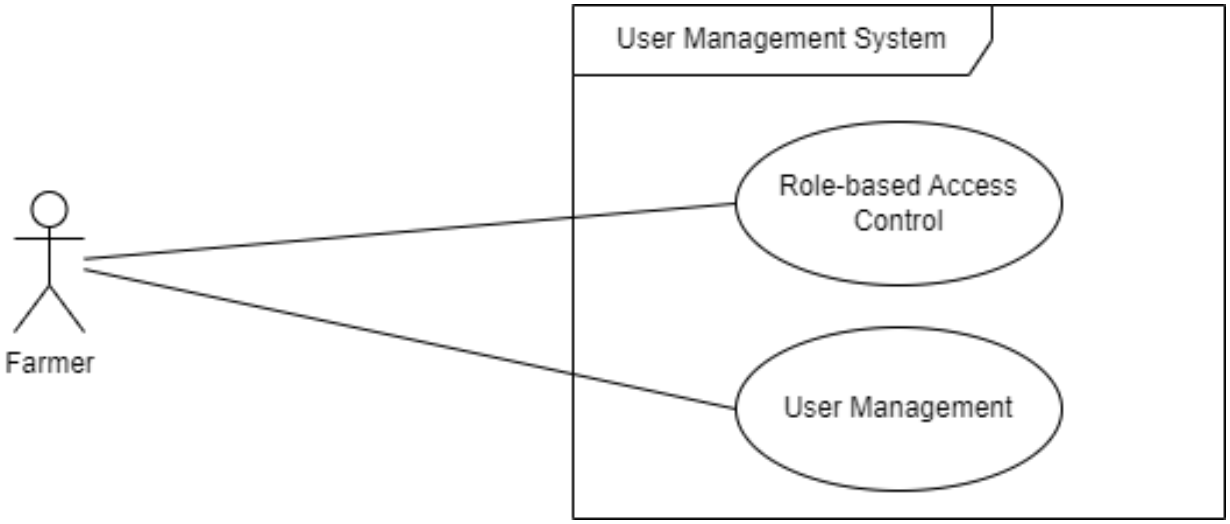


Figure 3: User management system use cases diagram

- **Name**: User Management.
- **Actors**: Farmer.
- **Description**: The farmer can configure user management and assign role access to other users.
- **Pre-conditions**:
  1. A farmer is the administrator.
  2. The roles are defined.
  3. There must be more than one user.
  4. The user must be logged into the system.
- **Basic flow of actions**:
  1. The user login into the system.
  2. The user can assign a role to another user.

Fire detection system use cases

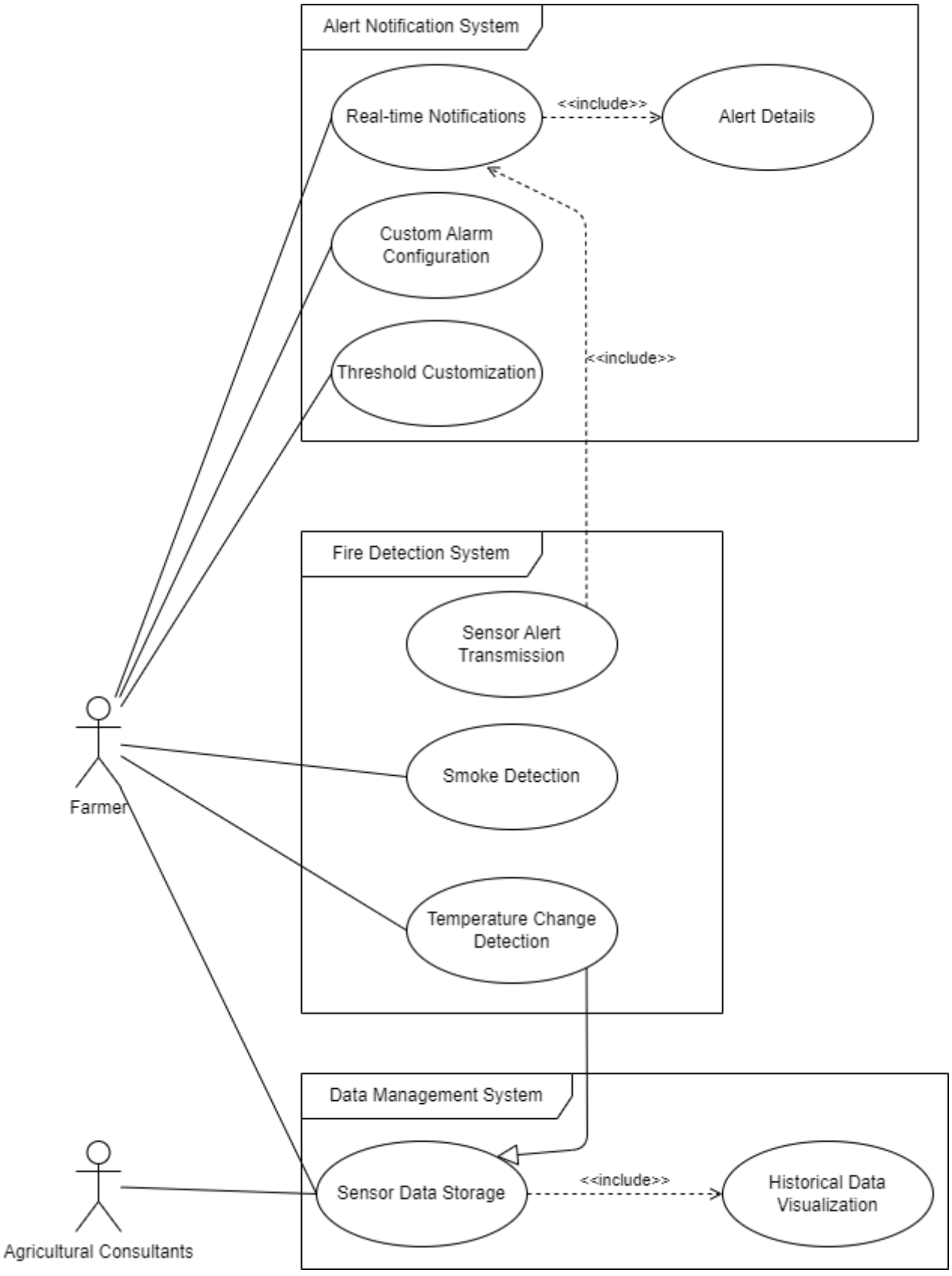


Figure 4: Fire detection system use cases diagram

- **Name**: Fire Detection.
- **Actors**: Farmer.
- **Description**: The farmer can receive alerts of fire detection on the farm perimeter and can monitor the different sensors.
- **Pre-conditions**:
  1. The fire detection system must be operational.
  2. The system must have the alert sending function enabled.
  3. The user must be logged into the system.
- **Basic flow of actions**:
  1. The user login into the system.
  2. The perimeter of the farm can be viewed on the map.
  3. If a fire is detected in the perimeter, an alert is sent.
  4. The user views the fire detection alert.
  5. The fire position will be detected in the map.

Energy system use cases

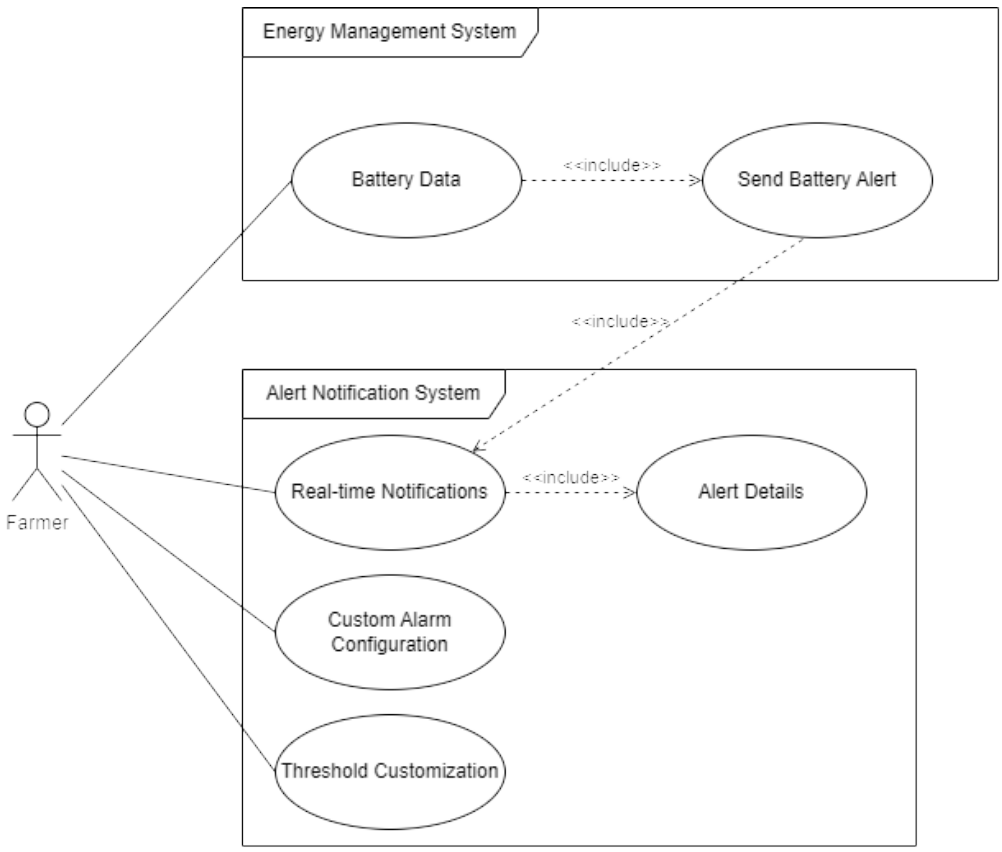


Figure 5: Energy system use cases diagram

- **Name**: Battery Management.
- **Actors**: Farmer.
- **Description**: The farmer can check the battery live and receive alerts of the battery status.
- **Pre-conditions**:
  1. The system must have the alert sending function enabled.
  2. The user must be logged into the system.
- **Basic flow of actions**:
  1. The user login into the system.
  2. The user can check the battery status of the node.
  3. If the battery is low, the system sends an alert.

Weather conditions monitoring use cases

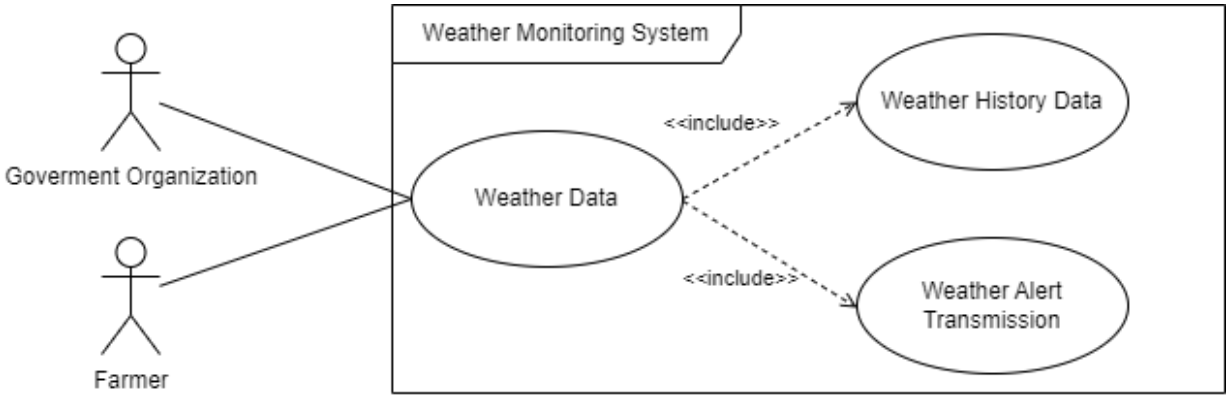


Figure 6: Weather conditions use cases diagram

- **Name**: Weather Monitoring.
- **Actors**: Farmer.
- **Description**: Farmer can monitor the weather data such as wind speed.
- **Basic flow of actions**:
  1. The user login into the system.
  2. The user checks the weather data.

6.2 Logical view: Class diagram

In this section the class diagram for the PIR sensor node is presented.

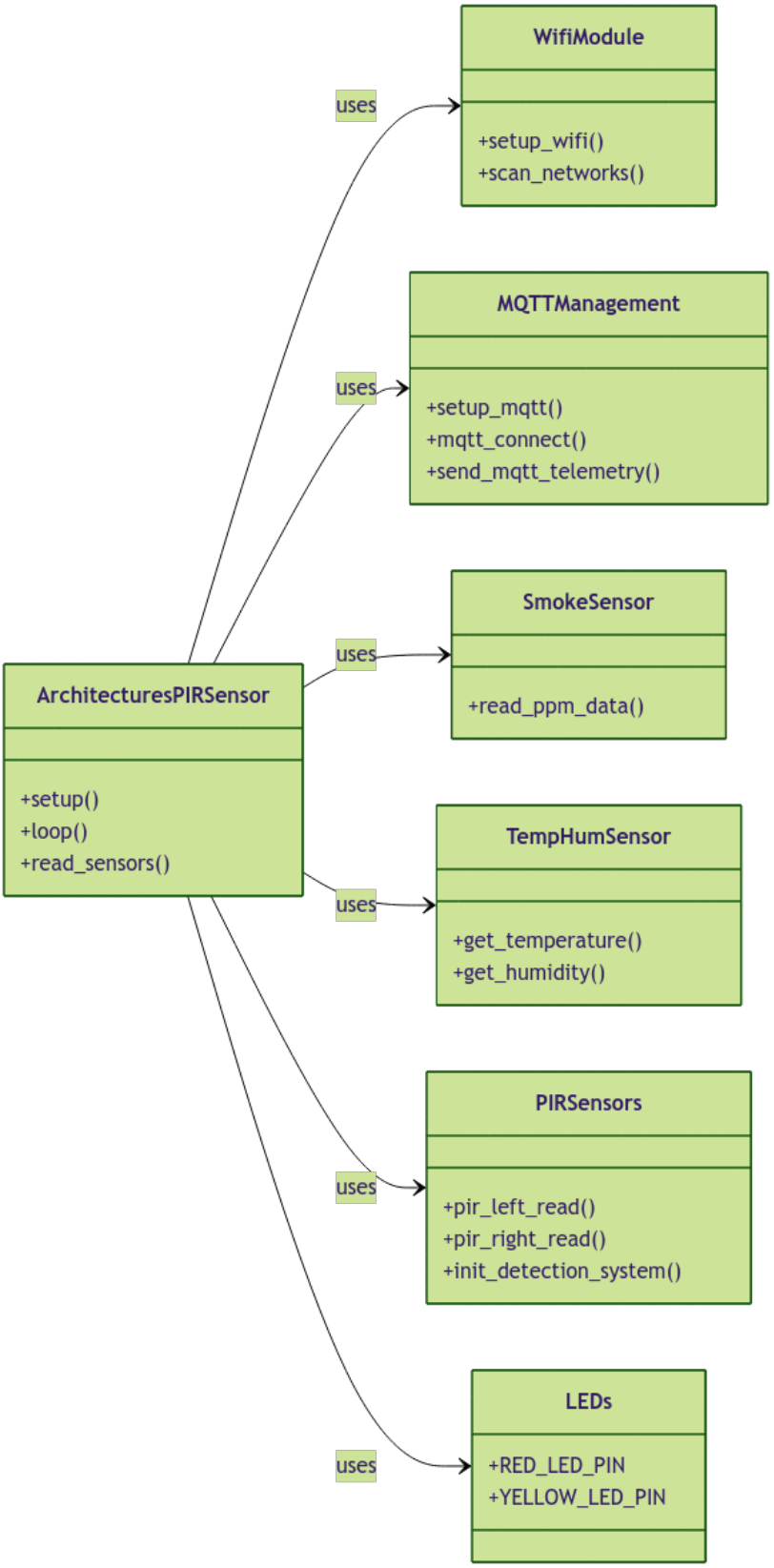


Figure 7: Presence detected sequence diagram

6.3 Process view: Sequence diagrams

Sequence diagrams are used in this project to represent the flow of the system in a selected use case. This view is used to achieve implementation by representing the expected steps to integrate a functionality.

Presence detected sequence

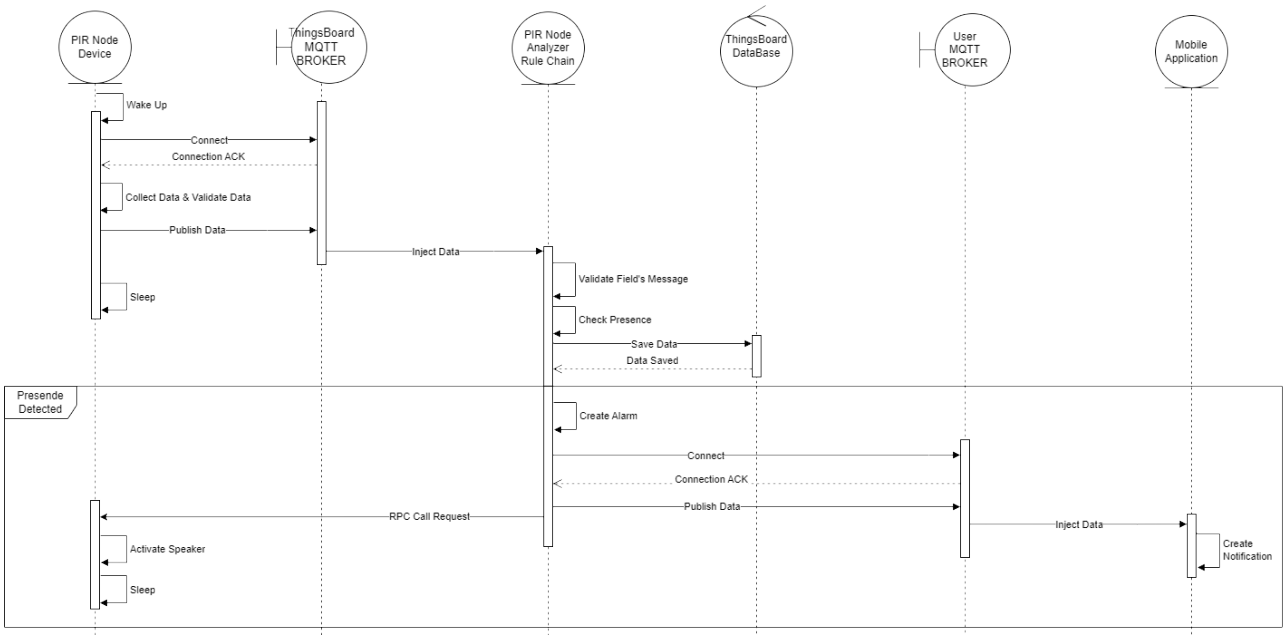


Figure 8: Presence detected sequence diagram

Fire hazard detected diagram

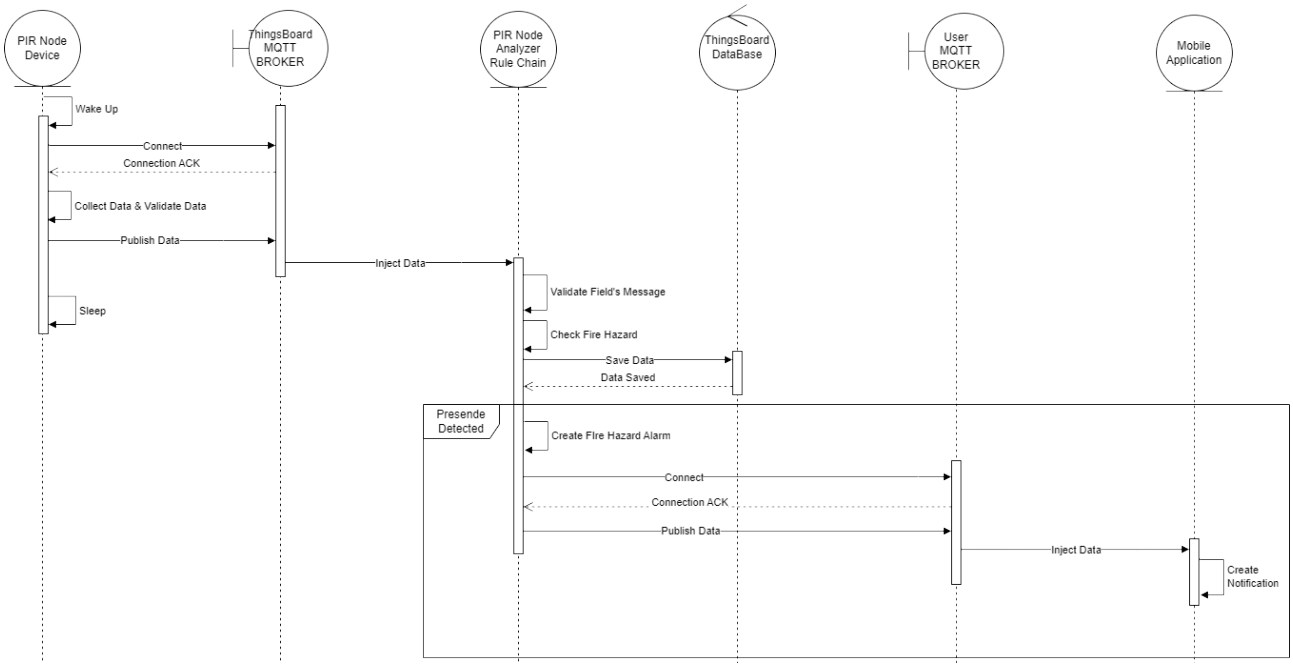


Figure 9: Presence detected sequence diagram

Send command from dashboard

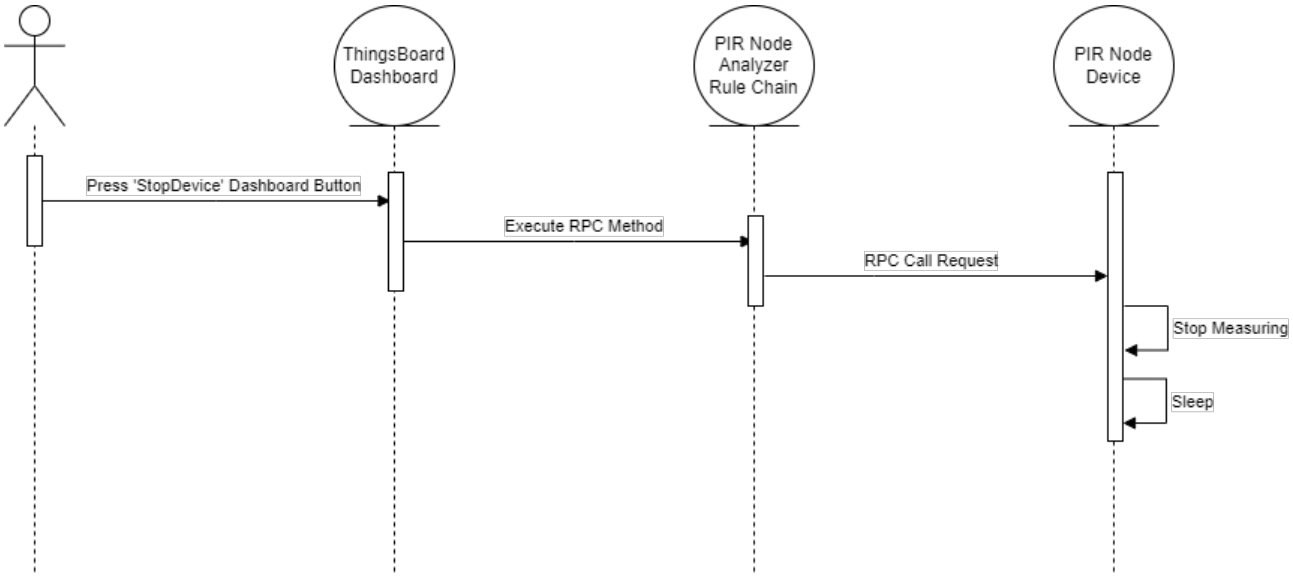


Figure 10: Send command from dashboard sequence diagram

Send command from mobile app

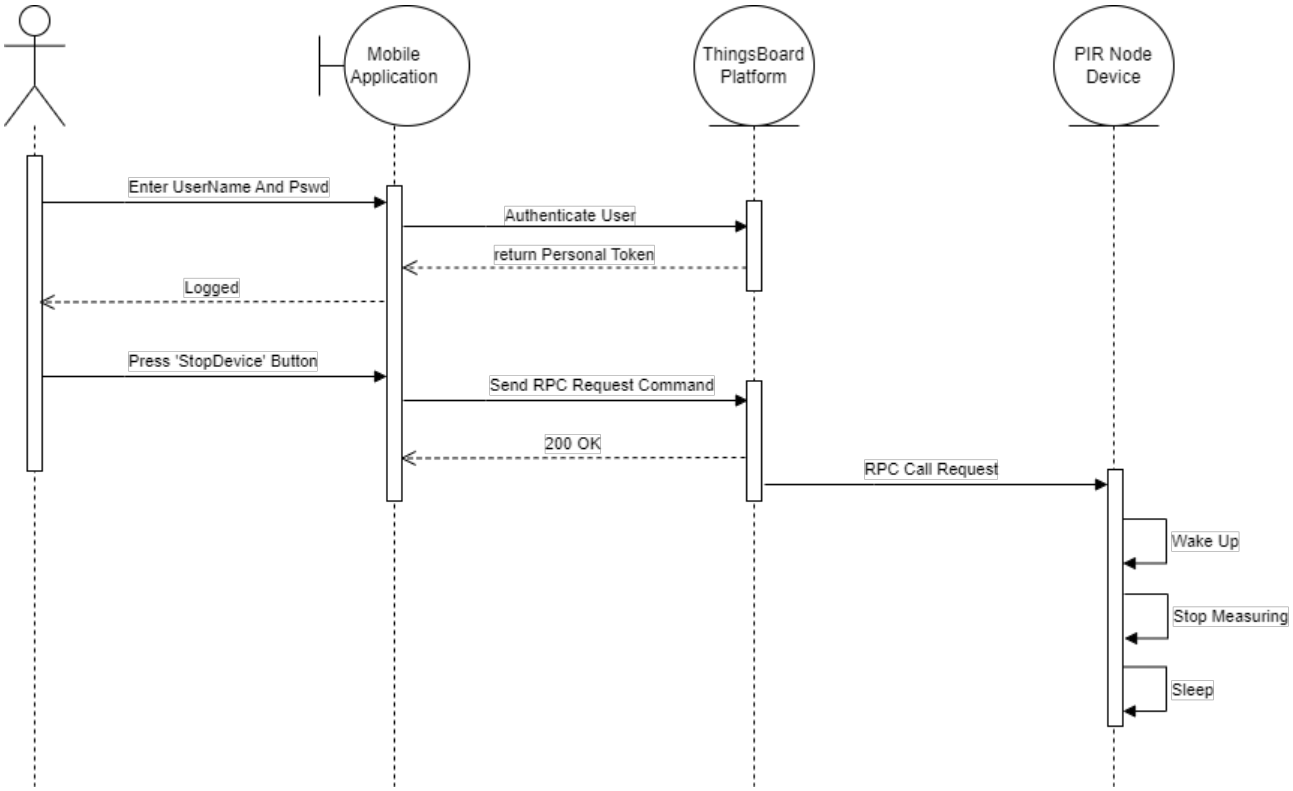


Figure 11: Send command from mobile app sequence diagram

6.4 Implementation view: Component diagram

The Figure 12 shows the different components of the system and interfaces needed for it to work.

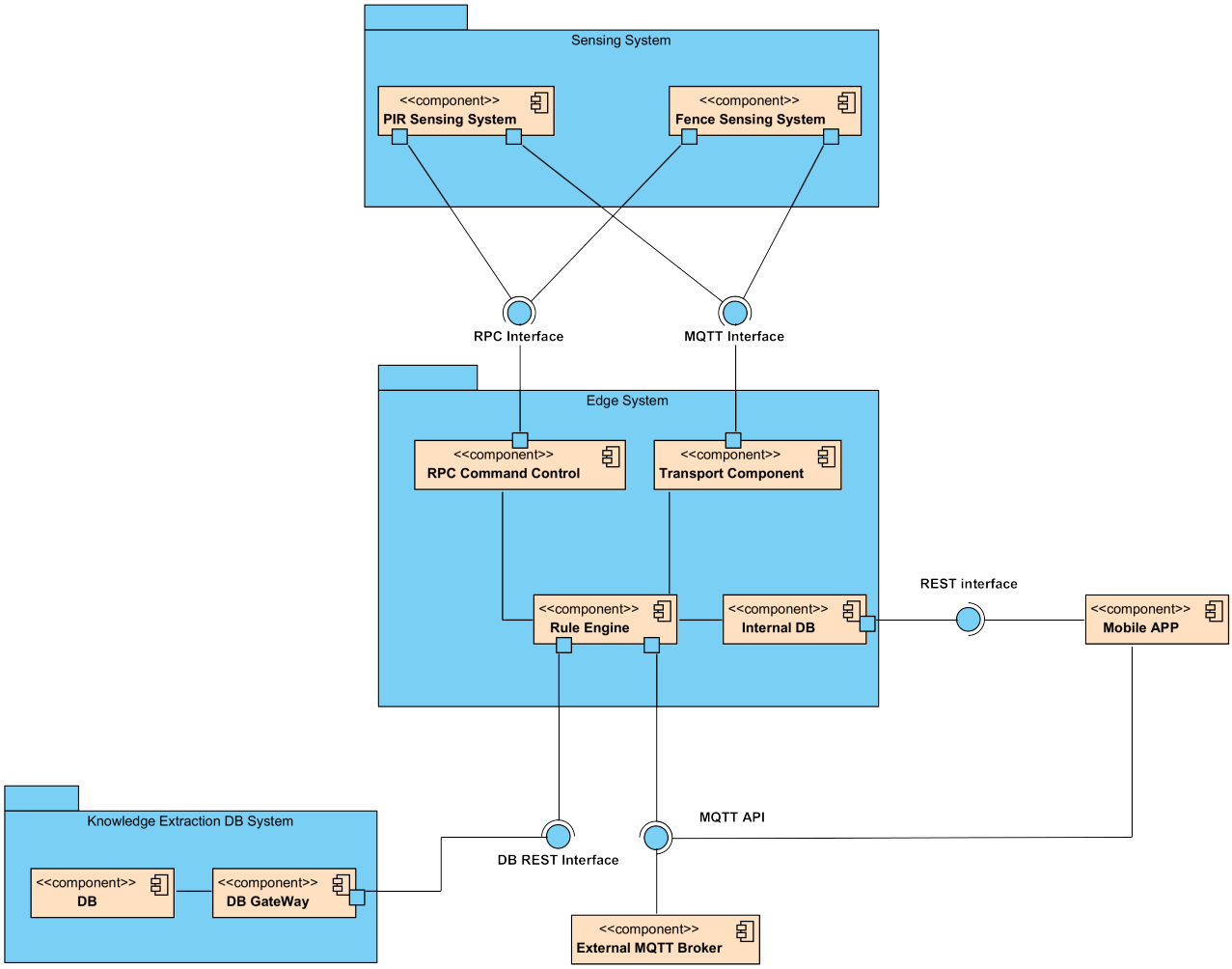


Figure 12: Component diagram for the system

6.5 Deployment diagram

The diagram in Figure 13 shows the node distribution in the system. It also shows the different connection protocols used between them.

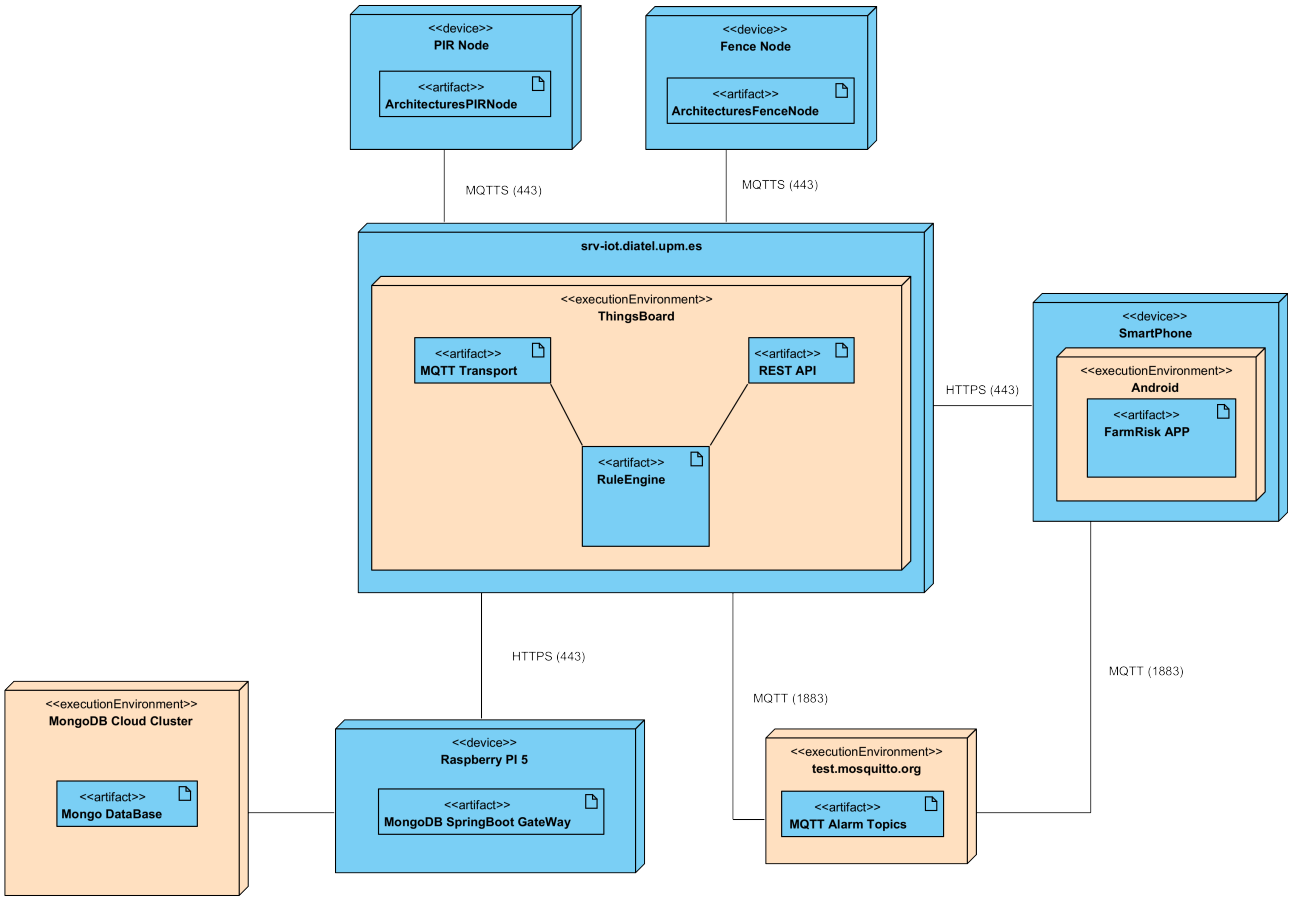


Figure 13: Deployment diagram for the system

7 System Architecture

This system mainly focus in the interconnection and interoperability of different elements. To achieve this, an architecture was designed in response to the requirements.

A general overview of the proposed architecture can be seen in Figure 14.

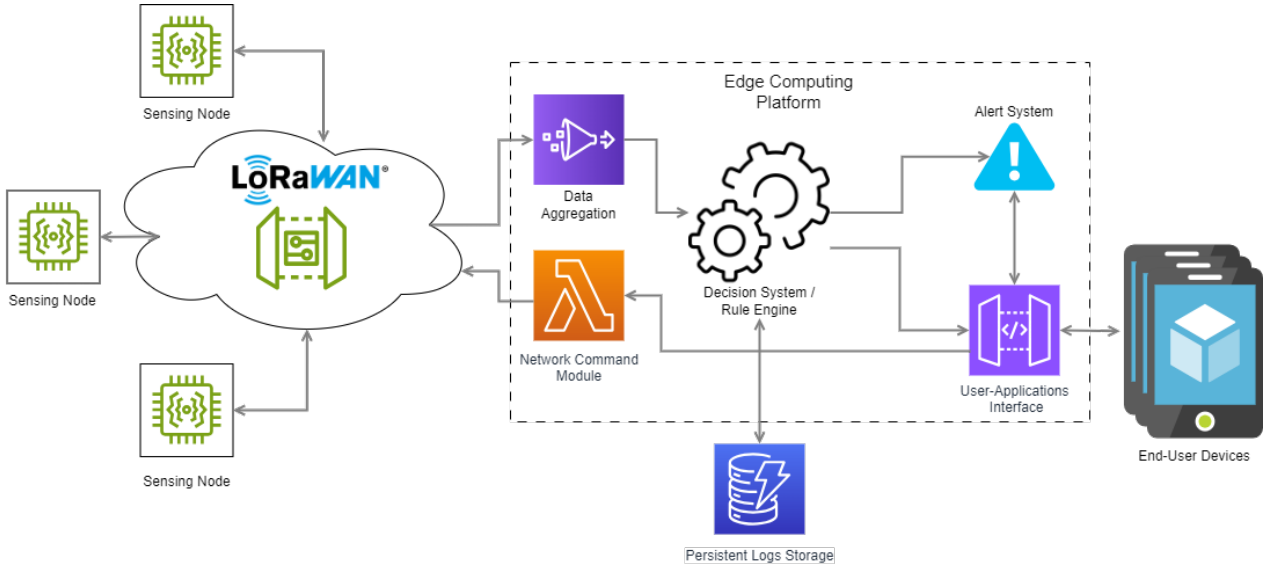


Figure 14: General architecture design for the proposed system.

Thanks to the research done, the selection for the intercommunication of the nodes with the edge platform will be done using LoRaWAN. This protocol ensures good coverage in rural areas, moreover, it ensures scalability in the system designed. The connection is done with lora from the nodes to the gateway. This gateway connects to the edge computing platform with IP technologies to send all the data from the nodes. Finally, this data will be collected in the data aggregation module, which will allow information coming from other wireless technologies or other LLN (Low power lossy Networks).

When all the data is aggregated, it will pass through the rule engine module. This step analyzes the data in search for patterns that indicate the existence of a fire or will animal near the system. All these processing is logged into a persistent data storage that can serve as a knowledge base for future projects.

If a fire or wild animal is detected, a set of actions are done, this is done through the alarm system of the edge platform and the User-Application interface. These modules allows for interaction with the users, allowing different functionalities like:

- Access from mobile applications to the processed data.
- Real-Time alarm and registration of events.
- Allowing users to react, sending commands to the network to solve problems.

8 Implementation

For the project, all the focus was made to achieve the next core requirements:

- Achieve both a simulated and real deployment of the required nodes.
- Real and simulated sensors are used, finally actuators such a virtual speaker are simulated.
- Intelligence distributed both in the nodes and in the edge platform.
- Possibility to command the network to stop the sensing.
- Fusion of information in the Edge Platform. This must result on some decisions and visual representations of the system's global state.
- A solution must be implemented to support future knowledge extraction in the system.

An important factor for the implementation of this project was the limited access to a LoRaWAN gateway. To solve the communication part, MQTT was used, as the edge platform supports this communication protocol by default.

8.1 Sensing nodes

In this project, two distinct nodes were developed, each with its specific characteristics. On one hand, we have a node equipped with sensors for presence detection, periodic measurements of temperature, humidity, wind, smoke, and flame detection via infrared sensors, located at the perimeter of the field. On the other hand, a simpler node equipped with an accelerometer is placed on the fence delimiting the crop field to detect potential unauthorized crossings.

Pir Node

These nodes are positioned along the perimeter of the field. These PIR Nodes are built using the `PI PICO-W` board[14]. This board is used for it's utility in fast deployment and possibility for Wi-Fi and BLE (Bluetooth Low Energy). This node has the capability of sending the information related to:

- 2 real PIR Sensors, one for each side. The sensor used for this is the `HC-SR501`[15]. This sensor was selected for it's adjustability and low power consumption.
- 2 Infrared Sensors to help detect fires near the sensor.
- An MQ-2 smoke sensor[16] to analyze the presence of alarming particles in the air.
- A temperature and humidity sensor from `Adafruit`[17].
- A wind speed sensor. In this project this sensor is simulated due to the time constraints, but the support for the data it's included.
- A GPS Module is also simulated.

Finally, to represent the actuation of the network over the system. Two LEDs are used:

- Red led to represent the status of the system (Sending data or not).

- Yellow led to represent the activation of the speaker of the node to scare wild animals.

The node is presented in the Figure 15.

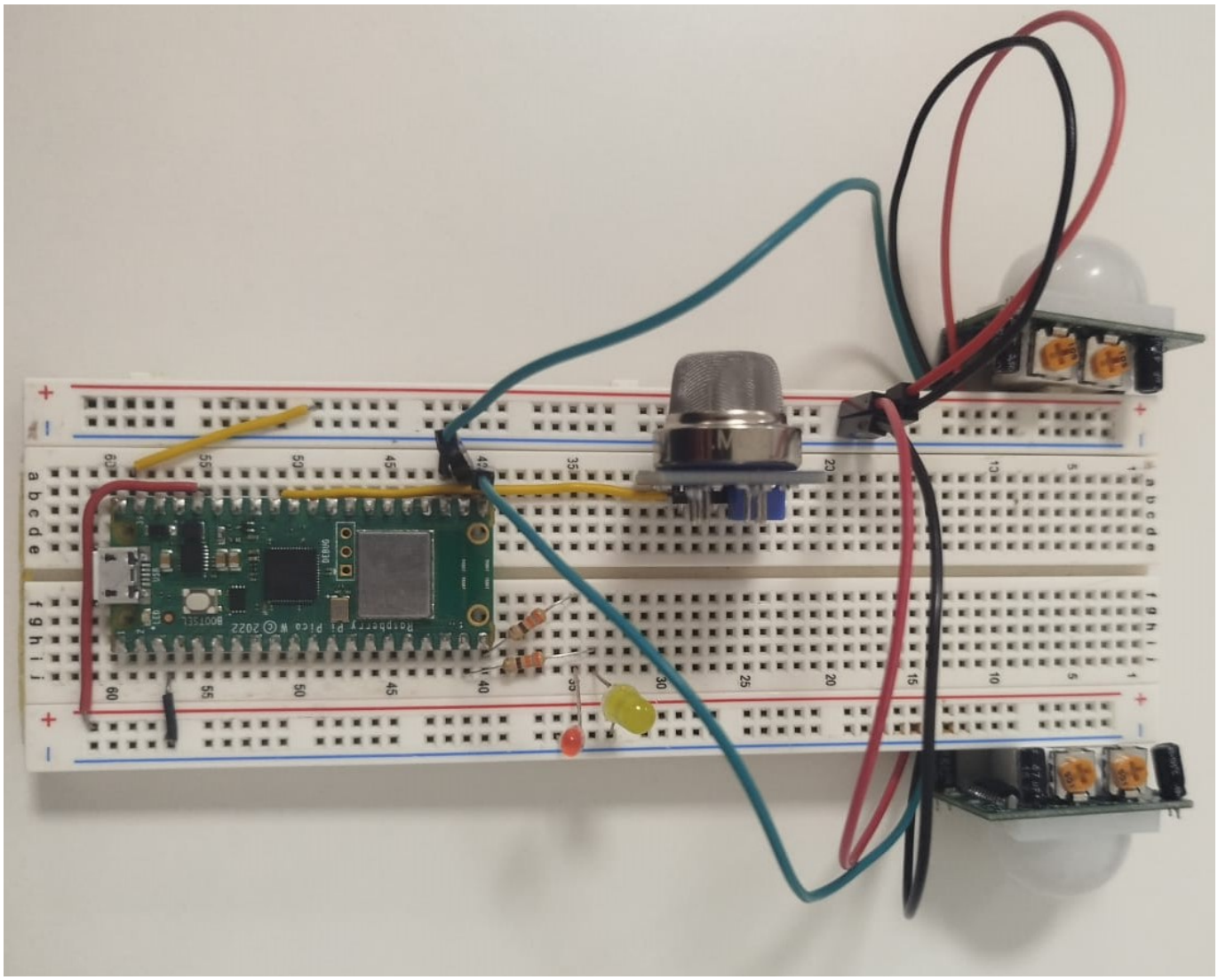


Figure 15: Final PIR Node

Fence Node

These nodes are positioned along the perimeter of the field, in the positions between the PIR nodes. The Fence Node contains an accelerometer that will sense any movements in the fence and send information with interruptions generated by the accelerometer. This node uses as the MCU an ESP32[18]. The accelerometer is an `MPU-6050`[19]. The node is presented in the next figure:

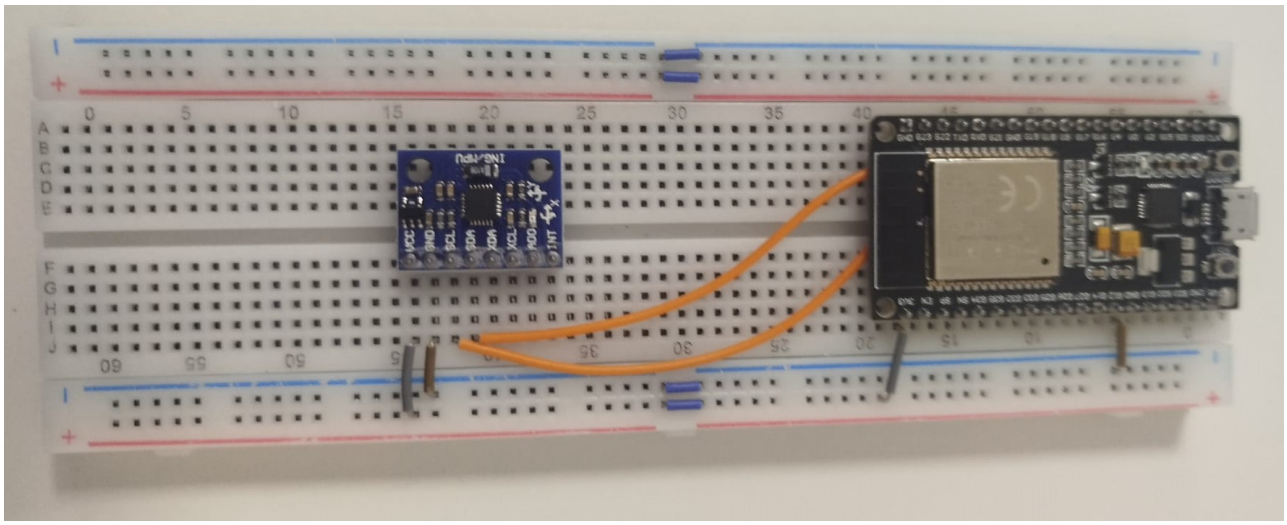


Figure 16: Final Fence Node

8.2 Edge Computing: ThingsBoard Platform

Devices and Assets Profiles

To manage the information from all the sensors, actuators, and elements involved in this system within the Thingsboard platform, we utilize virtual representations of these components through Devices and Assets. Devices represent each node virtually, while assets represent the corresponding physical elements. For our use cases, we distinguish between the two types of nodes, each with its own device profile, and define one asset type as listed below:

- PIR Node Profile.
- Fence Node Profile.
- Presence Asset Profile. This Asset Profile indicates real-time presence detection for each fence section. It differentiates between a presence detected on a single side, multiple detections, or a possible unauthorized crossing.

For this project, and considering the specified constraints, we deployed a total of 8 devices to cover the perimeter of the crop field, 4 devices for each defined Device Profile. Additionally, we defined 4 assets using the Presence Asset Profile type, representing the presence detected by each fence segment. Figure 17 provides a schematic representation of the nodes and assets placement.

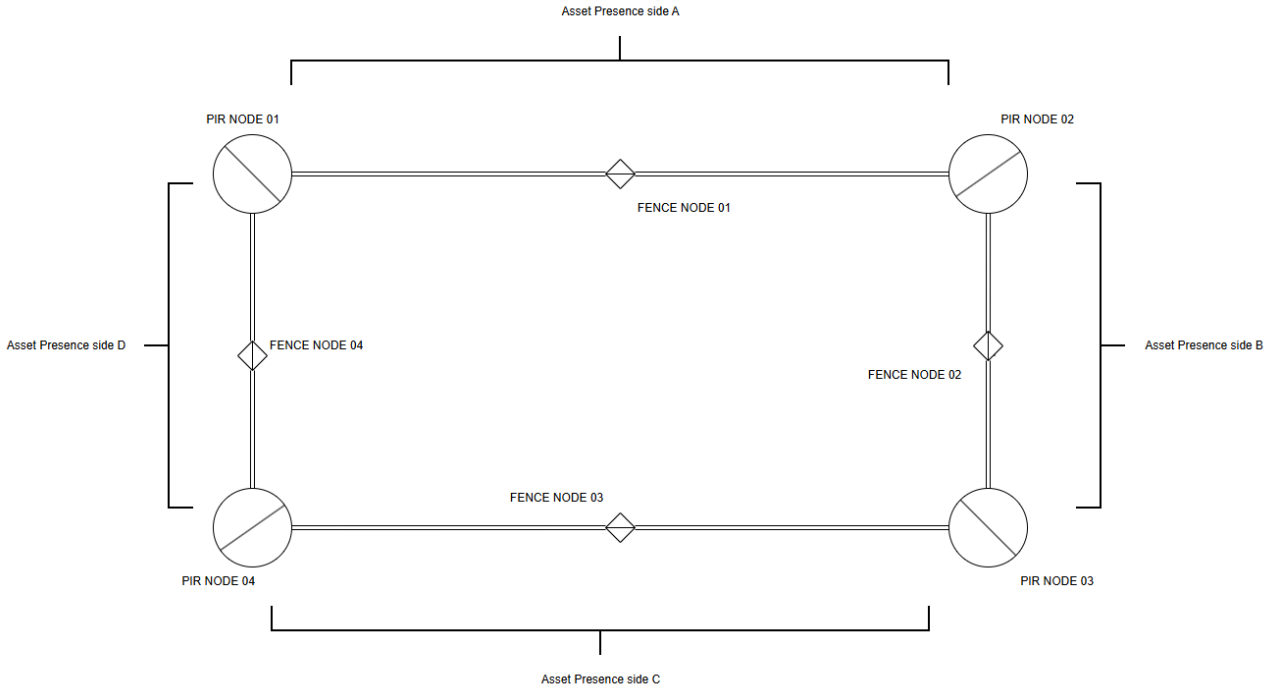


Figure 17: Node Schema

To detect presence or crossings effectively, the PIR Node Devices are configured with relationships to two neighboring PIR Nodes and two defined assets. These relationships are as follows (the terms left and right are oriented outward from the crop field):

- (`leftneighbour`): Links a PIR Node to the PIR Node located to its left.
- (`rightneighbour`): Links a PIR Node to the PIR Node located to its right.
- (`leftpresence`): Links a PIR Node to the Presence Asset on its left side.
- (`rightpresence`): Links a PIR Node to the Presence Asset on its right side.
- (`features`): Links a Fence Node to the Presence Asset associated with its section.

By leveraging these relationships, the system can monitor detected presences, determine which side of the field they occur on, and generate appropriate alarms accordingly. Therefore, the relationships for each node, based on the previous diagram, would be as indicated below:

Rule Chains

In this project, two distinct nodes were developed, each with its specific characteristics. On one hand, we have the PIR node and on the other hand, the fence node. Each device profile is assigned a specific Rule Chain to analyze the data received from the node devices and trigger the corresponding alarms.

PIR Node Analyzer

The alarms generated by this Rule Chain include:

- Low Battery.
- Fire Detected.
- Fire Hazard.
- Presence Detected.

The Rule Chain first validates that all required fields are present in the incoming message. If the message format is incorrect or incomplete, an alarm is triggered to indicate that the incoming message cannot be processed. Once the message format is validated, the data is parsed for easier subsequent processing, producing a JSON structure similar to the one shown in Figure 18.

Output


```json
{
    "msg": {
        "latitude": 1,
        "longitude": 2,
        "pir_left_detection": false,
        "pir_right_detection": false,
        "pir_data_left": 1,
        "pir_data_right": 2,
        "fire": true,
        "humidity": 0,
        "smoke": 0,
        "temperature": 0,
        "wind": 0,
        "battery": 0
    },
    "metadata": {
        "deviceType": "default",
        "deviceName": "Test Device",
        "ts": "1736896296752"
    },
    "msgType": "POST_TELEMETRY_REQUEST"
}
```


Figure 18: Parsed JSON for Rule Chain Processing

The low battery alarm is triggered when the node reports a battery level below 15%.

Fire detection relies on the value in the `fire` field. In the Check Fire node of the Rule Chain, if the `fire` field returns `true`, a critical alarm for "Fire Detected" is triggered. If the value is `false`, the alarm is deactivated.

The "Fire Hazard" alarm is activated by evaluating the parameters for temperature, humidity, wind speed, and smoke levels. According to some studies[20], the conditions for a fire hazard include:

- Temperature above 30°C.
- Relative humidity below 30%.
- Wind speeds exceeding $30 km/h$.

Additionally, a smoke concentration exceeding 200 PPM (Parts Per Million) triggers the alarm. If none of these conditions are met, the alarm is deactivated.

For presence detection, four PIR nodes are placed at each vertex of the square representing the crop field. The field is divided into four zones corresponding to each side of the polygon. The presence detection algorithm works as follows:

- If a presence is detected by a PIR sensor (indicated by 'pir_XX_detection', where XX can be 'left' or 'right'):
  - The status of the opposite side's neighboring node is checked.
  - If the neighboring node also reports presence (`true`) within a timespan of 10 seconds, the "Presence Detected" alarm is triggered, and a speaker is activated.
  - If no presence is detected on the opposite side, a lower-priority alarm is triggered, indicating presence on one side of the fence.
- If no presence is detected on the initial node, all related presence alarms are deactivated.

The pseudocode for this logic is as follows:

```
if (presence on one side){
    if ((presence on opposite-side neighbor node) && (ts < 10)) {
        Presence Detected;
        Activate Speaker;
    } else{
        Presence Detected on the corresponding side;
    }
} else{
    Deactivate all presence alarms;
}
```

Listing 1: Presence Algorithm

To implement this algorithm, relationships between the various nodes and assets were defined. Accessing values from neighboring nodes was achieved through the Change Originator node in Thingsboard, which specifies the target node using the predefined relationships. Similarly, the originator change process is used to update related asset values. The relations establish are detailed in the tables below.

PIR Node	**Left Neighbor**	**Right Neighbor**
PIR Node 01	PIR Node 04	PIR Node 02
PIR Node 02	PIR Node 01	PIR Node 03
PIR Node 03	PIR Node 02	PIR Node 04
PIR Node 04	PIR Node 04	PIR Node 01

Table 7: PIR Nodes relationships

Asset	Left PIR Node	Right PIR Node
Presence Side A	PIR Node 01	PIR Node 02
Presence Side B	PIR Node 02	PIR Node 03
Presence Side C	PIR Node 03	PIR Node 04
Presence Side D	PIR Node 04	PIR Node 01

Table 8: Assets relationships

After a PIR Node detects presence and the previously detailed logic is applied, if presence is detected on at least one of its sides, the system will not only generate the corresponding alarm, whether for presence detection on a single side of the fence or on both, but will also trigger the speaker activation command via the `rpc call request` rule chain node. This command set the speaker to emit sounds to deter the detected presence. The speaker will remain active until neither of the two PIR sensors in the node detect any presence. Once no presence is detected on either side, a new command will be sent to deactivate the speaker.

The full flow of the Rule Chain is illustrated in Figure 19.

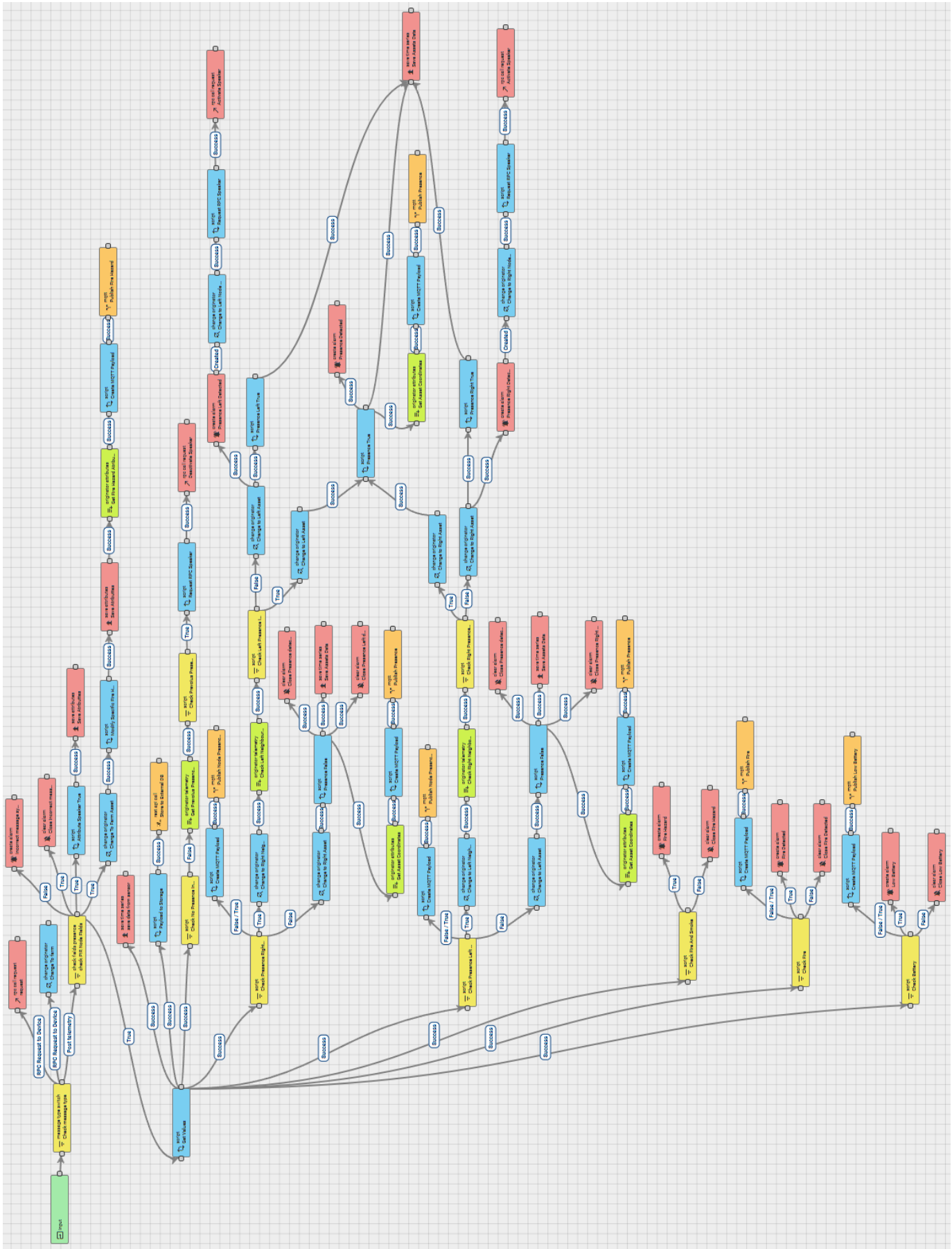


Figure 19: PIR Node Analyzer Rule Chain FLow

Fence Node Analyzer

This Rule Chain is designed to process messages from nodes placed on the fence to detect movements and generate alarms accordingly. Similar to the previous Rule Chain, it first validates that all required fields are present in the incoming message. However, since this node is simpler, the Rule Chain specifically generates the "Fence Trespassed" alarm based on the value of the 'acceleration_interruption' field. The flow of this process is illustrated in Figure 20.

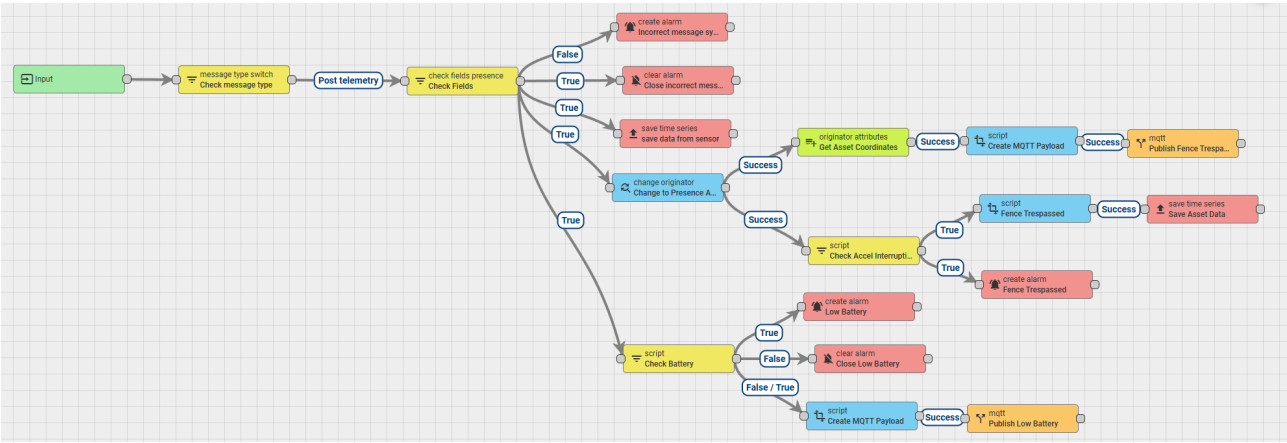


Figure 20: Fence Node Analyzer Rule Chain FLow

Alarms

To streamline the process of sending notifications about alarm activation or deactivation, both Rule Chain utilizes MQTT nodes to publish this information to an external broker accessible to the user. For this purpose, specific topics have been defined to publish the necessary information (detailed in the appendix)

METER JSONs de cada topic en el apendice

for triggering specific alarms in the mobile application, which is explained in more detail in its corresponding section. Te topics defined are:

Alarm	MQTT Topic
Low Battery	UPM/MIoT/ARS/RiskSystem/alarms/LowBattery
Fire Hazard	UPM/MIoT/ARS/RiskSystem/alarms/FireHazard
Fire Detected	UPM/MIoT/ARS/RiskSystem/alarms/FireDetected
Presence Detected	UPM/MIoT/ARS/RiskSystem/alarms/PresenceDetected
Node Presence Detected	UPM/MIoT/ARS/RiskSystem/alarms/NodePresenceDetected
Fence Trespassed	UPM/MIoT/ARS/RiskSystem/alarms/FenceTrespassed

Table 9: MQTT Topics defined for mobile application notification

Dashboard

To represent all the information generated during the processing of the aggregated data. The dashboard functionality from ThingsBoard was used. To implement the dashboard utilities like aliases and dashboard states were used.

The dashboard has 2 views, which were implemented using ThingsBoard states:

- The map dashboard, that shows the status of the farm perimeter and the different PIR nodes. This map can also change colors in reaction to some events, like presence detection or fire detection. This also affects the markers of the nodes. Finally, the nodes can be clicked to obtain a summarized view of the state of the node and the latest telemetry. This view is presented in Figure 21.

  This view also contains extra elements:

  - A Alarm table in which the alarms related to the whole system are presented.
  - A set of buttons to command the start/stop of the sending mechanisms for the PIR nodes. This can be used in case a worker needs to clean the perimeter.

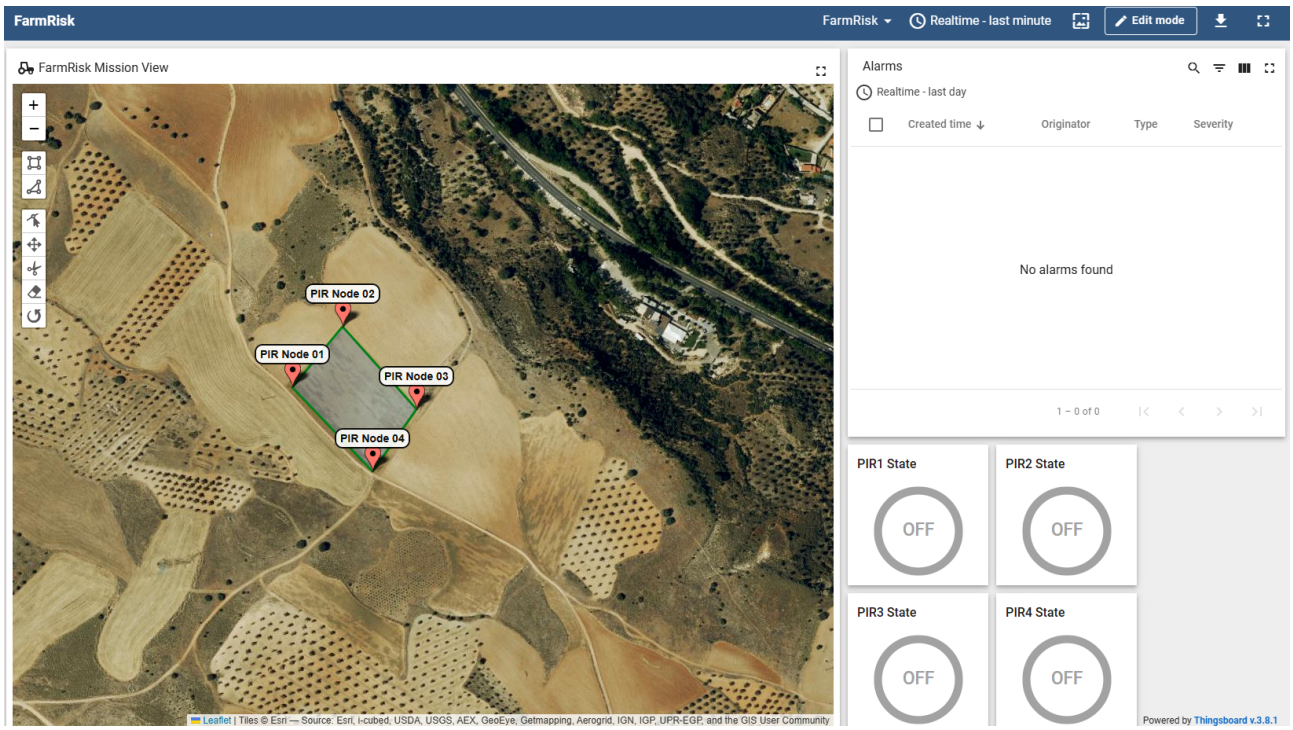


Figure 21: Map dashboard implemented in Thingsboard

- When any tooltip's "further details" is clicked, the dashboard changes to the sensor view of the sensor clicked. This dashboard shows time series data and further details about the state of the sensor. This view is presented in Figure 22.

  In this case, this view will only show alarms related to the status of that specific device, like battery alarms or presence on only 1 side.

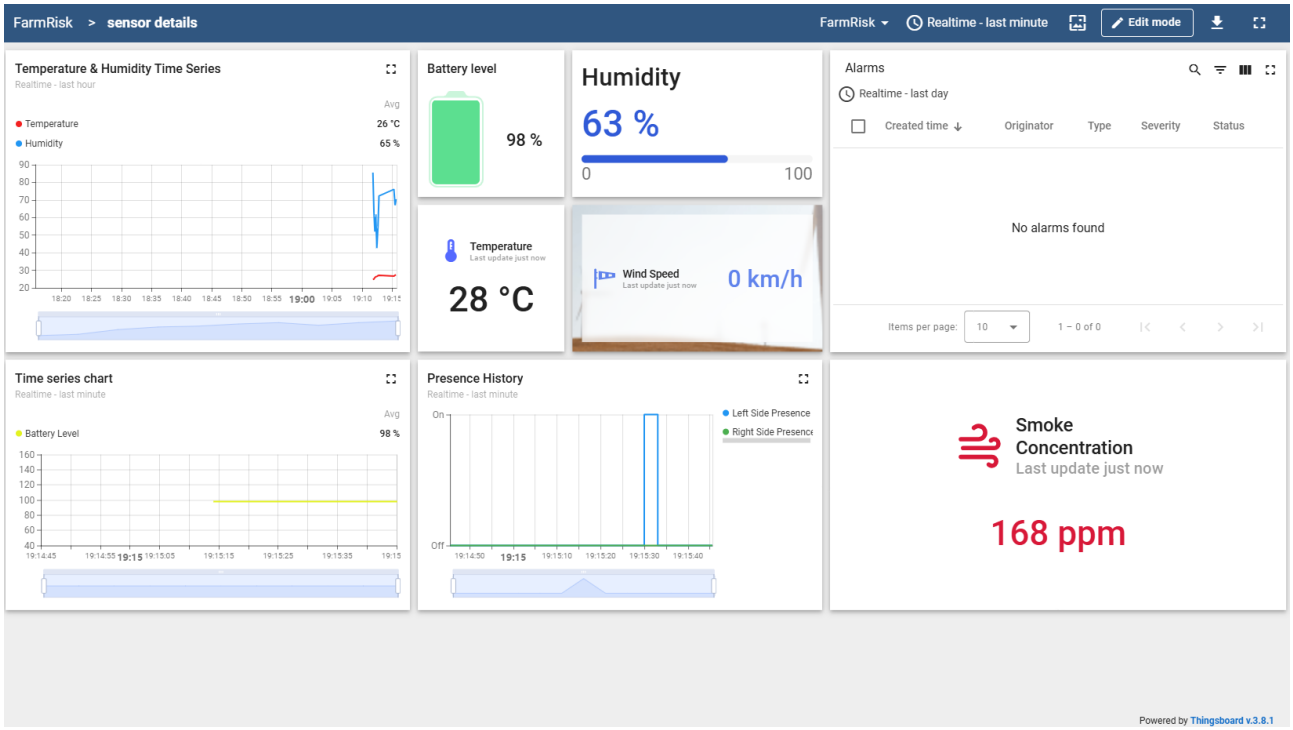


Figure 22: Node dashboard implemented in Thingsboard

8.3 Commanding the network

The possibility of network commanding is one of the key objectives of this project. This was implemented using the RPC (Remote Procedure Call) capabilities of ThingsBoard. In this case the implementation of RPC is done by sending a JSON in a MQTT topic and waiting for a response in another topic.

The commanding has two parts:

- An automatic commanding which is used when an animal is presented by a node. When all the information is processed in the edge, a response is generated and the edge platform command the node to activate the smart speaker to scare the animal. The format of this RPC is as follows:

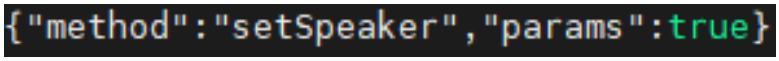


Figure 23: Automatic speaker RPC

- A manual commanding which is for turning off and on the sensing capabilities of the PIR nodes. This is implemented with two RPCs, as the edge platform needs to know before hand the state in which the node it's configured. The calls can be seen in the next figures:

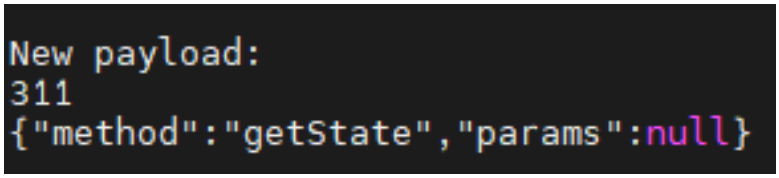


Figure 24: GetState RPC

Figure 25: GetState RPC Response

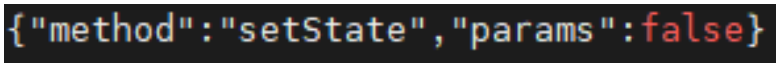


Figure 26: SetState RPC

8.4 Mobile Application

For this project an application was designed to integrate an easy to use interface.

The application has a MVVM (Model-View-ViewModel) with clean architecture, in this way the application has three main layers for the separation of functionalities, facilitating maintenance, scalability and testing of the application. The general system is presented in Figure 27.

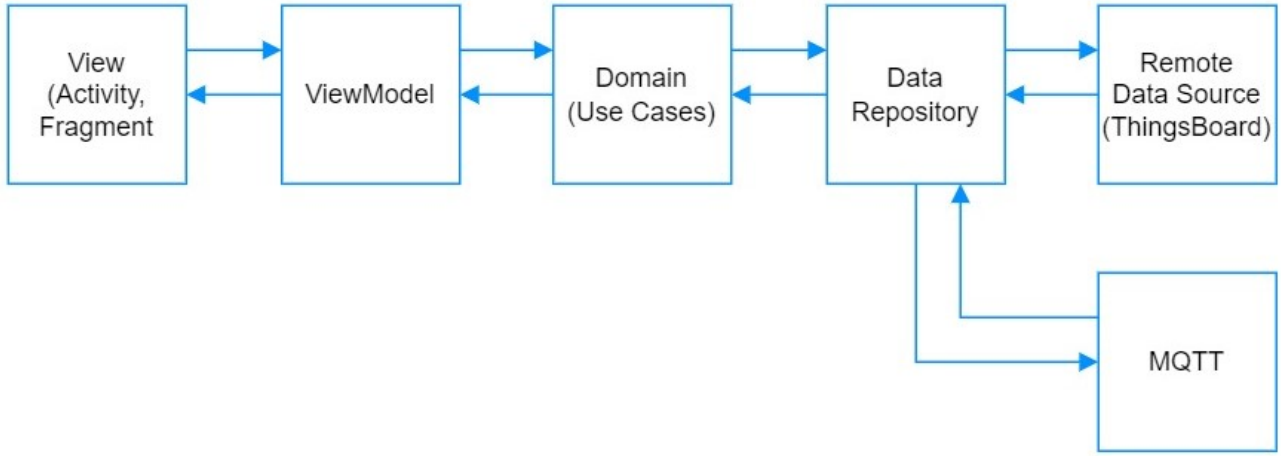


Figure 27: General architecture for the mobile application

- **Model**: This layer is responsible for managing data from external sources such as APIs or local databases, as well as transforming and storing them. It will apply the business rules that are established and will be the provider of the necessary data to the ViewModel layer.
- **View**: The View layer is responsible for presenting all the graphic elements, for data visualization, interaction with the user and adapting to changes in the data provided by the ViewModel.
- **ViewModel**: The View layer is responsible for presenting all the graphic elements, for data visualization, interaction with the user and adapting to changes in the data provided by the ViewModel.

App development

In the development of the app, a clear separation of responsibilities between business logic, user interface and data management has been ensured by using the architecture mentioned above. This decision has implied the need to use dependency injection, and for this, Hilt has been used, a tool that facilitates the management and provision of dependencies in an efficient manner. Throughout the next sections, the classes and methods developed will be detailed, encompassing the main functions of the application.

The app was built with a set of dependencies that help achieve our desired functionalities:

- **Dagger Hilt**: A dependency injection library for Android, built on top of Dagger. It simplifies the process of managing dependencies and promotes modular, testable code by automating the injection of objects into various parts of an Android application.
- **Retrofit**: A type-safe HTTP client for Android and Java. It makes it easy to interact with REST APIs by converting API responses into Java objects, handling network requests, and providing a convenient interface for dealing with asynchronous operations.
- **HiveMQTT**: A type-safe HTTP client for Android and Java. It makes it easy to interact with REST APIs by converting API responses into Java objects, handling network requests, and providing a convenient interface for dealing with asynchronous operations.

In the case of the usage of `Hilt`, the application uses four key modules for efficient dependency management and communication with the services:

- **ApiServiceModule**: A type-safe HTTP client for Android and Java. It makes it easy to interact with REST APIs by converting API responses into Java objects, handling network requests, and providing a convenient interface for dealing with asynchronous operations.
- **DatabaseModule**: Manages the local database. Provides the database instance, as well as access to the necessary data access objects.
- **MqttModule**: Configures and provides an asynchronous MQTT client for real-time communication with the MQTT broker. The HiveMQ library is used to establish connections and manage subscriptions and messages.
- **RetrofitModule**: Configures and provides an instance of Retrofit, enabling HTTP communication, as well as integrating Gson for handling data in JSON format.

For the `RetroFit` library, it was used to make API calls to Thingsboard. Out application is based around the next calls:

- **/api/auth/login**: User credentials are sent to obtain the access token.
- **/api/auth/token**: When the access token has expired, this call is used to refresh the access token.
- **/api/tenant/deviceInfos**: Retrieves a list of registered devices, filtered by deviceProfileId.
- **/api/plugins/telemetry/DEVICE/deviceId/values/timeseries**: This endpoint is used to obtain telemetry data from each of the devices identified by their deviceId.
- **/api/alarms**: Gets a list of active alarms on ThingsBoard.
- **/api/rpc/oneway/deviceId**: This endpoint is used to send an RPC command to a device identified by its deviceId.

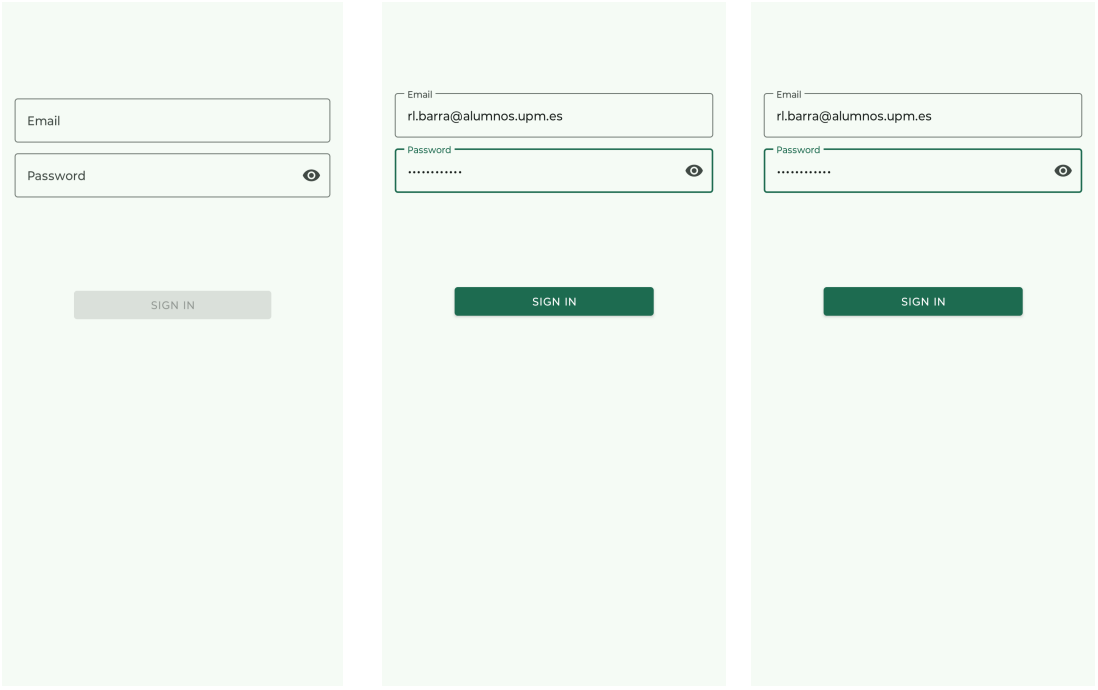
Finally, MQTT communication has been established with the free public broker offered by the HiveMQ platform itself so that users can share the status of the farm, and users who are close can be informed of any incident in real time. To do this, we use the MQTT client library, which offers functions to connect to the broker, publish messages, subscribe to topics and unsubscribe from them. The MQTT repository is responsible for managing all these functionalities. The following topics have been used to receive notifications sent from the ThingsBoard:

- **UPM/MIoT/ARS/RiskSystem/alarms/PresenceDetected**: Receive notifications about presence detection in the fence area.
- **UPM/MIoT/ARS/RiskSystem/alarms/FireHazard**: Receive alerts in case of fire risk in the area.
- **UPM/MIoT/ARS/RiskSystem/alarms/FireDetected**: Triggered when a fire is confirmed near a device.
- **UPM/MIoT/ARS/RiskSystem/alarms/LowBattery**: This notification is received when a device has a low battery.
- **UPM/MIoT/ARS/RiskSystem/alarms/FenceTrespassed**: Received when an animal has breached the fence.
- **UPM/MIoT/ARS/RiskSystem/alarms/NodePresenceDetected**: Triggered if an animal is near the device.

Screen design and flows

This section specifies the design and flow of the application. The focus was for the user to have an easy interaction and understanding of what's happening in the farm.

1. The user is prompted with a request for location permissions. If those are not accepted, the user won't be able to move to the login screen.
2. On the login screen, the user will be prompted to login with the same credentials as the one used in the ThingsBoard platform. This functionality can be seen in Figure 28.



(a) Login screen of the app (b) User input for the login (c) Error checking in login

Figure 28: Login screen flow

3. Once successfully logged in, the main screen of the application loads. It consists of a map with the points corresponding to our PIR devices and a delineated area representing the field to be protected. Additionally, there are two buttons: the first one, with a bell icon, will take us to the alarms screen; the second button will disable the devices so they stop collecting information.

   Additionally, clicking on a device on the map will display the following information: Node name, Latitude and Longitude, Temperature, Battery level, Smoke concentration, and Wind speed. This device information updates every 10 seconds.

   All this flow can be seen in Figure 29.

(a) General map view (b) Device click tooltip

Figure 29: Main screen of the application

4. As mentioned above, clicking the first button takes the user to a new screen that will display a list of active alarms found on ThingsBoard. It is presented in Figure 30.

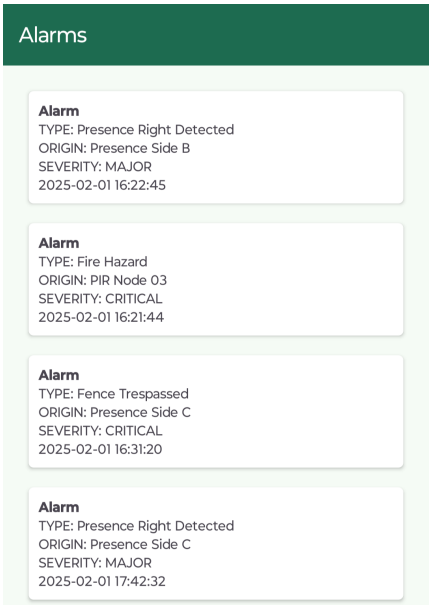


Figure 30: Active alarms list screen

5. The second button of the main screen is used to make an RPC call to ThingsBoard to enable or disable the devices. The usage can be seen in Figure 31.

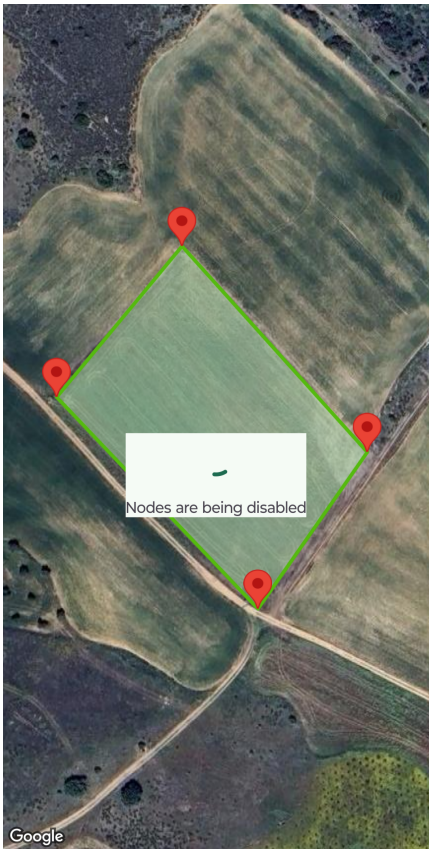


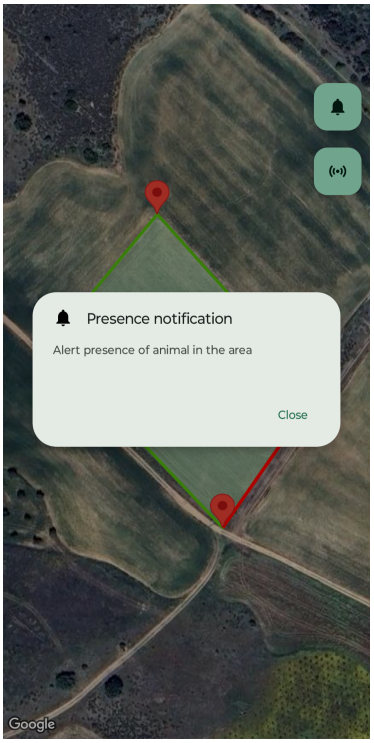
Figure 31: Node disabling from the application

6. Finally, one of the features of the mobile app is the reception of notifications sent from ThingsBoard. all notifications, when received display a dialog, activate an audible and vibrating alert on the mobile device, as well as making changes to the map screen. The notifications implemented are:

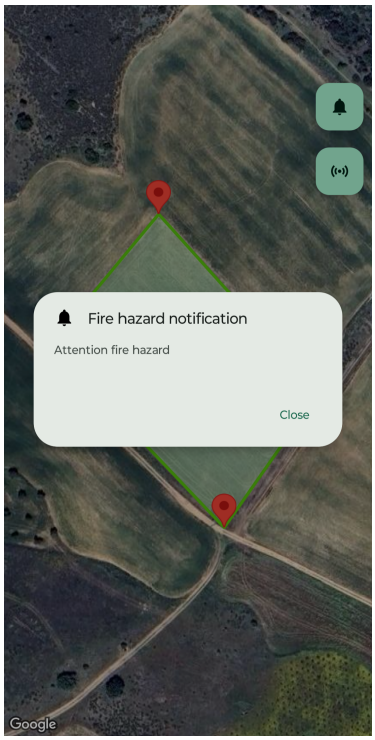
   - Presence Notification: This notification will display a dialog with the following message: ”Animal presence alert in the area” and will mark the affected area on the map in red.
   - Fire hazard notification: This notification only display a dialog with the following message: ”Attention fire hazard”.
   - Fire detected notification: This notification will display a dialog with the following message: ”A fire has been detected on the device (Name of the device)” and a flame icon is displayed on the map.
   - Battery notification: This notification only display a dialog with the following message: ”The device (Device name) has low battery”.
   - Fence Notification: This notification will display a dialog with the following message: ”The fence has been trespassed in (Device name)” and will mark the affected area on the map in yellow.

- Presence notification: This notification will display a dialog with the following message: ” Alert of animal presence near the device (Device name)” and a attention animal icon is displayed on the map.

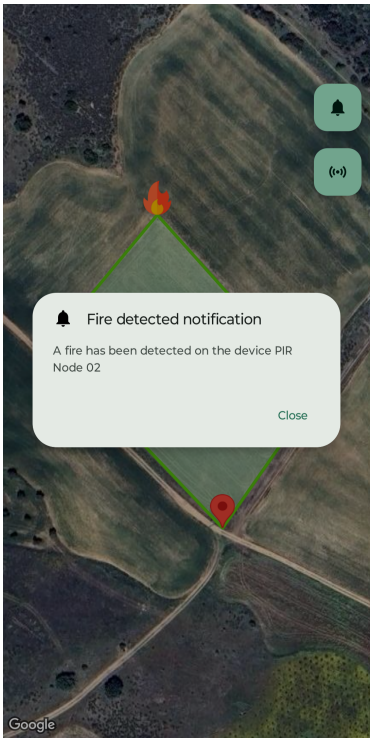
All these notifications are presented in Figure 32.



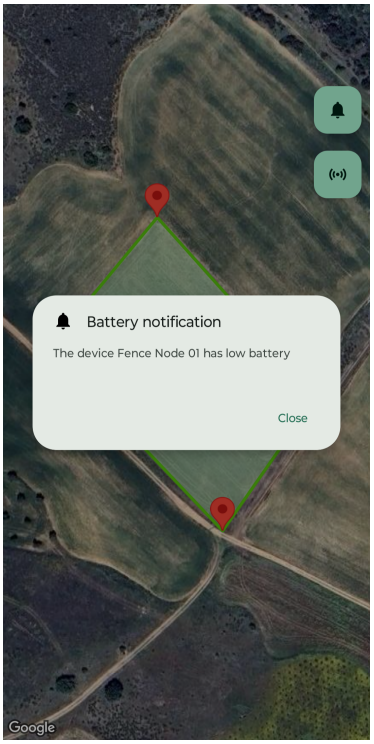
(a) Presence Notification



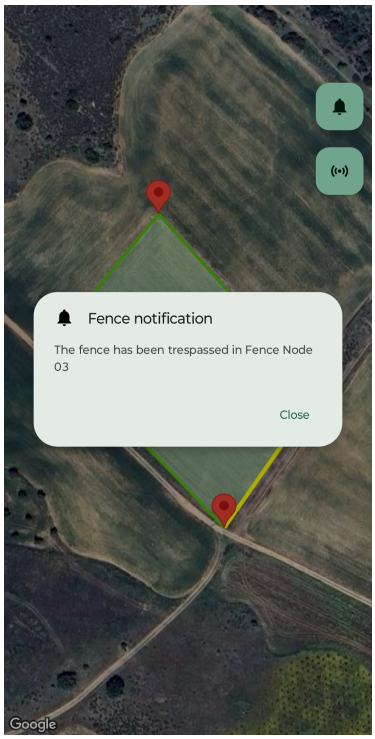
(b) Fire hazard notification



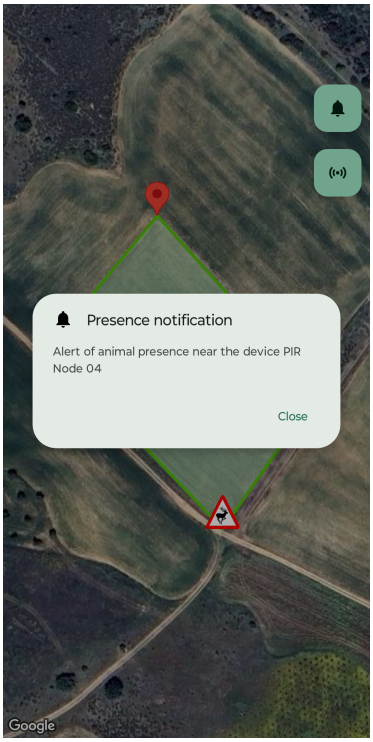
(c) Fire detected notification



(d) Low battery notification



(e) Fence notification



(f) Presence near notification

Figure 32: Set of notifications of the application

9 Testing

For the testing of the project, `NodeRed`[21] was used. This application allows for block-based programming and it's mainly focused on fast development for data pipelining and high level testing.

In this project, this tool was used to simulate the nodes and all the traffic sent by MQTT. The main flow constantly sends telemetry and for the validation, event are inserted manually. The flow can be seen in Figure 33.

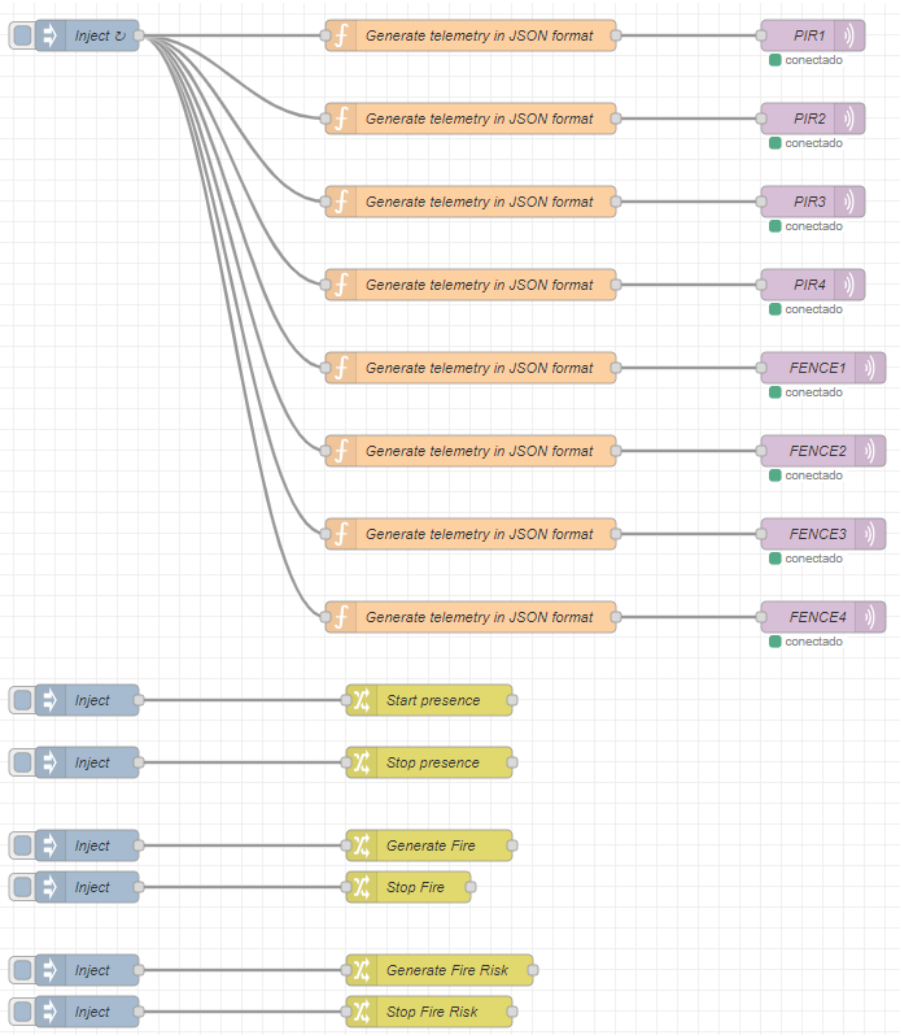


Figure 33: NodeRed Validation Flow

This tool was used to:

- Test the best format for the traffic of the nodes.
- Constantly check the correct implementation of the functionalities.
- Insert alarms in the system to further implement new functionalities in the dashboard.

Finally, the debug utility of ThingsBoard was well used in order to test insertion of data and other conditions. This ensures that the system is built in a more robust manner.

10 Validation

This section wants to present the efforts applies to ensure the complete functionality of the system as a whole.

First, a validation process was followed for each expected functionality. First, a set of independent component validations was made:

1. The validation of the nodes, both from the expected information sent to the edge platform and to ensure complete RPC integration.
2. The validation of the edge platform, mainly focused on the rule chains. This was done first without the nodes or node-red connected. Then, a constant flow of events was introduced in the system in order to check the correct functionality.
3. The validation of the mobile application involved cross-validation both in the rule chains and the events that appear on the app.
4. Lastly, a validation of the information sent to the knowledge base was done.

Finally, an integration validation was done, were all the nodes were sending events and different users were interacting with the system, both from the dashboard and the mobile application.

11 Conclusions and Future Work

Conclusions

The research done for this project shows the importance of IoT solutions in the farming industry. More specific, new solutions need to be achieved in the design of crop protection applications. And most importantly, fires are not the only risk that farm faces.

Some possible solutions that are being studied in this project are the use of PIR sensors in order to achieve a high level of power efficiency. If this reactive data is sent using low power wireless technologies like LoRaWAN, it can be use for new scenarios like machine learning, applied decision support algorithms and more. To support this, the use of MEC is mandatory.

In the first stage, the edge platform main functionalities and decision systems were implemented, along with simulated devices and dashboard solutions to give the user all the necessary information processed in a real-time manner.

In the second stage, the real hardware implementation of the nodes was achieved. In addition, an enhancement of the solution was made, building an application around the MEC platform in order to use the most common IoT devices, mobile phones. Finally, the possibility of commanding the network for cleaning tasks was added.

Finally all the necessary validations and test were made in order to check the required functionalities.

Future Work

Due to the limitations in time, there are some good ideas that could apply to this project:

1. The nodes need further enhancement in the selection of the MCU in order to obtain the lowest possible power consumption. Also, an encapsulation design for the nodes is needed.
2. Due to the limitations, the targeted technology of LoRaWAN wasn't used and needs to be tested. A gateway would be needed before the Thingsboard platform.
3. On of the key aspects that need to be tackled is the modification of the system in order to achieve full scalability. This could be done creating a mission manager that controls the entities in ThingsBoard, allowing creation and expansion with new nodes in an already existing farm.
4. Finally, we propose a system that utilizes the knowledge extraction database combined with weather data[22] in order to generate a predictor of animal intrusion and fire hazards.

12 References

[1] D. de Burgos, *"Ya no solo es vaciada, ahora estamos en la España quemada"*, Jul. 2022. Accessed: Dec. 30, 2024. [Online]. Available: https://www.diariodeburgos.es/noticia/z471df8f0-fcd1-f8ef-5d505495c8c2856f/202207/ya-no-solo-es-vaciada-ahora-estamos-en-la-espana-quemada.

[2] J. Arce, *La importancia de la caza en la España vaciada: una actividad que genera 186.750 puestos de trabajo*, Apr. 2022. Accessed: Dec. 30, 2024. [Online]. Available: https://revistajaraysedal.es/importancia-caza-espana-vaciada/.

[3] "(PDF) Wild Animals Intrusion Detection using Deep Learning Techniques", *ResearchGate*, Nov. 2024. DOI: 10.31838/ijpr/2020.12.04.164. Accessed: Jan. 1, 2025. [Online]. Available: https://www.researchgate.net/publication/344387269_Wild_Animals_Intrusion_Detection_using_Deep_Learning_Techniques.

[4] B. Dave, M. Mori, A. Bathani, and P. Goel, "Wild Animal Detection using YOLOv8", *Procedia Computer Science*, 3rd International Conference on Evolutionary Computing and Mobile Sustainable Networks (ICECMSN 2023), vol. 230, pp. 100–111, Jan. 2023, ISSN: 1877-0509. DOI: 10.1016/j.procs.2023.12.065. Accessed: Jan. 1, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050923020707.

[5] T. R. Hp, "Study of Methods for Animal Detection in Crop Fields", vol. 4, no. 2, Feb. 2023. [Online]. Available: https://ijrpr.com/uploads/V4ISSUE2/IJRPR9883.pdf.

[6] M. Knyva, D. Gailius, G. Balčiūnas, D. Pratašius, P. Kuzas, and A. Kukanauskaitė, "IoT Sensor Network for Wild-Animal Detection near Roads", *Sensors*, vol. 23, no. 21, p. 8929, Jan. 2023, ISSN: 1424-8220. DOI: 10.3390/s23218929. Accessed: Jan. 3, 2025. [Online]. Available: https://www.mdpi.com/1424-8220/23/21/8929.

[7] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on LPWA technology: LoRa and NB-IoT", *ICT Express*, vol. 3, no. 1, pp. 14–21, Mar. 2017, ISSN: 2405-9595. DOI: 10.1016/j.icte.2017.03.004. Accessed: Jan. 3, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959517300061.

[8] C. Gomez, J. C. Veras, R. Vidal, L. Casals, and J. Paradells, "A Sigfox Energy Consumption Model", *Sensors*, vol. 19, no. 3, p. 681, Jan. 2019, ISSN: 1424-8220. DOI: 10.3390/s19030681. Accessed: Jan. 3, 2025. [Online]. Available: https://www.mdpi.com/1424-8220/19/3/681.

[9] Panasonic, *Ast-ind-250613*. Accessed: Jan. 3, 2025. [Online]. Available: https://industrial.panasonic.com/cdbs/www-data/pdf/EWA0000/ast-ind-250613.pdf.

[10] D. S. Deepa, A. G, I. R, K. M, and P. K. S, "Implementation of Crop Protection System against Wild Animals Attack", *Annals of the Romanian Society for Cell Biology*, pp. 3638–3646, May 2021. Accessed: Dec. 30, 2024. [Online]. Available: http://annalsofrscb.ro/index.php/journal/article/view/5031.

[11] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey", *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb. 2018, ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2750180. Accessed: Dec. 26, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8030322.

[12] thingsboard, *ThingsBoard — Open-source IoT (Internet of Things) Platform*. Accessed: Jan. 5, 2025. [Online]. Available: https://thingsboard.io/.

[13] P. Kruchten, "Architectural Blueprints—The "4+1" View Model of Software Architecture",

[14] *Pico-w-datasheet*. Accessed: Jan. 31, 2025. [Online]. Available: https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf.

[15] *MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501*. Accessed: Jan. 31, 2025. [Online]. Available: https://puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf.

[16] *Mq2*. Accessed: Jan. 31, 2025. [Online]. Available: https://www.pololu.com/file/0j309/mq2.pdf.

[17] A. Industries, *DHT11 basic temperature-humidity sensor + extras*. Accessed: Jan. 31, 2025. [Online]. Available: https://www.adafruit.com/product/386.

[18] *Esp32-wroom-32_datasheet_en*. Accessed: Jan. 31, 2025. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf.

[19] *MPU-6000-Datasheet1*. Accessed: Jan. 31, 2025. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[20] EITB, *La norma 30-30-30: Los tres factores que indican que el riesgo de incendios es alto en Euskadi*, Jul. 2022. Accessed: Jan. 14, 2025. [Online]. Available: https://www.eitb.eus/es/noticias/sociedad/detalle/8909254/la-norma-303030-los-tres-factores-que-indican-que-riesgo-de-incendios-es-alto-en-euskadi/.

[21] *Node-RED*. Accessed: Jan. 15, 2025. [Online]. Available: https://nodered.org/.

[22] A. E. de Meteorología, *AEMET OpenData - Agencia Estatal de Meteorología - AEMET. Gobierno de España*. Accessed: Jan. 5, 2025. [Online]. Available: https://www.aemet.es/es/datos_abiertos/AEMET_OpenData.

13 APPENDIX

A. Work organization chart

TASK	NAME OF TEAM MEMBER	NAME OF TEAM	NAME OF TEAM	DESCRIPTION OF THE ACTIVITIES CARRIED OUT DURING THE WEEK	WORK HOURS
M1.Sketch of the project					
M1.1 Brain Storming	Dennis T.	Diego A.	Renato L.		3
M1.2 Final Decission meeting	Dennis T.	Diego A.	Renato L.	Decission on the topic for the project	2
M1.3 PPT Design and preparation	Dennis T.	Diego A.	Renato L.		3
M2. IoT Appplication Deployment - Stage 1					
M2.1 Task assignment meeting (26 december)	Dennis T.	Diego A.	Renato L.	Meeting to select who does the formating, research on sota and decisions on the constraint for the project	2
M2.2 Research on SOTA	Dennis T.	Diego A.	Renato L.	Research on projects and IoT systems related to crop protection	20
M2.3 Meeting for decisions on the solution (30 december)	Dennis T.	Diego A.	Renato L.	Decissions of the functionalities that will be presented for the first demo as well as the sensors to be used	2
M2.4 Report: Formating of the document in LaTeX		Diego A.		Preparing the format for the writting	2
M2.5 Report: Introduction		Diego A.			1
M2.6 Report: State Of The Art	Dennis T. (50%)	Diego A. (50%)			10
M2.7 StakeHolder and Requirement analysis	Dennis T. (5%)	Diego A. (5%)	Renato L.(90%)	Details on the specification of the requirements	12
M2.8 Draft of the Use Case diagram			Renato L.	Draft of all the possible use cases	10
M2.9 Meeting for decisions for the demo and on the report (5 January)	Dennis T.	Diego A.			2
M2.10 Sensors selection and buying		Diego A.		Selection done in relation to the needs and budget	3
M2.11 Report: System description		Diego A.			1
M2.12 Report: Project scope	Dennis T.				2
M2.13 Meeting for architecture validation and demo working start	Dennis T.	Diego A.	Renato L.	Mainly for aggregating decissions post-holidays	1.5
M2.14 Rule Chains Developement	Dennis T.			Development of the two defined rule chains	10
M2.14 Rule Chains Validation	Dennis T.(80%)	Diego A.(20%)		Validation of the correct functioning of the rules chains	5
M2.15 Research and buying of sensors		Diego A.			2
M2.16 Node Red		Diego A.			2
M2.17 Final Use Case validation and description			Renato L.		5
M2.18 Dashboard Integration		Diego A.			5
M2.19 Stage 1 document formating	Dennis T.	Diego A.	Renato L.		15
M2.20 PPT Design	Dennis T.	Diego A.	Renato L.		4
M3. IoT Application Deployment - Stage 2					
				Team Hours accumulated in the project:	124.5

Figure 34: Tasks for the first stage of the project