# END-TO-END IOT PLATFORM CONCENTRATING IN DATA AND DEVICE DIVERSITY IN AGRICULTURE 4.0

by

MARKOS – VASILEIOS FRAGKOPOULOS

H.M.U., School of Engineering (ScENG) MSc in Informatics & Multimedia

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SCHOOL OF ENGINEERING

HELLENIC MEDITERRANEAN UNIVERSITY

2023

Approved by:

Major Professor
Athanasios Malamos

# Copyright



# Άδεια χρήσης Creative Commons

# Abstract

Smart farming can be defined as the application of supplementary technologies to help minimize waste and boost productivity. Agriculture sector involves a huge number of heterogeneous devices that are used for collecting, transferring, exchanging and processing data. Integration into a common infrastructure of diverse data from such devices is challenging due to compatibility issues. Data fusion, data transmission protocols, and serialization formats are essential components of agriculture IoT-based solutions as they enable seamless communication, data exchange, and interoperability. Despite the fact that IoT technologies are promptly evolving, some issues concerning the interoperability and the semantic annotation of heterogeneous data have to be handled within rural deployments that necessitate meeting certain requirements such as long range and coverage in areas with challenging terrains, where radio communications are difficult or not available. We have developed a platform that can deal with data and device diversity while supporting edge processing and dynamic context-based operation profiles for end nodes, by leveraging low energy consumption communication protocols and ultra simple end-to-end deployment. On the edge there is an ARM-based single-board computer (SBC) Hybrid IoT node that is able to be adapted in any deployment. It's RTOS is based on a distributed middleware that supports heterogeneity and offers flexibility on working both as an extreme edge dummy node, as edge computing node with processing capabilities, or as a Fog gateway able to communicate with subnetworks. Data transaction between the end nodes and the cloud is agnostic and is feasible through an application-level zero-copy binary serialization approach. For the data transfer LoRaWAN offers us flexibility due to its support for multiple spreading factors (SFs) and device classes. On the cloud core application, all IoT devices are distributed to virtual subnets that are handled by functionally independent resource managers. End devices can start working automatically once they are registered according to the operating scenario and system wide preferences. Decision making is done through adaptable computational models that could be fused, depending on the use case with third-party data in order to enrich the application context and improve the decision efficiency.

# Table of Contents

# Chapter 1 - Introduction

Industrialized food production and distribution has facilitated the use of modern technologies such as sensor networks, satellite imagery, and cloud computing for remote monitoring and control of agricultural operations. Precision Agriculture and Smart Farming use technologies such as IoT, big data, and AI to improve agricultural practices and maximize productivity while minimizing waste. The Internet of Things (IoT) provides a global infrastructure that connects physical objects, allowing them to collect and exchange data. Low Power Wide Area Networks (LPWANs) facilitate long-range, low-power, and cost-effective communication. Big Data analysis can extract meaningful information from a large amount of data collected from various sources, including IoT devices. Computer vision involves developing algorithms and techniques to analyze images and extract information, which can be applied to monitor plant growth and yield. Artificial Intelligence (AI) enables machines to perform tasks that typically require human intelligence and it has a wide range of applications in agriculture, including predicting weather patterns, optimizing irrigation, and identifying pests and diseases. Within this context, multiple aspects of agriculture should be included in data collection, within scalable and interoperable smart farming systems that can support wide deployments and diverse low-cost and versatile IoT devices. Generally, IoT platforms serve as an interface between devices and end-users and offer services such as device management, data analytics, connection management, storage, processing, and visualization. It is significant for IoT-based agricultural solutions, to enable seamless communication, data exchange, and interoperability between devices, sensors, and systems. There are a lot of proprietary and open-source platforms that are designed to provide a standardized architecture for building IoT applications that can be easily integrated with different sensors and devices, making them suitable for various domains, including agriculture.

These platforms offer a range of tools for data collection, analysis, visualization, and support for machine learning and AI. Nevertheless there are remaining some critical issues and challenges in the development and implementation of IoT technologies in the agricultural sector. These issues include interoperability, standardization, connectivity, security, fault tolerance, energy management, data management, analytics, and sustainability. It is crucial to address these challenges to fully realize the potential of IoT in agriculture. In parallel, the agriculture industry

in many cases dictates long-distance communication in rural and suburban areas where there is no internet connection or cellular network coverage. While there are several solutions, LoRaWAN is suitable for rural IoT deployments due to its ability to handle a large number of devices, long battery life, low operating costs, but also because it is simple and has a low-cost and sustainable architecture. Nevertheless, there are some trade-offs between scalability, coverage, and power consumption in LoRaWAN deployments that can affect network performance.

The current thesis represents the development of an integrated IoT platform for agriculture 4.0 scenarios that overcomes challenges such as system simplicity, scalability, and energy optimization. Our solution is built on a 4-tier architecture: perception, network, processing, and application layers. It includes edge processing, dynamic context-based operation profiles, and low energy consumption communication protocols, while supporting data and device diversity and provides value-added services for crop management and precision agriculture.

Data transaction between the nodes and the cloud is agnostic, and the contextualization is done on the cloud using predefined configuration schemas. The data is structured in a pointer-based approach and uses bit-packing to generate a buffer sequence with interdependent offsets. The cloud core application handles IoT devices distributed to virtual subnets managed by resource managers. It includes the physical capabilities of edge devices as digital representations at the software level, creating a virtual ecosystem between end-users and remote end devices. The Cloud also includes a decision support service (DSS) that operates based on edge node data and manager-wide data processing models. The system uses adaptable computational models for decision-making based on sensor data collected at the edge nodes, with the option to fuse third-party data to improve decision efficiency. REST API endpoints provide read and write services for accessing and manipulating data in JSON format.

The IoT platform has been evaluated through two water resource management scenarios. The first scenario covers the management of water resources of multiple fields at the point of distribution, and the second scenario covers the autonomous management of irrigation of a farm consisting of several hectares of olive crops.

This thesis is organized into several chapters that aim to provide a comprehensive analysis of the current state and future potential of agriculture 4.0, as well as to propose an innovative solution for the challenges faced within IoT-based systems. In chapter 2 we are making an analysis of Agriculture 4.0 and its current state. In chapter 3 we are discussing the background technologies behind agriculture 4.0 digitization. In chapter 4 we are summarizing the most important aspects of IoT platforms while also presenting the state-of-the-art and assessing the communication background of such systems. In chapter 5 we are presenting our innovative solution, including its architecture, subsystems and workflow. In chapter 6 we summarize our evaluation use cases and describe them. In chapter 7 we are discussing the outcomes and conclusions that have been made. In chapter 8 we are representing our future work goal regarding our solution.

# Chapter 2 - Agriculture 4.0

## Agricultural Digitalization

It is accepted that global food production needs to be increased by 60% until 2050 due to the population growing to 9 billion people [2]. On the other hand the climate change that is enforced, between others, from the agriculture industry needs to be taken into account and the land needs to continue to be arable as much as possible, so fertilizers and pesticides usage should be minimized. Around 50% of the world's vegetated land is already used for food production where 70% of global freshwater is consumed, while 33% of soils are degraded by erosion [13]. Efficiency should be increased by simultaneously avoiding misuse of resources and the pollution of the environment. At the same time, food production meets several challenges, like the reduction of the workforce in rural areas, but also the increase in production costs, while sustainability tends to be crucial from now on, so the usage of natural resources, like water, should be reasonable [3]. These facts are giving more and more attention to the aforementioned agriculture orientation, where scaling is taking place for a decade now facilitating the industrialisation of food production. In our days, industrialized food production and distribution is dominating the global agriculture industry following the tendency of increased productivity and cost effectiveness [1]. This trend leverages a variety of modern technologies, such as sensor

networks, satellite imagery, and cloud computing, to enable remote monitoring and control of agricultural operations. [13]. Generally, Agriculture 4.0, as the 4th evolution in farming technology, is defined by four essential requirements: (a) increasing productivity, (b) allocating resources reasonably, (c) adapting to climate change, and (d) avoiding food waste [34].

Although, two important barriers are faced throughout this evolution period: lack of digitization and intelligence in agriculture. [1] Thus, the adoption of modern technologies within the agriculture domain is a slow process. Indicatively, 10% to 15% of US farmers are already using IoT solutions on their farms across 1200 million hectares and 250,000 farms [2]. Though it is considered that sensors will be used in approximately 525 million farms globally by the year 2050 [14]. Young farmers have a more positive attitude than older ones in utilization of new smart tools providing key information. However, the average age of farmers in the last decades has been remarkably increasing. For instance, in the USA and Europe the mean age is about 58 years old, 60 in sub-Saharan Africa or 63 in Japan. Generally, several motivations are given to farmers in order to support a generational renewal (e.g. European policies etc). Although, there are also opportunities for young farmers due to their high familiarization with technology. Data-less intuition-driven management will no longer represent the workflow of professional farms in the future. This is a positive assessment in which there is an upcoming balance of the aging population in rural areas, mainly in industrialized countries. [2]

## Precision agriculture

Precision Agriculture consists of applying what is needed when and where is needed [2]. Precision agriculture is not a new concept, but is still in its preliminary stage [1]. Precision agriculture emerged in the 1990s, as academic and industry groups began exploring the potential of information technology and automation to improve agricultural practices. [1]. The concept incorporates yield monitoring, guidance farming systems and variable rate applications.

It is proven that the adoption of precision agriculture technologies is increasing net returns and operating profits. For these reasons agricultural digitalization has been greatly accelerated globally and a lot of companies dealing with farm management have established [2]. Based on USDA, the highest adoption rates included three technologies named computer mapping, guidance, and variable-rate equipment. [2] Nevertheless, within the last thirty years the involvement of such technology in agriculture by the application of telecommunications and

automations has not yielded the results that were expected, but the technology evolution of the last decade seems promising for the further improvement of the efficiency of managing farms by the adoption of data-based digital systems. [2]

## Smart Farming

New technologies such as IoT, big data, computer vision, artificial intelligence, blockchain and fuzzy logic are playing an important role in agriculture's increased intelligence. Smart farming, which is the foundation of Agriculture 4.0, can be defined as the application of supplementary technologies to agricultural production techniques to help minimize waste and boost productivity [3]. It refers to the use of technologies such as the Internet of things (IoT) to collect data, monitor levels, detect patterns and prevent problems. Smart farming is not limited to performing precise measurements, but rather in how to access data and how these data will be used for applications [14]. The main objective is to identify how the collected information can be used intelligently [6]. This concept involves all farm operations [14]. For instance, the use of modern technologies can be utilized for collection of weather data, monitoring of crop's growth, early detection of crops diseases, prevention of crops wastage dues to effective harvesting of crops, monitoring of livestock's behavioral patterns, animal location within and outside the farms, increase of production for both crops and livestock. [14] Productivity, agri-food supply chain efficiency, food safety, and the use of natural resources are issues that will benefit from such an ecosystem that incorporates real-time farm management, a high degree of automation, and data-driven intelligent decision-making [1]. In that sense it is a logical consequence that agricultural tech startups have raised over 800 million dollars in the last five years [2].

## Farm management information systems (FMIS) and Agricultural Decision Support Systems (ADSS)

Crop management based on field data already evolved when Precision Agriculture firstly adopted thirty years ago. Though, the digital information era is greatly transforming the evolution of this domain [2]. Nowadays, large-scale monitoring of farmlands, yield forecasting and crop identification are available through remote sensing [1], thus a wide range of strategic and operational decisions can be supported [2]. The new era of cultivation systems include various innovative technologies, like IoT, Unmanned Aerial Vehicles (UAV), machine learning

etc, and moving beyond legacy decision support systems that were equipped with predefined time scheduling functions of the past [6].

Farm management information systems (FMIS) designed to assist farmers with their tasks, ranging from operational planning, implementation and documentation to the assessment of performed field work. Such a system may support the automation of data acquisition and monitoring, processing, planning, decision making, documenting, and managing the farm operations and include basic functions for record keeping like crop production rates (harvests and yields), profits and losses, farm tasks scheduling, soil nutrients tracking, weather prediction and field mapping, up to more complex functionalities for automating field management accounting for farms and agribusinesses (accounting, inventory management, or labor contracts) [2]. An important part of such systems is the Agriculture Decision Support System (ADSS) where the processing of the data happens, in order to facilitate the decision stage, by filtering routines and AI algorithms. Generally, an ADSS can be defined as a human-computer system which utilizes data from various sources, aiming at providing farmers with a list of advice for supporting their decision-making under different circumstances [34]. This helps getting only the right data and helping the grower make efficient decisions. In this way, an assessment by quantitative data produces objective decisions. In contrast, traditional farm growers judge by visual assessment, so decisions are relative and subjective. These decisions produced by an ADSS may be utilized for manual or even automatic actuation through the system. Actuation refers to the physical execution of an action which is commanded by the ADSS, and is typically carried out by advanced equipment on the field that can receive orders from a computerized control unit [2]. Illustratively there are various kinds of ADSS including among others crop data management, water resources management, mission planning, climate change adaptation and food waste control [1].

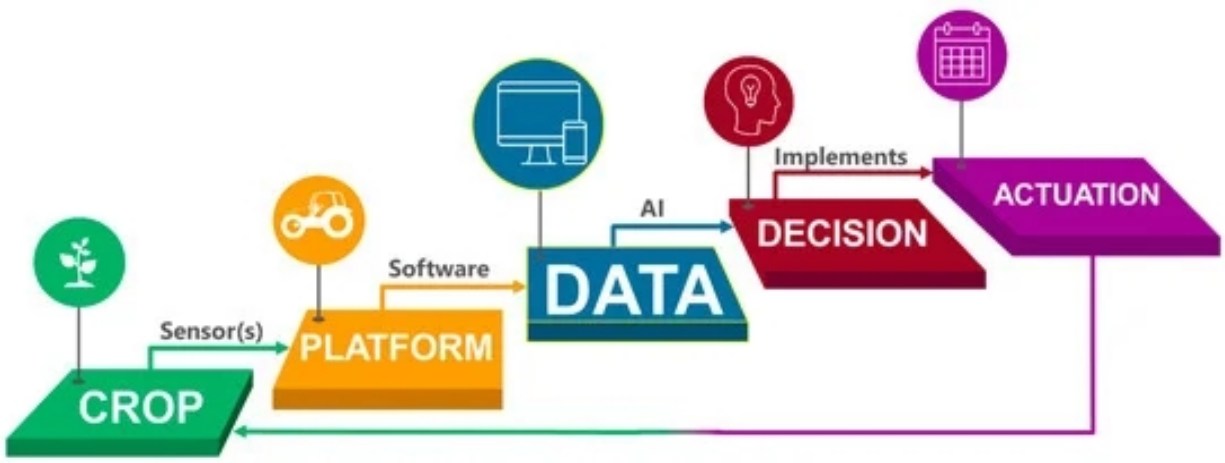


**Figure 2.1.** Cycle that embodies a general data-driven management system for advanced agriculture [2]

Nevertheless, at the moment the majority of systems that developed on the domain are still in prototype form (not commercial) and usually address a specific cultivation process or a set of them. Systems that incorporate the management of a group of cultivation processes, the processes of the whole cultivation period (from sowing to harvest), or even the processes for a whole farm are still missing [6]. The shift from just a specific crop, to platforms that can support multiple smart farming applications for crop productivity is crucial. These platforms should form systems without geographical limitations that can be easily modified to support multiple agricultural applications ranging from monitoring to managing [4]. Furthermore, to improve the automation capability, which is a necessity for autonomous tasks, full connectivity of agricultural machinery is very important during the whole production process [1]. At the moment, the decisive evolution of the whole agriculture domain is perceived as the incorporation of robotics within data-driven farms and embedded AI algorithms, which will lead to autonomous farming, which is noted by some researchers as the next agricultural revolution (Agriculture 5.0) [2].

# Chapter 3 - Background of Smart Farming solutions

## Technological advancements

### *IoT*

IoT may be defined as “A global infrastructure for the information society enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies” [28] In [28] the authors define IoT as a network of physical objects, devices, vehicles, buildings, and other items embedded with electronics, software, sensors, and network connectivity, that enables these objects to collect and exchange data. The goal of IoT is to make the physical world smarter by connecting everyday objects to the Internet, allowing them to communicate with each other and with other systems to make intelligent decisions and provide new services. IoT cloud platforms provide a framework for managing the data generated by IoT devices and making it accessible to other applications and services. Within the agriculture domain, IoT can be used to monitor environmental conditions in the fields, such as temperature, humidity, and soil moisture. This data can be used to optimize irrigation, fertilization, and other management practices.

### *Low Power Wide Area Networks (LPWANs)*

IoT applications for rural areas are mostly characterized by sensors reporting small amounts of data at a time, low cost and limited battery life. The geographical allocation of these sensors may be covering, from a suburban area to a whole country’s uninhabited areas. Such kinds of IoT deployments have some very specific requirements, in order to function but also to be viable, such as long range communications, low power consumption and cost effectiveness. [36] Application domains that are taking place in such suburban and wide areas, are precision agriculture, asset tracking, smart metering and more. Such applications have relaxed delay constraints and a reduced number of periodic messages, but at the same time their communication coverage must be wide enough to handle connections with dispersed location of end devices. [35] For these purposes network operators started to deploy machine-to-machine solutions to cover the necessities of market and vendors, by using low power wide area networking (LPWAN) technologies, because short-range radio protocols like ZigBee or Bluetooth can not be adapted for such scenarios that require long range transmissions. The fundamental constraints for the deployment of such solutions are limited in the ability of handling a huge batch of devices and the ability of functioning for several years with zero touch operation. LPWAN offers low power, long range and low-cost communication characteristics.

It includes technologies with long-range communication up to 10–40 km in rural zones, long battery lifetime of 10+ years, low cost of radio chipsets and low operating cost per device per year. In general to achieve such multi-kilometer communication range LPWAN protocols are combining low data rate and robust modulation. [35] LoRaWAN, NB-IoT and Sigfox are reported as the three leading LPWAN technologies that compete for large-scale IoT deployment. [36] Apart from these there is a wide range of other LPWAN protocols in the market like Ingenu, Weightless W, N and P, DASH7 and eMTC. However, LoRaWAN technology has received a lot of attention from network operators and solution providers. It is arguably the most adopted because it offers a great deal of flexibility, simplicity and performance [35]. Within the agriculture domain, LPWANs can be used to connect IoT devices over long distances and in areas with limited connectivity. This can enable farmers to monitor and manage their fields remotely and in real-time.

### *Big Data*

The large amount of data generated by sensors stored in databases and form the so-called Big Data [3]. With Big Data analysis meaningful information from a large amount of diverse data could be extracted [4]. In the current technology-based era, the concept of big data is present in many economic sectors, though in agriculture at the moment the volume of data retrieved from commercial fields cannot yet be considered as Big Data [2]. The concept can be characterized from six dimensions, namely Volume, Velocity, Variety, Veracity, and Valorization and Visualization [15]. Generally, within the smart farming concept the processing of Big Data may be used to obtain crop insights, optimize water resources and increase the crop quality by preventing disease and reducing the amount of chemical products employed [3]. Big data analytics can be used to analyze large volumes of data collected from various sources, including IoT devices, to identify trends, patterns, and insights that can inform decision-making.

Figure 3.1. The data chain of Big Data in Smart Farming

### *Computer Vision*

Computer vision is a field of study that focuses on enabling machines to interpret and understand visual data from the world around them, such as images or videos. It involves developing algorithms and techniques to enable computers to analyze and process images, and extract meaningful information from them. This includes tasks such as image recognition, object detection and tracking, facial recognition, scene reconstruction, and more. Computer vision has a wide range of applications in various industries, including healthcare, security, entertainment, and agriculture. [53] Within the agriculture domain, computer vision can be used to analyze images and videos of crops to detect diseases, pests, and other abnormalities. It can also be used to monitor plant growth and yield.

### *Artificial Intelligence*

Artificial intelligence (AI) refers to the development of computer systems that can perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation. AI involves the use of algorithms and statistical models to enable machines to learn from experience and improve their performance on a given task. AI is an interdisciplinary field that draws on computer science, engineering, mathematics, psychology, and other areas. Some of the key techniques used in AI include machine learning, deep learning, natural language processing, computer vision, and robotics.

AI has a wide range of applications in various industries, including healthcare, finance, transportation, and agriculture. In agriculture, AI can be used to improve crop yields, optimize resource management, and automate various tasks, such as planting and harvesting. [54]

### *Fog Computing*

Fog computing was introduced in 2012 by Cisco to assist Cloud computing to enhance QoS and to extend the kinds of supported applications within the IoT domain [16]. It is a paradigm of data processing distribution to the edge, where Fog gateways can intercommunicate and typically only the crucial data is sent to the Cloud [18]. Fog layer usually consists of Single-Board Computers (SBCs), which is the case for most Fog gateways setup, bringing network, processing, decision and storage closer to the source of data [28, 21]. Specifically, Fog gateways are converting raw data gathered from Fog nodes to smart data, by filtering, removing repetition, aggregating or classifying, which reduces the data size. In many cases Fog gateways are used to

run data processing or even decision based models providing a low latency availability of service on the edge, yet in scenarios without internet connectivity [16].

One significant purpose of fog computing is to leverage the on demand scalability of cloud computing resources by taking the advantage of both cloud and edge computing [4]. Among the advantages of Fog computing are the improved context-awareness, conserving network bandwidth, support for mobility, fewer resource requirements for edge devices, data fusion on the Fog layer and low latency real-time processing [24, 26, 29]. Sensors, gateways and services can potentially interoperate at different levels which enables the Collaborative IoT (C-IoT) concept where each of the three tiers (Edge, Fog, Cloud) manipulate and evolve the initial data within a context [19]. Fog computing can be used to process data locally on IoT devices or gateways, reducing latency and improving response time. This can be especially useful in real-time applications such as precision agriculture.

### *Digital Twins*

Evolutionaly, the distribution of virtual systems over separate processing environments that can work as a single system, where each subsystem or component can be fully simulated in several different computing environments is getting facilitated. Detecting errors, acquiring new information and predicting the behaviour of each subsystem or component are Digital Twins features that exploit the edge, fog and cloud computing capabilities and point to the "softwarization" of physical objects. The possibility to reconstruct a virtual representation of large environment contexts is strongly related to IoT because of the ability to measure and determine the state of physical objects [9]. Specifically, Digital Twin (DT) is an emerging concept within the IoT domain where physical objects (PO) have digital counterparts that are named Logical objects (LO). The physical entities that could be reflected, may include products, living and non-living resources, components and processes. Important properties, conditions, behaviour(s) and characteristics of the PO within a specific application context, are composing a comprehensive software representation through models and data (LO). Generally, the PO becomes context-aware within the deployed environment and its behaviour can be simulated through a set of realistic models. The capability to represent and deal with a constant flux of data turns out to be an enabler for Data Fusion, Artificial Intelligence (AI) and Machine Learning (ML) techniques [13, 14]. Within the agriculture domain, Digital twins can be used to create

virtual models of physical assets such as crops, soil, and machinery. These models can be used to simulate different scenarios and optimize management practices.

## IoT and Its Potential on Smart Farming

IoT within agriculture context refers to the transformation of every element and action during the production into data [2]. Within this context intelligent farms are using this technology to facilitate various stages of the production process, such as fertilizers and pesticides control, irrigation control, soil management, disease prevention, vehicles and machinery control, weed management, plant growth monitoring, etc [3]. Data-driven farms can increase efficiency by avoiding the misuse of resources and pollution of the environment [2], while it is estimated that with IoT solutions and new techniques the agricultural productivity can be increased up to 70% by 2050 [2]. At the moment there is evidence that the facilitation of IoT in some cases yielded 30% decrease in water consumption, 20% increase in crop production, reduction of labor cost by 60%, fertilization reduction by 60% and pesticides reduction by 80% [5].

### *Data-driven farming requirements*

A great challenge in agriculture is the utilization of big data because agricultural data are collected from individuals, research groups, and companies using different types of devices, which causes a multiple sources problem and also heterogeneity problems [1]. Additionally, the agriculture sector involves a huge number of heterogeneous devices that are used for collecting, transferring, exchanging and processing these data. The integration into a common infrastructure of diverse data from different sensors is challenging due to hardware and software compatibility issues [6]. Thus the interoperability issue is a high importance challenge for smart farming systems, that is not limited to the lack of standards for semantics, data modeling and agri-machinery, nor to technologies or protocols, but focus mainly on the great number of these devices and data, and the proper selection among them [5]. Additionally, the integration and multi-functionality of many, different protocol versions in IoT hardware can contribute towards a much more effective performance [5]. Summarily, within smart farming, collection of data from multiple aspects of agriculture is necessary, both on the extreme edge but also from third sources (e.g. Earth observation data, weather forecasts etc). Crop, substrate, environment and more need to be monitored within cultivations. Typically electronic sensors are used to collect environmental data, such as luminosity, temperature and humidity, or for monitoring the

temperature, moisture and nitrogen of substrate, or even for measuring the acidity or the alkalinity of the water in hydroponic cultivations. Cameras and multispectral sensors are used for crop monitoring, both installed on UAVs to obtain aerial images of large plantations or even used in robots to retrieve detailed images of plant leaves [3]. These devices, due to the nature of farming, need to be widely deployed [5], in rural areas, where sometimes there is no telecom infrastructure [1], and often have limited or no energy supply [3]. Different types of wireless communication technologies based on radio frequency, sonar, vibration, and other signals could be required for information exchange depending on the case. Additionally, the physical equipment is directly exposed to harsh environmental experience such as rain, high level temperature, extreme humidity, hard winds and other possible dangers which can destroy the electronic circuits [4]. It is noted that there is also an impact of vegetation in the signal propagation, and more oftenly sensor node communication is affected by rain, snow or solar radiation [3]. Such weather conditions and the dynamic agricultural environment dictate that robust network protocols are required to cope with such conditions, but also to overcome issues related to wireless interference due to heterogeneous agricultural IoT networks and the dense deployment [1]. To avoid quality degradation of the service several methods need to be utilized like, efficient channel scheduling between heterogeneous sensing devices, cognitive radio-assisted WSNs and concurrent transmissions [1]. Network deployments are getting even more complicated due to the potential diversity of network size, node density, transmission distance, throughput and latency between different agricultural use cases [1], while the communication under different physical layers and multi-protocol chains is essential [1, 5]. Specifically, cross-technology communication is needed between cellular-based networks, Bluetooth-low-energy networks, 802.15.4 mesh networks and LoRa networks that would coexist in the same location [1]. Typically, implementation of networks in an IoT agriculture ecosystem may be physically restricted, while in most cases we have to deal with farming applications over wide geographical areas. There is a requirement for architectures that are scalar and can support wide deployments, preferably separated in several smaller networks which could minimize the operational hazards over a huge agricultural area [5]. Edge and fog computing utilization in smart farming could be a way to deal with challenges associated with centralized cloud computing solutions such as high communication latencies, lack of support for real-time reaction to detected events, large bandwidths, etc. [3] In any case, new devices should be able to be added over the existing

infrastructure without affecting the functionality, the performance and QoS, to make the systems scalable which is a key factor [5]. In any case, developers will be influenced on their choices from the availability of an IoT application protocol, the cost of implementing or integrating these protocols, the easiness of development, the openness of software or the existence of suitable hardware and software that can be used to deliver the desired services within the frame of IoT [5]. Smart farming products have to consist of low cost but versatile hardware, like Single board computers (SBCs) [3], but also software, in order to be globally available to markets with economic diversity [5]. For instance, Wireless Sensor Networks (WSNs) at the moment are small scale and short term due to the high deployment and maintenance cost [1]. Improved signal coverage in rural areas, based on telecommunications infrastructure upgrade will help to increase agricultural productivity, while network bandwidth and delivery latency improvements may ensure large-scale high-throughput plant phenotyping [1]. Reliability is required from cases of simple measurements with low periodicity, to real-time multimedia that may have huge variability in velocity. Unreliable data can lead to inaccurate decisions and inappropriate automated procedures which may result in significant or even intolerable cost. Critical factors that affect data veracity and accuracy need to be taken into account, such as diverse and unpredictable propagation conditions which may range from free space to severe attenuation and fading [5]. To evaluate all these requirements, fully developed IoT networks with a huge number of nodes under different environmental conditions that create a big volume of diverse data should be tested in real-world and not only in emulations [5].

### *Escalating with Edge & Fog computing*

Modern ICT domain and especially IoT technology implies a tremendous amount of raw data to be generated and an exponential amount of computation power. It is estimated that the annual amount of data generated by the millions of globally deployed IoT sensors in the near future will be of the order of zettabytes. Within this context the viability of handling these data from cloud centric systems is doutable [17]. Additionally, energy consumption and operational costs of Fog computing systems are very low in comparison to traditional Cloud systems [20]. These aspects turn the adoption of Fog computing into a sustainable and efficient solution for high performance regional IoT applications with a relatively lesser carbon footprint [17]. An International Data Corporation prediction showed that the amount of data processed on devices

with high proximity to the edge is approaching 40%, which indicates the rate of Fog computing adoption [23].

Within the agriculture domain there are a lot of cases that mobility support, data processing, better power management, high hardware costs and poor internet connectivity should be addressed by utilising appropriate technologies [20]. Especially, when dealing with a range of time-sensitive tools and technologies such as UAVs and UGVs that are requiring immediate response to actions and co-operation support, data processing should happen in real time and in high proximity. Alongside, irrigation systems, phenological observation models etc, are delay sensitive applications, requiring real-time control for better performance [21]. Furthermore, the wide density of deployments over rural areas drive the adoption of Low-Power Wide Area Networks (LPWAN) communication protocols that have the disadvantage of low-bandwidth transmissions. Fog computing deals with this issue, by implementing data compression, data aggregation and data processing methods near end devices and thus reducing the large amount of data to be transmitted [22]. Although, at the moment the state of research for agriculture oriented publications, is noticed to be mainly limited to Cloud-based approaches without the utilization of the Fog computing paradigm [20]. While smart agriculture is evolving, the proposed solutions are based mostly on the Cloud computing model, which is not an environmentally friendly technology. Cloud computing majorly contributes to global warming due to ever increasing carbon emissions of huge data centers which could negate the benefits gained from smart agriculture. On the other hand, Fog computing by reducing the large amount of data to be transmitted can augment Cloud computing and play a pivotal role in the growth of smart agriculture in sustainable terms. United Nations Sustainable Development Goals (SDG) emphasize on climate protection and smart agriculture for sustainable development. [17]

### *Improving efficiency with Digital Twins*

Manufacturing, Healthcare, Autonomous Vehicles, and Aviation are the main domains that are at the moment gaining the benefits of the Digital Twin technology [12]. While the concept has an identified potential for application in agriculture, the research outcomes are still in early stages. At the moment there are a lot of prototypes arising within the domain, but there is a lack of openness over these solutions that is a reason for not being widely used [9]. Specifically, at the moment there is a lack of case studies and models [13]. This kind of slow

adoption can be reasoned by the fact that agriculture is a highly complex and dynamic domain because the production process introduce aspects like the mobility of resources (like cattle), lack of communication in rural areas, unpredictable environment, continuous climatic changes, lack of willingness to share farm information and lack of technical skills of farmers [12].

From this perspective, the development of Digital Twins and their facilitation within agriculture domain case studies is still an open challenge that seems promising as an enabler for the evolution of sustainable agriculture and the increasing smart farming efficiency.

While Digital Twins is a relatively new concept, it is already implicitly used within smart farming but most existing applications, including ours, focus on basic monitoring capabilities [8].

By utilising Digital Twins, a bigger picture of the different aspects and parameters that impact a farm's behaviour, the final yield production and resource consumption will be available. [7] Resources will be saved, irreversible damages will be treated precautionarily and critical decision-making will be facilitated. Spatiotemporal characteristics and human observation, that are fundamental constraints for remote and automated monitoring, control, and coordination of farm operations, are decoupled from planning and control. Physical proximity could be optional as long as information from sensors, satellites or other holders, that also oftenly cannot be counted in by humans, are enriching the digital counterparts' context. [8] The Digital Twin concept is already well accepted by both the academic and industrial environments, though it is kind of abstract and its implementation in IoT software may be ambiguous. [13, 14]

Nevertheless, with a typical implementation of this concept, producers and customers can have a clear idea on the functioning of a product in each moment of its lifecycle through this kind of softwarization, which is taking over the processes of many industries, converging to an increased rate of servitization, which however is controversial for many people.

## IoT Platforms

### *Overview*

Despite the fact that there are a lot of definitions for what an IoT platform is, a common sense can be extracted. An IoT platform can be mainly described as an interface between devices and end-users, responsible for devices management and control, as well as gathering, managing and analyzing data to achieve the goals of IoT solutions. Generally, IoT ecosystems are offering

services through gathering and processing end device data. Data analytics, device management, connection management, storage, processing and visualization are mandatory mechanisms that should be delivered with these ecosystems in order to be able to provide central management services to the end user [29]. Additionally, it can be concluded that generally, operating systems of devices, gateways, application development tools and platforms, but also central management services are considered as IoT platform's components. Providing a set of development tools and application deployment is considered as the most important function of an IoT platform [29]. In a more comprehensive and novel approach based on [28], the concept of IoT cloud platforms may be formulated by definition as: "a platform offered by a service provider as a hosted service which facilitates the deployment of software applications without the cost and complexity of acquiring and managing the underlying hardware and software layers to hinder a model designed to facilitate the information society, enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies through ennoblement of ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction that leverage the need and heterogeneous connectivity issues of the user centric things in well defined fashion". In simple terms, IoT platforms have a common goal, to simplify the input and the utilization of data from all kinds of sources, using a common Application Programming Interface (API) [6]. In focus, data fusion, data transmission protocols, and serialization formats are the most essential components of agriculture IoT-based solutions as they enable seamless communication, data exchange, and interoperability between devices, sensors, and systems.

### *Data Fusion*

Data fusion (DF) is a concept that includes the theory, techniques and tools which are used to combine sensor data or data derived from sensory data (e.g. third party data) and construct a common representational format. In this way information coming from multiple sensors and sources gets more intelligent, decisive, sensible and precise, while enabling the ability to extract inferences that are not feasible from a single sensor. Data from a sensor alone may have no sense on its own, which is a common case, because of the increased establishment

of low power sensors that achieve energy efficiency but have low accuracy [25]. Different types of information are handled in different ways in most modern DF applications. Information is distinguished into three different categories with different nature and potential support to the output of the fusion process: (a) Observational Data are collected from heterogeneous sensors with different observational capability of real world entities of interest, (b) Contextual Information can be defined as the set of circumstances surrounding a task that are potentially of relevance to its completion, (c) Learned Knowledge is the potential extraction of non-existing knowledge, such as relationships among targets and behaviors of entities of interest, through online machine learning processes operating on observational data and context information [27]. Fusing data from heterogeneous observations brings the potential to extract complex multivariate relationships among data sets. Though, the transformation of heterogeneous data from several feature spaces to homogenous space is required. Hopefully, Artificial neural networks (ANNs) can extract patterns and find new trends in highly complex data sets by complex training and learning, while they support adaptive learning with self-organization in a real time environment and can achieve a high degree of fault tolerance [25]. Furthermore, DF has multiple challenges to overcome such as, sensor data imperfection including impreciseness and inconsistencies, conflicts in data, data alignment (sensor registration problem), velocity of data triviality and iterative nature of such algorithms. When DF is utilized within an IoT platform these challenges are forced to be lateral constraints for the whole IoT ecosystem [25]. Agriculture IoT solutions generate massive amounts of data from various sources, such as sensors, drones, weather stations, and satellite imagery. Data fusion techniques can combine such data from different sources and sensors to create more accurate and reliable data that can be used for decision-making. For example, data fusion can combine data from weather sensors, soil sensors, and crop sensors to provide farmers with more precise information about soil moisture, temperature, and nutrient levels.

### *Data transmission protocols*

In IoT ecosystems, the communication network architecture and technology play a crucial role in collecting and managing data. The diverse requirements of IoT agricultural applications, including data types and node installation environments, pose significant challenges related to the volume, variety, veracity, and velocity of data. To address these challenges, two

major protocols have emerged as effective solutions: Message Queue Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) [5]. Such protocols are facilitating the communication between non-standardized IoT devices and IoT platforms [3]. MQTT is a bandwidth-efficient machine to machine (M2M) network protocol, introduced in 1999 and is based on the publish/subscribe model. Therefore is suitable for devices with resource constraints (limited computing resources) and non-ideal networking conditions. It uses little battery power and is mainly designed for receiving and transmitting sensor information [4]. It is one of the most popular choices for IoT applications due to its simplicity and the very small header size. MQTT defines three QoS levels through which the message could not be confirmed or stored by the receiver, the message could be followed by a confirmation message with the option of many retransmissions, or each message is received once and only once by the receiver with confirmation (four messages being exchanged between the client and the broker). MQTT clients should support TCP and will typically hold a connection open to the broker continuously. [5] The CoAP (Constrained Application Protocol) protocol was developed by the Internet Engineering Task Force CoRE (Constrained RESTful Environments) to cater to devices with limited computing capabilities. It is a Web transfer protocol that operates based on the principles of the Representational State Transfer (REST) architecture and is specifically designed for use in constrained environments. CoAP is known for its lightweight nature, offering minimal overhead. When using CoAP, clients send requests to servers, but unlike traditional request-response protocols, the corresponding response is not sent over a pre-existing connection. Instead, it is sent asynchronously, allowing for efficient communication and resource utilization in constrained environments. [5] MQTT with QoS = 0 provides balanced trade-off between latency and transmitted data volume when reliability is not crucial, when it is, instead of using MQTT QoS = 1, CoAP offers a better trade-off [11]. Although, generally CoAP has UDP embedded which is not that reliable as TCP that MQTT uses. On the other hand UDP offers the lowest overhead and presents lower bandwidth requirements than MQTT, but with worse throughput. MQTT was found to be more energy efficient than CoAP, a parameter that has very significant importance because networking protocol plays a pivotal role in the overall energy consumption of IoT nodes. At the moment, MQTT seems to be the safest option, either for end-to-end network architectures, or for gateway-server architectures. It is noted that in some large scale implementations, MQTT is combined with REST HTTP. Especially for agricultural applications

that have already been deployed using HTTP, usage of MQTT for IoT node-to-gateway communication is beneficial. Also, it is preferable for smart farming due to its resiliency, its interoperability across different network protocols and its transmission rate [3].

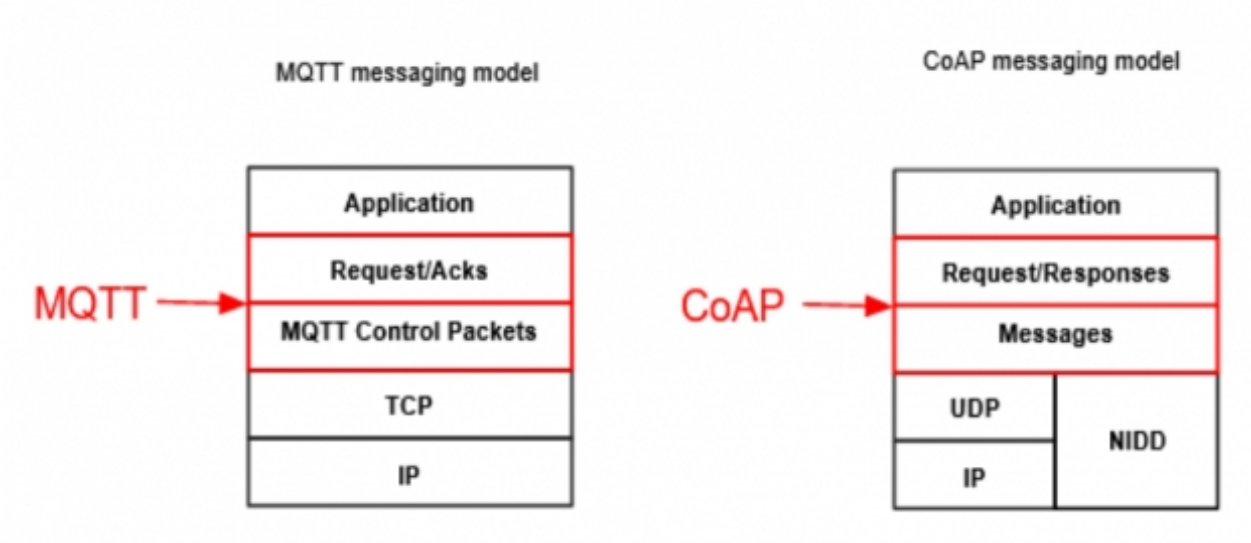


**Figure 3.2.** Comparison of MQTT and CoAP messaging models

### *Serialization formats*

Serialization is the process that translates an in-memory representation of a data structure into a language and architecture independent streamlined format ready to be sent to another peer across the network or stored locally to disk. Serialization formats are differing on how the schema is described or on how the data are encoded. Respectively, there are schema-less (self-describing) or schema-based methods, text-based or binary encoding methods. A great variety of data serialization formats have been proposed over the years, assembling different trade-offs like human readability versus space efficiency or performance versus expressiveness [32]. On self-describing serialization formats, text-based data is perceived as data that are typically structured into key-value pairs in a human readable text format. In binary serialization, the same key-value pairs are encoded in binary format which is a method that typically reduces the space requirements. Schema based methods are facilitated to further reduce space requirements, while the schema can be used for both serialization and deserialization processes. [11] Thus, serialization formats play a pivotal role when message size needs to be reduced. Reductions in payload size are helping in the reduction of transfer times, in lowering the risk of packets being dropped and in reducing transfer costs when using LPWANs or even mobile networks. Google has lead the way for such methods because it has released two formats that are widely adopted

and very efficient. Protobuf supports binary data serialization with great performance based on predefined message schemas, while Flatbuffers is a zero-copy serialization format designed to have shorter deserialization time and use less memory [11]. Additionally, there are also MessagePack and BSON that offer high flexibility, when implemented in an end-to-end application, because they are schema-less formats with significant small size and lightweight codebases [33]. However, schema-less formats introduce larger payload sizes which comes as a trade-off between their flexibility. MessagePack library shows significant fast string serialization and deserialization with extremely low power consumption. Thus, its schema-less implementations are basically directed towards embedded systems and low-level platforms. Alongside, the schema-based Protobuf protocol shows an adequate and resource-efficient performance when handling typical sensor node payloads. [33] Additionally, Protobuf achieves better size reduction than Flatbuffers make it more suitable for end-to-end communication scenarios as both latency and bandwidth usage are related to the message size. Flatbuffers format is based on a zero-copy methodology which means that serialization should structure the data in the same way as in working memory. Thus, the pointer-based serialized data has a specific offset in memory from the start of the buffer which removes the need for expensive encoding and decoding processing that speeds up deserialization. In this way, Flatbuffers show higher processing and serialization times only on the sender side. [11] Generally, in many cases serialization could be a bottleneck for applications. Typically, during serialization the application accesses fresh data recently received over the network. This requires a number of steps. Accessing encoded data for the first time results in cold misses through memory hierarchy. Later the CPU must perform the computations for data encoding or decoding. It ends by writing the native data back to memory [32].

### *Scalability*

Scalability is an important consideration when it comes to IoT platforms, especially those that involve large numbers of devices and data streams. In addition to using containerization technologies like Docker, platforms can employ a range of techniques to ensure scalability. One such technique is to use a microservices architecture, which involves breaking down an application into smaller, loosely coupled services that can be developed, deployed, and scaled independently. This allows the platform to be more agile and flexible, since different services

can be scaled up or down as needed to meet changing demand. Another technique is to use horizontal scaling, which involves adding more nodes or servers to a system to increase its capacity. This can be achieved through load balancing, clustering, and other techniques that distribute processing across multiple machines. Regarding the management of IoT devices, platforms can use a range of techniques to ensure scalability. These might include device auto-provisioning, remote device management, and the use of edge computing to distribute processing and storage closer to the devices themselves. Additionally, platforms can use data compression and other techniques to reduce the amount of data transmitted from devices to the cloud, which can improve scalability and reduce costs. Auto-provisioning is a process in which devices or resources are automatically provisioned, or configured, without the need for human intervention. In the context of IoT platforms, auto-provisioning refers to the automatic provisioning of IoT devices on the platform, including the assignment of unique identifiers, authentication credentials, and other configuration settings. Auto-provisioning can save time and reduce errors that may occur when manually provisioning devices, especially in large-scale IoT deployments.

### *Domain independent solutions*

Numerous multinational corporations are currently engaged in the development of their own IoT ecosystem platforms, employing pay-per-service cloud services such as Amazon's AWS IoT, Microsoft's Azure IoT Suite, Facebook's Parse platform, Samsung's ARTIK technology, ARM's ARM mbed, and more. These platforms are characterized as proprietary and closed, as they operate exclusively within a cloud deployment managed by a third-party entity. Conversely, there exist open source initiatives that readily facilitate utilization and customization for academic, industrial, and scientific research through private deployments. Moreover, researchers are also undertaking prototyping endeavors in this field. [37,40]

#### *ThingSpeak*

ThingSpeak is an open IoT platform developed by MathWorks that provides capabilities to collect, store, analyze, visualize, and act on data from sensors or other devices. It is built on top of the MATLAB analytics platform, which is designed to support large-scale data analysis and processing. It supports several protocols such as MQTT, HTTP, and HTTPS, and provides APIs for integration with other applications. ThingSpeak enjoys a robust user community and finds extensive applications in agriculture and various other industries for the purpose of data

monitoring and analysis. It facilitates real-time data collection, analysis, and actuation through its Open API. This platform encompasses a range of plugins designed for tasks like data storage, visualization, monitoring, and integration with diverse third-party platforms. Notably, it seamlessly integrates with leading IoT platforms such as ioBridge, Arduino, Twilio, Twitter, ThingHTTP, and MATLAB. ThingSpeak offers several supported reactions, including automated execution of actions at predetermined times, real-time responsiveness to Twitter activity, triggering responses based on specific conditions within the channel data, and queueing up commands for the user's device. One negative issue is that few devices can connect simultaneously. ThingSpeak is an open-source platform for building IoT applications. The platform provides a range of tools for data collection, analysis, and visualization, as well as support for machine learning and AI. The development process typically involves creating a custom configuration using the ThingSpeak API, writing custom code, and deploying the application to a cloud-based or on-premise infrastructure.[28]

***Open Remote***

Open Remote is an open-source IoT platform that allows users to monitor and control various devices and sensors. It is built on Java and can run on various platforms such as Windows, Linux, and Android. Open Remote has a modular architecture, and users can create their own modules or integrate third-party modules. It supports various communication protocols such as MQTT, HTTP, and WebSocket. The middleware of OpenRemote provides users with the flexibility to integrate any device or protocol by utilizing available resources such as iOS, Android, or web browsers. By leveraging OpenRemote's cloud service, users have the ability to design and develop highly personalized solutions that can incorporate a wide range of protocols, including but not limited to Wi-Fi and ZigBee. This empowers users to create customized solutions that seamlessly integrate diverse devices and protocols according to their specific requirements. The most negative issue is that it is too costly for developers. The platform provides a range of tools for data collection, analysis, and visualization, as well as support for machine learning and AI. The development process typically involves creating a custom configuration using the Open Remote Designer, writing custom code, and deploying the application to a cloud-based or on-premise infrastructure. [28]

***Thinger.io***

Thinger.io is an open source platform for deploying data fusion applications in IoT environments. It is able to collect, manage and analyze big amounts of heterogeneous sensor data. It supports several communication protocols such as MQTT and CoAP and provides APIs for integration with other applications. Thinger.io also provides several features such as data logging, rules engine, and dashboard creation. The preceding research effort behind the Thinger.io platform successfully addressed various challenges associated with transmission efficiency, real-time bidirectional communication, interoperability, and simplified deployment of data fusion applications. Thinger.io stands out for its unique capability to model sensorized environments using a high-level language, facilitating straightforward implementation of data fusion applications. Moreover, it is hardware-agnostic, ensuring scalability and cost-effectiveness. Thinger.io incorporates its own modeling language, allowing designers to transparently model services, supported by efficient communication protocols. Furthermore, Thinger.io promotes interoperability with other platforms, enabling third-party access to devices for sensing and actuation via a REST API, effectively abstracting the underlying protocol optimizations between devices and the server. The wide community that maintains, evaluates and tests the software is creating a confidence that is important for production deployments. The platform provides a range of tools for data collection, analysis, and visualization, as well as support for machine learning and AI. The development process typically involves creating a custom configuration using the Thinger.io Dashboard, writing custom code, and deploying the application to a cloud-based or on-premise infrastructure. [27]

***FIWARE***

FIWARE is one of the popular platforms for IoT solutions in agriculture. It is an open-source project that provides a set of APIs and components for developing smart applications. It is based on Java and other open-source technologies and provides a standardized architecture for building IoT applications that can be easily integrated with different sensors and devices, making it suitable for agriculture IoT solutions. It also offers several tools for data management, data visualization, and analytics, which can help in making informed decisions for farming operations. Additionally, FIWARE has a large and active community that provides support and contributes to the development of the platform. It is an open source platform with many available enablers for agriculture due to its novel promising architecture. Its underlying technology is based on Fog Computing that improves the services of Cloud Computing at the edge of the

network. Due to this fact low latency is promised since Fog Nodes are at the proximity of edge devices. Thus it supports low latency and immediate response to actions. Local Fog Nodes near the field can offer their computational and storage resources with low latency and reliability, in order to support heavily automated machinery. Additionally, due to the utilization of Fog computing paradigm the amount of data transferred from the field to the Cloud is reduced because Fog Nodes can participate in computational efforts and accomplish tasks on the field while filtering the results before send them to the Cloud. FIWARE provides a set of tools and components that can be used to build IoT applications. The development process typically involves creating a custom configuration using the FIWARE Generic Enabler Catalogue, writing custom code, and deploying the application to a cloud-based or on-premise infrastructure. FIWARE supports a range of programming languages and deployment options, including Docker containers, Kubernetes, and more. It can be easily scaled horizontally by adding more instances to meet increasing demand. It also provides a container-based architecture using Docker, which enables easy deployment of applications to different environments. FIWARE has been used in large-scale smart city projects such as the SmartSantander project, which involved over 12,000 sensors deployed across the city of Santander in Spain. In addition, FIWARE has been used in the development of the European Data Portal, which serves as a central access point to open data from public administrations across Europe. These projects demonstrate the scalability and reliability of FIWARE in large-scale deployments. [15]

***CHARIOT***

CHARIOT (Cloud Hybrid Architecture for IoT) is a cloud-based IoT platform that provides tools for data collection, storage, analysis, and visualization. CHARIOT also provides several features such as data encryption, access control, and machine learning. It has a middleware for the integration of heterogeneous entities that deals with networking of IoT entities representing devices or services, with support to device heterogeneity while ensuring continuous availability. It consists of a middleware that provides an SDK that facilitates the integration of any type of device, software entities to human actors, and development tools that deal with monitoring, maintenance and control. It offers context- and location-awareness and error-resistant planning for operations by continuously monitoring QoS parameters of services. Through a fog computing architecture is able to decrease latency and enable data processing at the edge without transmitting all the data to the cloud. Under the hood it enables device querying

through their semantic service descriptions which are a way to map the physical devices to their virtual entities. Additionally, the services of the platform may transfer obtained knowledge previously extracted between them, which forms a learning capability. The platform supports different communication protocols such as Wi-Fi, BLE, Zigbee, different data transmission protocols like CoAP, MQTT, REST, SNMP, ModBus, TCP, UDP and different data formats between server and devices like XML and JSON. CHARIOT is mainly a platform for building IoT applications for the industrial sector. The platform provides a range of tools for data collection, analysis, and visualization, as well as support for machine learning and AI. The development process typically involves creating a custom configuration using the CHARIOT Designer, writing custom code, and deploying the application to a cloud-based or on-premise infrastructure. It also supports scalable deployment of applications across multiple nodes and provides a mechanism for dynamically adding or removing nodes to support scalability. [31]

## *State of the art*

### *AgriLoRa*

AgriLoRa is an open-source digital twins-based smart agriculture framework. It is a LoRa-based platform that focuses specifically on precision agriculture. It includes a range of sensors and devices for monitoring soil moisture, temperature, and other key parameters, as well as a cloud-based platform for data storage and analysis. Its codebase is available on GitHub under the Apache License 2.0. The project is built using the LoRaWAN technology and the ThingsBoard platform. AgriLoRa provides tools for crop monitoring, irrigation management, and pest control, among others. In terms of soil analysis, AgriLoRa can integrate with various sensors and devices that measure soil parameters such as temperature, moisture, and pH. The data from these sensors can be transmitted via LoRaWAN to the AgriLoRa platform, where it can be processed and analyzed. AgriLoRa also provides a dashboard where farmers can view real-time data on soil conditions and make adjustments to their irrigation and fertilization schedules accordingly. It also integrates with third-party services such as weather forecasting and soil analysis to provide more accurate insights. Third-party soil analysis tools enable AgriLoRa to provide more detailed information on soil health, nutrient levels, and microbial activity. This integration allows farmers to optimize their use of fertilizers and other inputs, leading to improved crop yields and reduced environmental impact. The platform provides a range of tools

for data collection, analysis, and visualization, as well as support for LoRaWAN networks. The development process typically involves creating a custom configuration using the AgriLoRa Dashboard, writing custom code, and deploying the application to a cloud-based or on-premise infrastructure. [7]

***RIoT***

RIoT (Rural IoT) is an open source platform that aims to provide low-cost, easy-to-use IoT solutions for rural areas, including agriculture. It includes a range of hardware and software components, as well as a cloud-based platform for data storage and analysis. It is focused on providing a modular architecture to support a wide range of IoT applications. It is built on top of the Eclipse IoT framework (Python) and uses several popular open-source technologies such as Apache Kafka, MongoDB, and Docker. RIoT allows users to connect different IoT devices and sensors to collect data and store them in a database for further analysis. It also provides tools for data visualization and real-time monitoring. It provides a mechanism for dynamically adding or removing nodes to support scalability The platform provides a range of tools for data collection, analysis, and visualization, as well as support for machine learning and AI. The development process typically involves creating a custom configuration using the RIoT Console, writing custom code, and deploying the application to the cloud-based RIoT infrastructure. [52]

## *Critical issues and cutting-edge challenges*

Several reviews have remarked in common that although the IoT technologies are promptly evolving and providing considerable novel agricultural applications and services, some critical issues concerning the interoperability as well as the semantic annotation of heterogeneous data have to be handled [26]. Additionally, various challenges related to technological complexity, parameterization, user-friendliness, installation, performance, and system efficiency need to be overcomed [6]. Furthermore, Cloud infrastructure introduces some extra challenges to Cloud based IoT ecosystems like interoperation, trust, and complexity of spontaneous management of cloud and IoT systems, because of their different resources and components [29]. As summarized by [27] and [28], there are some critical issues within IoT platforms and the

corresponding technology ecosystems that need to be progressively addressed to get the full potential of such systems:

- **Complex environment monitoring**: Currently several sensors allow the system to capture a great number of different parameters, which creates a highly dense environment. It is necessary to select the type of sensors to be integrated in our data fusion applications especially when there are physical limitations that in many cases inferred low computing resources and energy autonomy.
- **Modeling**: The IoT environment is characterized by having to gather information from highly heterogeneous devices, technologies and protocols that should be modeled in order to enable processing and data fusion by respecting the constraints of all the technologies involved within the ecosystem.
- **Context awareness**: Due to the great number of sensors that are already deployed it may not be feasible for users to handle all the data collected to the cloud. A way should be used to decide what data needs to be processed. Surrounding environmental parameters and self-assessment may transfer the localized context to others while making a well connected independent periphery aware IoT cloud ecosystem.
- **Standardization**: Many applications utilize their own proprietary formats, which complicates the sharing of data among data acquisition and processing systems [2]. IoT centric applications need to include standardization as a core component which may precisely be operated for their growth. Barriers that create interoperability issues between different applications or systems need to be overcomed. Documentation of industry-specific guidelines and required standards within specifications are important for efficient implementation of IoT.
- **Connectivity**: Currently, devices have the capability to connect through a wide range of technologies, including WiFi, Bluetooth 4.0, 4G LTE, and more. However, this diversity of interfaces poses a technological challenge when it comes to ensuring compatibility and support for these various technologies. Meeting this challenge involves developing optimized network protocols that can effectively handle communication with sensors that have limited resources. Additionally, it requires addressing the scalability of computing architectures to

accommodate the concurrent connection of millions of devices. Moreover, ensuring the permanent access required by these devices further adds to the complexity. Overall, these challenges necessitate innovative solutions to handle the diverse technologies, optimize network protocols, and provide the necessary scalability and availability for seamless connectivity and functionality of a large number of devices.

- **IoT node identity**: All the attached enormous number of devices and data shall be retrievable by a unique identity. New addressing policies should be used, apart from IPv4 that is nearly overflowing.
- **Security**: The security in the IoT ecosystem is a real concern, to avoid leaking sensitive data, or granting access to non-authorized actors to actuate over our environment. It is required to have secure clouds, secure connections, anonymity in the information stored in the cloud, etc.
- **Fault tolerance**: Hardware modules may fail due to depleted battery or any other reason. Similarly generation of erroneous value by the sensor, faulty calibration, and failure in communication may develop a fault situation. A flawless system should keep a very high level of fault tolerance so that despite a technical error it keeps working.
- **Autonomy**: Many data fusion applications require the deployment of distributed battery powered sensors and actuators in a geographic area. Optimizations at different levels of the IoT architecture should be carried out to extend the lifetime of the deployment.
- **Energy Management**: Non-conventional source of energy harvesting solutions such as solar power, wind, biomass, and vibration cloud should be tested while designing IoT based systems.
- **Interoperability**: With so many devices and protocols in use, interoperability is a major challenge. IoT platforms need to be able to connect to a wide range of devices and systems in order to provide a comprehensive view of operations.
- **Data management**: The sheer amount of data generated by IoT devices can be overwhelming. IoT platforms need to be able to store, manage, and process this data efficiently in order to derive meaningful insights.

- **Analytics**: IoT platforms need to be able to provide sophisticated analytics capabilities in order to make sense of the data they collect. This includes real-time analytics as well as more complex machine learning algorithms.
- **Edge computing**: As more devices are connected to the internet, there is a growing need to process data closer to the source. Edge computing involves processing data on the device itself, rather than sending it to a central server, which can reduce latency and improve performance.
- **Sustainability**: The increasing use of IoT devices is leading to concerns about their environmental impact. IoT platforms need to consider the full lifecycle of devices, from manufacturing to disposal, in order to minimize their environmental impact.

## Communication in suburban and rural areas

Within the smart farming domain IoT devices communicate using different types of network connections, wired or wireless. Wired networks, such as CAN and Ethernet, are mainly used for indoor agriculture (e.g., greenhouses), especially due to the high proximity in between them. On the other hand, wireless networking is used both in indoor and outdoor applications. The ubiquitous solution of Wi-Fi protocol is widely used, but its power consumption and signal range characteristics limit its utilization in larger projects or in projects with power restrictions. To overcome such barriers, energy-efficient protocols such as ZigBee and BLE are used to form Wireless Sensor Networks (WSNs) within agriculture, but these protocols are effective only for short distance coverage areas [14]. Also, cellular networks are prevalent in IoT solutions for Smart Farming because they allow communication of devices in long distances and with a high data rate. Nevertheless, Sigfox and LoRaWAN also enable communication in very long distances while requiring low energy to operate and are used as an alternative to cellular networks or in rural and suburban areas where there is no cellular network coverage. [3]

Generally, IoT deployments in rural areas encompass a wide range of scenarios for diverse types of data gathering. Depending on the specific case, data collection can involve sensors with or without latency, transmitting payloads of varying sizes at short or long intervals. However, rural IoT deployments necessitate meeting certain requirements. The primary

considerations for such applications are long range and extensive coverage. The network should function effectively in areas with challenging terrains, where radio communications are inherently difficult, and where telecommunication providers may not offer coverage. Additionally, rural IoT networks often involve a substantial number of end devices, particularly in smart metering applications. Therefore, the communication infrastructure must exhibit excellent scalability to accommodate all devices without compromising the overall network performance. Furthermore, the required lifespan of rural IoT deployments may extend up to 10 to 15 years, or even longer, emphasizing the need for long-term reliability and durability. Replacement of batteries may be infeasible because of the access in environments that are deployed. Therefore, the end devices of technology that will be used for communication should consume minimum energy in order to prolong the lifetime of their batteries. Finally, the total cost of each end device as long as the annual operating cost should be low because such deployments contain a huge number of end devices. Maintenance and installation cost should also be taken into account. The last few years a huge number of LPWAN deployments have been achieved, connecting millions of end devices into several private or public networks. When it comes to adopting a technology solution and designing an application, one important choice is to take advantage of the technology characteristics and get rid of any constraints. It is stated that LPWANs can handle a huge number of devices and have the ability of functioning for several years with zero touch operation. These statements applied from LoRaWAN to NB-IOT. When we dive into these technologies we are realizing that each one has its own cons and pros, but also several technical differences.

### *LoRaWAN*

LoRaWAN is a network stack based on the LoRa physical layer and sometimes is touted as the connectivity enabler for any Internet of Things (IoT) use case. It was firstly introduced by Cycleo (France, Grenble) in 2009 and was acquired by Semtech in 2012. The standardization took place in 2015 by LoRa-Alliance. Currently deployments exist in more than 40 countries while it continues to be developed in other countries due to many network operators investments. LoRaWAN features include low power operation (lifetime of battery is over 10 years), low data

rate varying between 300 bps and 50 kbps and long channel communication range (more than 10km in suburban areas). When we are talking about LoRaWAN we are referring to two separated distinct components: the LoRa physical layer that uses a Chirp Spread Spectrum (CSS) modulation on the bottom and the ALOHA-based MAC layer protocol (LoRaWAN) on the top [40]. There are also other communication protocols that work on top of the LoRa physical layer like Symphony LinkTM and LoRaBlink [45].

***LoRa Physical Layer***

LoRa uses Sub-GHz Industrial, Scientific and Medical (ISM) bands while its wide band nature allows better handling of low Signal to Noise Ratio (SNR). This means that it can demodulate signals at even 19.5 DB below the noise floor, enabling very long communication distances [41]. LoRa uses a chirp (Compressed High Intensity Radar Pulse) spread spectrum (CSS), with integrated Forward Error Correction (FEC) [41], for modulation of wideband linear frequency pulses that continuously vary in frequency (increases or decreases based on the encoded information). This feature offers substantial increase in receiver sensitivity due to the processing gain of the spread spectrum technique and a high tolerance to frequency misalignment between receiver and transmitter [39]. Based on analysis and evaluations, it looks like, LoRa physical layer, thanks to the Chirp Spread Spectrum (CSS) modulation and high receiver sensitivity [40], offers resilience and robustness against interference [37], while also allows longer communication ranges than Frequency-Shift Keying (FSK) without increasing the power consumption that derives to a great communication link budget [41]. Simultaneously, it is resistant to multipath propagation fading and Doppler effect [37] [38]. LoRa is a patent protected technology and thus the evidence on its workflow is mostly based on researchers that reverse engineered the modulation over SDR-based platforms [41]. The modulation can be customized by several parameters including Bandwidth (BW), Spreading Factor (SF) and Code Rate (CR) [39]. In LoRa the large channel bandwidths achieve higher data rates but experience more noise, limiting the range [44]. The minimum number of mandatory channels is three [41] while in Europe the upper limits for channels is 10. The spectrum access is regulated by duty cycle restrictions while there are no channel dwell time limitations [42]. The maximum duty cycle, defined as the maximum percentage of time during which an end device can occupy a channel, and it is a key constraint for networks operating in unlicensed bands. Therefore, pseudo-random channel hopping selection implementation and be compliant with the maximum duty cycle, is

required at each transmission [35]. Because LoRa assumes an unslotted ALOHA protocol and implements Random access capacity, it does not depend on which end device is transmitting, but on its packet duration, which is transmitted unscheduled in random time and channel [48], while there is no collision avoiding mechanism [39]. Each frame is transmitted with a specific spreading factor (SF) which is a trade-off parameter between transmission duration and data rate with communication range [35]. The nominal packet format consisted of a preamble block for synchronization, a physical layer header and the payload. Optionally the header and the payload including a Cyclic Redundancy Check (CRC). Additionally, Forward Error Correction (FEC) is used to allow the recovery from transmission errors due to bursts of interference, that is adding some more encoding overhead. Extra data overhead is added when low data rate optimization is enabled, which reduces the impact on transmission due to drift in the reference frequency of the oscillator. For detecting the channel activity, the Carrier Activity Detection (CAD) is adopted, which is faster than Received Signal Strength Indicator (RSSI) identification [46]. The maximum transmission power is 14 dBm (25 mW), with a link budget of 154 dB and a data rate of 0.3 kbit/s up to 50kbit/s [38].

***LoRaWAN Protocol***

LoRaWAN is a well-established LPWAN technology which is on the top of choices because it enables simple deployments that support large scale networks without involving operators. The architecture implements a star-of-stars topology, in which the end devices communicate directly to a base station (gateway), who works as a relay and forwards messages transparently through an Internet backbone to a network server. [43] Device to device communication is not enabled to achieve a single wireless hop, but instead the end devices transmit packets to the network server through the gateway [40]. Communication is bidirectional, although uplink communication from end devices to the network server is strongly favored [35]. Each end device (ED) transmits data that is received by multiple gateways [2]. In some cases it is noted that gateways received packets from thousands of end devices that are deployed kilometers away, which is a performance target for these base stations. [35] Each gateway is connected to the back-end system, which includes the network server (NS), via ethernet, cellular, WiFi etc. Each NS can be connected to many gateways. The gateways send the received data to the network server, thus, an end-device is associated with a NS, which is responsible for detecting and discard duplicate packets, checks data integrity, choosing the appropriate gateway to send a

reply (if any), sending back packets to the end-devices and identifies the corresponding application server (AS) and transmits it the decrypted data [40]. The application server decrypts the message completely, interprets the payload and makes the data available to the user [38]. LoRaWAN messages are encrypted with two keys as can be seen in Fig. 1. One for the MAC commands and application payload (NWkSKey), and one for end-to-end encryption of the application payload (AppSKey). The NWSKey is only known by the network to prove data-integrity and the AppSKey is distributed to the application-server, for decrypting the application payload [39]. LoRaWAN provides three different classes for end devices (A, B and C). This is an additional way it provides a balance between energy consumption and performance. Class A devices utilize pure ALOHA access for the uplink communication. Following the transmission of a frame, a Class A device listens for a response during two designated downlink receive windows. Each receive window is characterized by its duration, offset time, and data rate. Although the offset time can be customized, the recommended values for the two receive windows are 1 second and 2 seconds, respectively. Notably, downlink transmission is only permitted following a successful uplink transmission. The data rate employed in the first downlink window is calculated based on the uplink data rate and the receive window offset. However, in the second window, the data rate is fixed at the minimum value of 0.3 kb/s. Consequently, downlink traffic cannot be transmitted until the gateway decodes a successful uplink transmission. It's important to note that the second receive window is disabled when the end device receives downlink traffic in the first window. [35]. Class A is the class of LoRaWAN devices with the lowest power consumption because after receiving windows, it goes to sleep in order to conserve energy [45]. Class B devices are specifically designed to cater to applications with higher downlink traffic requirements. These devices operate in synchronization with periodic beacons transmitted by the gateway. These beacons enable the scheduling of additional receive windows for downlink traffic, even without the requirement of prior successful uplink transmissions. However, it's important to note that a trade-off exists between the amount of downlink traffic and the power consumption of Class B devices. Increased downlink traffic may lead to higher power consumption, which needs to be carefully managed to ensure optimal device performance and battery life. [35]. Last, class C devices, as they are usually not battery-powered, can afford to continuously have their radio in receive mode (as long as they are not transmitting themselves), allowing for instantaneous transmission of data towards a device

without having to wait for a receive window to open [45]. The three classes can coexist in the same network, and devices can switch from one class to another, but there is no defined message that informs the gateway about the class of a device [35]. An ED, independently on its class, can be activated over the air called Over the Air Activation or by Personalization called Activation By Personalization. Once an ED is activated, it joins a LoRaWAN network and can communicate with the NS [36]. The long range with low consumption orientation is achieved with the LoRa Modulation technology in conjunction with the Adaptive Data Rate (ADR) mechanism. The logic behind the ADR system is the reduction of the energy that will be consumed to transmit a packet. This mechanism continuously tries to use the highest possible data rate and lowest SF, to reduce the Time-on-Air of each packet. This has an additional benefit of increasing network capacity, as messages sent with different SFs are orthogonal and can thus be received simultaneously [47]. It is concluded that the 10-year operational goal of the specification could be met if care is taken on using proper payload size, transmission interval and the lowest possible spreading factor. The maximum payload size that is supported is 243 bytes, while the uplinks and downlinks that could be transmitted in a day are unlimited [36].

### *Comparison with NB-IoT*

NB-IoT (Aka LTE Cat NB1) is a 3GPP standard published in 2016 and based on narrow-band radio. NB-IoT reduces LTE protocol functionalities to the minimum based on the needs of IoT applications. [49] It is a cellular based protocol and in most cases requires an operator renting licensed frequency bands (e.g., 700MHz, 800 MHz, and 900 MHz) and providing infrastructure (e.g. Vodafone NB-IoT network). [49] The technology supports up to 100 thousands end devices per cell by taking advantage of existing cellular network operators and also offers the potential for scaling up the capacity by adding more NB-IoT carriers. It provides 8-10 years of battery lifetime (when transmitting 200 bytes per day on average), long range and high network security. The communication data rate is 200 kbps for uplinks and 20 kbps for downlinks. The maximum payload size for each message is 1600 bytes. NB-IoT uses for uplinks the single-carrier frequency division multiple access (FDMA) and for downlinks the orthogonal FDMA(OFDMA), while employing the quadrature phase-shift keying modulation (QPSK). It occupies a frequency band width of 200 KHz, which corresponds to one resource block in GSM and LTE transmission. It can function in two operations, stand-alone in which the

utilization of GSM frequencies bands currently used is possible and guard-band in which utilize the unused resource blocks of an LTE carrier's guard band. [36] NB-IoT core network is based on the evolved packet system (EPS) and two optimizations for the cellular internet of things (CIoT). The architecture complexity is high due to the cellular technology, the evolved UMTS terrestrial radio access network (E-UTRAN) is responsible for handling the radio communications while the network consists of evolved base stations called eNodeBor (eNB). The overall data transmitted to the packet data network gateway (PGW) via serving gateway (SGW). The protocol structure is divided into control plane and user plane. The radio resource control (RRC) layer minimizes signaling by suspend/resume operation of the user plane. [50] The LTE radio resource control (RRC) protocol has only two states, the RRCconnected and RRC idle. During RRCconnected, the UE can access the network and request communication resources [49]. Non-access stratum (NAS) of the protocol conveys non-radio signals between UE (user equipment) and core network. The NAS performs security control, authentication and mobility/bearer management. Access stratum (AS) is one layer below NAS and functions between UE and radio network. There is also a random access channel (RACH) procedure which is always contention based and starts with the transmission of a preamble. In the case that preamble transmission fails, the UE will retransmit. [50] NB-IoT technology can be seen as a new air interface from the protocol stack point of view, while being built on the well-established LTE infrastructure, but is not known yet what an adopter can expect in the long term from this technology. [49] In general terms we see that LoRaWAN is advantageous in terms of battery lifetime, capacity, and cost, while NB-IoT offers benefits in terms of latency and quality of service. [36] As LoRaWAN is the major actor in IoT, the goal is to compare deeper NB-IoT to it. [49]

***Power consumption***

In NB-IoT energy consumption is more or less unpredictable. This happens due to the inability to control inactivity timer of the UE, and at the same time the restrictions on adjustment of the allocated bandwidth and path loss through the API. Performance observations showed that mainly these characteristics are linked with consumption peaks on transmissions that approximately range from 100 to 220 mA, when the average peaks in transmission of LoRaWAN's radio are around 40 mA. In these terms the average expected battery lifetime for an NB-IoT end device is about 2–3 years. These values are comparable to LoRaWAN, with SF12,

which is the lowest data rate selection, sending an average of 64 bytes per hour in messages of 51 bytes. On the other hand, NB-IoT is able to send large messages (up to 512 bytes) which has almost no impact. LoRaWAN performs better with short payloads, when more than one message per data block is required there is a great penalty. [49]

***Deployment***

LoRaWAN system has a major difference in outdoor IoT applications from any cellular technology. Star topology makes the architecture simple, low cost and easily maintainable. In NB-IoT, which is based on cellular network structure, the complexity of the devices behavior increases, which leads to unpredictability. Also, because LoRaWAN, which works over ISM bands, allows the user to reduce energy consumption of the devices by deploying a gateway closer to them, which is impossible for Nb-IoT because it is based on private operator infrastructure, which also has limitations on application development of user equipment where the API only exposes a subset of operating points. [49]

***Quality of service***

It is noted that unnecessary interference can be eliminated by using NB-IoT which accesses a licensed spectrum. But, when using cellular licensed spectrum the reprocess of non-orthogonal multiple access method can create hindrance due to restricted pilot assignment in cellular band [4]. While LoRaWAN can bounce interference, multipath and fading, an NB-IoT network guarantees the delivery. This is an important aspect because LoRaWAN can incur significant energy costs for guaranteed delivery, while it is also limited by duty cycle regulations because of the ISM band. [49]

***Coverage and range***

NB-IoT is limited only to suburban or rural regions that benefit from LTE coverage. [36] In this case NB-IoT provides better network capacity for large packets while also providing deep indoor coverage. [49] LoRaWAN with the use of the highest spreading factors (SF11 and SF12), which implies a wide coverage that can reach an entire city with only three gateways. While the range of LoRaWAN is on average 20 km, NB-IoT only offers 10 km range. [36]

***Cost***

The proprietary spectrum used by NB-IoT NB-IoT is offered as a connectivity service under a contract that mainly charges for each transmitted byte. In LoRaWAN the infrastructure is owned by the user who needs to acquire this but the use of it is free. Also the end device equipment is cheaper in for LoRaWAN than NB-IoT. [36]

***Performance***

NB-IoT by default offers a greater scalability, while LoRaWAN is based on the gateway deployment and setup to scale without side effects and increase capacity. [35] LoRaWAN scalability also depends on payload size and data rate. It is designed to transmit a few bytes per hour, even per day. NB-IoT on the other hand can handle larger messages with very low latency but with high variability in delivery time. [49] LoRaWAN confronts an open challenge regarding the duty-cycle regulations in the ISM bands that arise as a key limiting factor which may affect the actual capacity of large-scale deployments. [35]

***Comparison conclusion***

As a conclusion, LoRaWAN seems more appropriate for several types of rural IoT deployments, due to its long range and low power consumption and also the fact that it works on an unlicensed radio spectrum, meaning that no costs are associated. More specifically offers the ability to reduce device cost, increase battery lifetime on devices, improve network capacity and support a large number of devices. NB-IoT provides the best scalability compared to LoRaWAN and thus is able to support a huge number of devices with a low packet error rate. While in terms of latency NB-IoT is the best choice it uses synchronous communication and is a cellular technology. This consumes additional energy and thus decreases end devices' battery lifetime. LoRaWAN offers a great deal of flexibility due to its support for multiple spreading factors (SFs) and device classes while the number of LoRaWAN network deployments is increasing continuously. NB-IoT will serve the higher-value IoT markets that are willing to pay more for very low latency and high quality of service. In our point of view, LoRaWAN is a technology that has been designed from scratch to handle the traffic generated by IoT applications and meets their requirements, on the other hand NB-IoT is based on cells, which is a pre-existing technology that partially serves different aspects of communication.

Based on the aforementioned comparison, we define LoRAWAN as the main protocol to utilize. It differentiates from its competitor especially because it offers a great deal of flexibility due to its support for multiple device classes and spreading factors, while claims that a single gateway can successfully receive data from thousands of nodes that transmit from kilometers away within the free ISM spectrum. License free spectrum utilization is an important choice because when opting for a network that requires pay-per-use, like NB-IoT, for a huge number of nodes a considerable operating cost is revealed.

### *Assessment of crucial factors that affect LoRaWAN performance*

In order to implement LoRaWAN solutions it is crucial for us to analyze the constraints and barriers of the communication technology that our IoT platform will be based on. We have conducted an assessment of some factors that are crucial for any deployment based on LoRaWAN. In this way we have extracted some key directions for designing our platform.

As LoRaWAN is using the license free ISM bands, one can think of that as a huge advantage. If we analyze it from a scope of fixed costs, it is. On the other hand, when we are planning massive capacity IoT networks in suburban and rural areas within Europe we are also facing problems that arise from the ISM spectrum access regulations. The duty cycle limitations unrevealed a lot of issues that are common for the majority of large scale deployments. To overcome this issue the obvious answer is to set communication parameters in a way to achieve balance between capacity and coverage without reaching the duty cycle limits. Some studies have proven that in this step of the deployment several other problems could occur related to collisions, interference and power consumption of the EDs. As we mentioned before, both ADR mechanism and bidirectional traffic issues are pointing somehow to these spectrum access regulations as a factor that exponentially affects the network performance in a non recovery way. ADR mechanism convergence time is increasing as Data Rate (DR) is increasing. Convergence time indirectly affects the time-on-air (ToA), which is increased directly when SF or payload size increases. Additionally, we have seen that bidirectional traffic causes issues on scalability, especially because downlinks in Europe are transmitted with SF12 which results in long transmission ToA. The impact is that as the number of EDs asking for ACKs increased, the

successfully transmitted downlinks decreased, while as the number of Retransmissions of EDs decreased also the acknowledged uplinks decreased.

The main issue is that all variables between uplinks that got acknowledged, successfully transmitted downlinks and convergence time, are indirectly negatively affected by the increase of the number of deployed EDs, which drives us to the conclusion that every crucial factor tends to reflect on scalability. Simultaneously, frequent and significant level of external interference is also argued as causing issues that lead to the limitation of LoRaWAN deployments capacity. This is a reasoning that forecomes the optimal planning of the network itself even if scalability issues have been solved and drives us to the restriction of the total network capacity.

In depth, there are several basic factor relations that are known. First of all in all radio communications PDR is clearly related to the range but in LoRaWAN more specifically, and based on the literature, PDR is also decreased proportional to the amount of EDs (EDs) that are communicating with a GW (GW) and while simultaneously Spreading Factors (SFs) or Acknowledgements (ACKs) or Retransmissions or Payload size, getting increased too. Of course, we have to mention that based on previous studies, deploying more GWs can overcome these casualties but also causing some scalability issues when more than one GW is accessible by an ED. Furthermore, if the ToA of a transmission between an ED and a GW increases, then the power consumption will also increase. Apart from ToA, power consumption of EDs is related inversely to the interval between transmissions and to the DR, and of course is directly affected from the TX parameter of the ED which controls the transmission power. If TX is decreased, the range of the communication is also decreased, which is also happening if the SF is decreased. Finally, capacity is by the book increased if the channels to be used for the communication also increased, which indirectly leads to more capacity on the long ED clusters, that implies better coverage. [55]

### *Safe pathway for LoRaWAN network deployment*

It is proved that collisions (interference between different transmissions within the same network channel) has the most significant negative impact on PDR. We have explored numerous possibilities and we have seen conclusively that Time-on-air (ToA) is directly related to collisions. Specifically, as bigger is the SF, the payload size and the distance, that longer a transmission would last. Further, we can say that as much time a spectrum channel is occupied,

that much is the possibility of collisions. This means that we don't care about the ToA of a payload transmission itself but the sum of its repetitions, acknowledgments or retries of all EDs that occurred within the same SF. Thus, we can say that the most crucial factors that affect LoRaWAN performance are those that affect the channel occupancy exponentially. Based on the observations we determine that those are the number of EDs and the transmission interval, with the respective cruciality. Though, the more direct aspect of the problem is not the overall number of EDs but the number of them within each SF, especially as the SF gets higher. It is clear that SF7 can handle a tremendous number of EDs itself but cannot work on its own if we want wider coverage. Finally we see that it is far better to use more GWs to avoid the usage of SF12 completely, which causes a huge number of problems for realistic data acquisition frequency requirements, especially when ACKs are active because in Europe most deployments use SF12 for RX2. Optimal deployments should aim to include as many EDs as possible (scalability), cover wide range (coverage) and use as less power as possible (power efficiency). In simple words, the parameters for the protocol adoption, is the trade-off between scalability, coverage and power consumption and a parallel goal is to offer LoRaWAN-wide solutions depending on the application requirements, in order to avoid the withdrawal to other LPWAN alternatives (NB-IoT, Sigfox) without reason. It is worth mentioning here that Things Industries proposed and facilitated some great concepts to overcome a lot of the issues arised. Fair access policy but also the ability to configure RX2 to SF9 are by far the most advanced methods to maintain great capacity common infrastructure LoRaWAN networks while respecting the shared ISM spectrum. [55]

# Chapter 4 - Development of an Integrated IoT solution for Agriculture 4.0

The purpose of this thesis is the development of an integrated IoT platform solution which facilitates agriculture 4.0 scenarios and trying to overcome a batch of challenges for smart farming, including system simplicity and scalability, user-friendliness, easy installation, reliability, energy and power optimization, parameterization and system efficiency.

Despite the fact that state of the art platforms at the moment covered our IoT needs, our final desire is to build a fully compatible platform with Agriculture 4.0 scenarios. Thus we

decide to build our solution from scratch due to certain key functionalities of the project and challenges we want to overcome. Our goal is a platform that can deal with data and device diversity while supporting edge processing and dynamic context-based operation profiles for end nodes, by leveraging low energy consumption communication protocols and ultra simple end-to-end deployment.

Our end-to-end IoT solution is built upon a 4-tier architecture that consists of the following layers:

- **Perception layer**: This layer is responsible for collecting data from sensors and other sources, and transmitting it to the network layer for processing. Generally it includes various types of sensors, such as temperature sensors, humidity sensors, and flow sensors, as well as actuators that can control devices based on the data collected. Our information system consists of a design that includes an extreme edge tier with sensors (and actuators) responsible for generating raw data, the Fog tier that consists of ARM-based gateways with sufficient resources to process data to reduce latency and forward them to the Cloud. Also, heterogeneous devices are able to be deployed in run time without an exhaustive installation process (plug-n-play).
- **Network layer**: This layer is responsible for transmitting data from the perception layer to the processing layer, and vice versa. Generally itt includes the communication infrastructure, such as Wi-Fi, Bluetooth, or cellular networks, as well as the protocols used for data transmission. Our platform’s network layer stands on a transport layer based on LoRaWAN, which publishes the edge messages through an MQTT broker.
- **Processing layer**: This layer is responsible for processing the data collected from the perception layer and providing insights and analytics. Generally, it includes various types of data processing technologies, such as big data analytics, machine learning, and artificial intelligence. In our case cloud is responsible for the storage, intelligent management and publication of the data trafficked to and from the end devices on the Internet. The core services are hosted on the Cloud and are subscribed to the MQTT broker messages, while they transform them into meaningful data that getting stored in the database, exposed through a REST API and processed by an agriculture

decision support system (ADSS) for mission planning and water resources management

- **Application layer**: This layer is responsible for providing value-added services and applications that leverage the data collected from the IoT devices. In general it includes various types of applications, such as smart home automation, industrial automation, and healthcare monitoring. In our domain specific solution we are concentrated in services regarding crop management and precision agriculture.

## Perception Layer

### *Hybrid IoT edge node*

Each use case has unique requirements that inferred some different network architecture, from the simpler to the most complex case. For that reason an ARM-based single-board computer (SBC) end node that is able to be adapted in any deployment is the ideal solution for our system. Our novel hybrid IoT node can work as an extreme edge dummy node able to just send and receive data, as edge computing node with capabilities of process, filter and aggregate self-sensed data, or as a Fog gateway able to process, filter and aggregate data of subnetworks edge nodes. The Hybrid IoT node in any condition can support multiple affordable and versatile sensors and/or actuators. The deployment initialization process is only based on a unique network identifier and on a preset of functionality, called Device Group. The latter is a set of commands that are describing templates of grouped physical end devices (e.g. SBCs, sensors, actuators, etc). These are shared configuration schemas, distributed as common assumptions on both Hybrid IoT nodes and the Cloud, consisting of a group structure, as well as tags and relations that describe the workflow orientation of this group. This is a vital way of handling hardware abstractions and form a robust system able to function with the minimum deployment effort, while supporting device diversity.

### *Distributed middleware that supports Heterogeneity*

The Hybrid IoT node is not aware of what it measures, nor of what it transmits. Thus, data transaction between the end nodes and the cloud is agnostic. The contextualization is being made on the cloud, which identifies and correlates the Hybrid IoT node Device Group with its

predefined configuration schema. Device Group schemas mainly include some predefined device addresses (GPIOs, Serial Ports, Subnetwork node IDs) of the connected devices. Additionally, the aforementioned data structures that are represented in memory, should be converted into architecture and language independent formats that are transmitted across the network [32]. This practice is also a way to facilitate the minimum communication message length between cloud and edge and thus utilizing the efficiency of our custom made Data Serialization Format for LPWANs.

### *Application-level Zero-copy Binary Serialization Library*

According to our study for communication performance influencing factors, there is a trade-off triangle between coverage, scalability and energy consumption. Also, there could be issues regarding ISM spectrum access regulations when designing mass-capacity IoT networks in suburban and rural areas within Europe. Thus, IoT end node communication requirements on rural areas dictates small size data exchange through the network, in order to prevail bandwidth capacity and save energy. In order to make targeted performance optimization in our system, we decided to use data serialization techniques in order to minimize payload size. In this way, we managed to achieve a balance between capacity and coverage without reaching duty cycle limits. In the IoT context numeric-based data is much more efficient because numbers constitute the majority of data that is exported from sensors. Consequently, schema-based logic becomes a better fit for us because in that way we avoid the encoding of field names into the payload as an additional string [33]. In this way we achieve smaller message size than schema-less techniques, which is essential for our requirements. Additionally, LPWANs derive the requirement of reducing expensive encoding and decoding steps due to additional power consumption and instead using fewer resources to directly read values contained inside a serialized data structure without deserializing them. For that reason our pointer-based approach follows the principle that data should be structured in the same way as in memory [11] to achieve space efficiency. We have designed and developed a bit-packing process that generates a buffer sequence where each chunk has an interdependent offset in memory from the start of the buffer. To achieve this we have divided the messages in known chunks with a known sequence. Each chunk has a corresponding scheme that defines its length in bits which is exported from the possible values that need to represent (e.g. with 8 bit an integer between 0 and 255 can be expressed which is

fine for addressing devices). Furthermore, scheme values in all cases are integers that are sometimes bit-packed as is, but also could be the index of an array of possible values (e.g. in 4 bits we can have 16 different values which are enough for indexing an array of the message type flags). We are also utilizing some chunks as relays for the existence or size of other chunks which generates interdependencies between message fields. Additionally, our solution supports multiple device reports per buffer that offer an exponential reduction on the message length which is vital to scale the system. The general idea is to have small messages that describe a lot of different devices independently. This means that we need to avoid communicating with end devices based on their identifiers that typically are big strings. For that reason we are using an enumeration for each device that is directly connected to an SBC or for each device from a subnetwork. This enumeration is corresponding to an integer value that is equal to the index of its identifier on the enumeration array. Despite the fact that we need extremely small data sizes, we also need to be temporally consistent, which introduces the requirement for time and/or date annotation within the message. To overcome this data size constraint we have implemented a time synchronization logic. Each payload has a sync timestamp (32 bits) as a custom local epoch declaration while each report or action on payload has the difference of the sensor measurement or action schedule since this custom epoch.

**Table 4.1.** Zero-copy binary serialization global chunks types

| chunk title | size in bits | values | min value | max value | dependent | comments |
|---|---|---|---|---|---|---|
| cmd | 8 | indexOf[0, 1, o, r, a, c] | | | | Flag for the message (0 = init, 1 = ping, o = operation profile, r = report, a = action, c = continue) |
| is_part | 8 | integer | | | | Indicates if the following message is just a part of the whole message |
| epoch | 32 | integer | | | | Epoch timestamp in seconds of the message generation |
| device_index | 8 | integer | 0 | 255 | | The root microcontroler index of a device for the data that follows |
| value_type | 3 | indexOf[m, e, t, p, i] | | | | The value type that follows (m = measurement, e = event, t = trigger, p = percent, i = instruction) |
| value_subvalues_number | 4 | integer | 0 | 15 | | Indicates how many chunks of values follows from the same device |
| value_has_time_diff | 1 | boolean | | | | If the value has a time difference related to the epoch timestamp |
| value_minutes_diff | 16 | integer | 0 | 2043 | value_has_time_diff | The time difference in minutes of the value related to the epoch timestamp |
| long_value_has_fraction | 1 | boolean | | | value_type | If the long value that follows has a decimal part |
| long_value_fraction | 8 | integer | 0 | 99 | long_value_has_fraction | The decimal part of the value |
| long_value_size | 6 | indexOf[8, 16, 24, 32] | | | value_type | The size in bits of the following integer base of the value |
| long_value_sign | 1 | indexOf[-, +] | | | value_type | The sign of the value |

| | | | | | | |
|---|---|---|---|---|---|---|
| long_value_base | 8 \|\| 16 \|\| 24 \|\| 32 | integer | | | value_type, long_value_size | The integer base of the value |
| boolean_value | 8 | boolean | | | value_type | General purpose boolean |
| report_percent_value | 7 | integer | 0 | 100 | value_type | Value in percentage |
| action_trigger_value | 8 | boolean | | | value_type | Boolean for trigger an action (on/off) |
| action_percent_value | 7 | integer | 0 | 100 | value_type | Value in percentage for actions |
| op_io_type | 2 | indexOf[gpio, port, subnetwork] | | | | The type of the physical address that the following device is connected to the root microcontroler |
| op_io_gpio | 6 | integer | 0 | 63 | op_io_type | The gpio that the following device is connected |
| op_io_port | 6 | indexOf[ttyACM0, ttyS0, ttyUSB0, ttyUSB1] | | | op_io_type | The port that the following device is connected |
| op_subnetwork_network_id | 8 | integer | 0 | 255 | op_io_type | The network id of a device that is within a subnetwork |
| op_subnetwork_node_id | 8 | integer | 0 | 255 | op_io_type | The node id of a device that is within a subnetwork |
| op_subnetwork_device_id | 8 | integer | | | op_io_type | The deviced of a device that is within a subnetwork |
| dev_code | 8 | indexOf[d, ir, mir, mth, cl, sn, snrs, rrv, cl_legacy, cl_debug, cl_debug_mini] | | | | The device code of a root microcontroler |

**Table 4.2.** Zero-copy binary serialization message format representing the example of a LoRa node that sends report to the cloud for device with index=0 with value=756.39 and time difference of 5 minutes related to the epoch time in seconds= 1631115456

| cmd | epoch | device_index | value_type | value_subvalues_number | value_has_time_diff | value_minutes_diff | long_value_has_fraction | long_value_fraction | long_value_size | long_value_sign | long_value_base |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 1631115456 | 0 | 0 | 1 | 1 | 5 | 1 | 39 | 1 | 1 | 756 |
| "00000011" | "01100001001110001101100011000000" | "00000000" | "000" | "001" | "1" | "0000000000000101" | "1" | "00100111" | "000001" | "1" | "0000001011110100" |

**Snippet 4.1.** Converting the value to base2 and then create a map of binary chars which is needed for bitwise operations

```
if (chunkScheme.min && parseInt(value) < chunkScheme.min) {
                value = parseInt(chunkScheme.min);
        }
        if (chunkScheme.max && parseInt(value) > chunkScheme.max) {
                value = parseInt(chunkScheme.max);
        }

        let size = 0;
        if (!Array.isArray(chunkScheme.size)) {
        size = chunkScheme.size;
        } else if (Array.isArray(chunkScheme.size)) {       size = value.size;
        }
```

```
if(typeof value.data != 'undefined') {       value = value.data;
}
let binary = parseInt(value).toString(2).split('').map(Number);
if (binary.length > size) {
        binary = Array.from({
length: size
}, (v, i) => 1);
}

let maskedPaddedBinary = self.addZeroPadding(size - binary.length, binary);
```

## Network Layer

### *End devices communication by using LMIC (LoraWAN-MAC-in-C)*

We have chosen the LMIC library to develop our end nodes, which is a useful library to deploy LoRaWAN devices that can communicate with LoRaWAN gateways using the LoRaWAN protocol. LMIC (LoraWAN-MAC-in-C) is an open-source, lightweight library that provides a way to communicate with LoRaWAN gateways using the LoRaWAN protocol. It is designed for use with low-power, low-data-rate wireless networks and is commonly used in IoT applications. The library is written in C and can be used on a range of microcontrollers, including Arduino boards, ESP32, and STM32. LMIC provides a range of functions for sending and receiving data packets, managing device communication with LoRaWAN gateways, and implementing security features such as message encryption. LMIC operates in two modes: ABP (Activation-By-Personalization) and OTAA (Over-The-Air-Activation). In ABP mode, the device is pre-configured with keys and network parameters, and the device can begin communicating with the network immediately. In OTAA mode, the device first performs a join procedure to obtain the necessary security keys and network parameters before it can begin communicating with the network. Our end nodes are developed to function with the OTAA mode because it provides greater security than ABP mode, as it allows for dynamic key exchange and reduces the risk of key compromise. Additionally, it allows for better scalability, as devices can join and leave the network dynamically without requiring manual configuration. The workflow we have followed for implementing OTAA mode using the LMIC library is as following:

- Configure the device parameters: The device parameters that need to be configured include the device EUI, application EUI, and application key. These values are obtained from the The Things Stack network server.
- Initialize the LMIC library: The LMIC library needs to be initialized with the device parameters and LoRaWAN frequency plan. This can be done using the os_init() and lmic_init() functions.
- Start the join procedure: The device sends a Join Request message to the network using the LMIC_startJoining() function.
- Handle the Join Accept message: The network responds with a Join Accept message that contains the necessary security keys and network parameters. The device needs to handle this message and extract the required values using the LMIC_getSessionKeys() function.
- Store the security keys and network parameters: The security keys and network parameters obtained from the Join Accept message need to be stored in the device's non-volatile memory for future use.
- Begin communicating with the network: Once the device has successfully completed the join procedure, it can begin communicating with the network using the LMIC library functions such as LMIC_setTxData2() to send data to the network.

### *Cloud Prerequisites*

For our deployment we have set up a virtual machine to be used as a server with an 64-bit version of Ubuntu Bionic 18.04 (LTS), 4 virtual CPUs and 16GB RAM. Our cloud infrastructure consists of several Docker containers with isolated environments. We have followed the official guide to install Docker Engine on Ubuntu and make the necessary configurations to manage Docker as a non-root user while configuring it to start on boot. Additionally we have used Docker Compose that offers us the ability to use a single YAML file to configure our application’s services. With a single command, we can create and start all the services. Compose is a tool for defining and running multi-container Docker applications.

### *The Things Stack (TTS)*

In our workflow, LoRaWAN devices communicate with LoRaWAN gateways, which forward the data to The Things Stack. The Things Stack processes the data and sends it to the application server, which can then use the data to perform various actions according to our needs.

The Things Stack is a LoRaWAN network server stack, built upon an open source core, developed and maintained by The Things Industries. The Things Stack allows us to build and manage our LoRaWAN networks with our own hardware and in our cloud. The Things Stack includes it's core which is written basically in Go language, along with a Redis server and a Cockroach database.

The Things Stack consists of several components, including:

- Network Server: The Network Server receives and processes data from LoRaWAN gateways, performs device authentication and authorization, and routes data to the correct application.
- Application Server: The Application Server is responsible for managing applications, receiving and processing data from devices, and triggering actions based on device data.
- Join Server: The Join Server is responsible for managing device activation and providing security keys to devices.
- Gateway Server: The Gateway Server manages LoRaWAN gateways and receives data from gateways.
- Identity Server: The Identity Server manages user authentication and authorization.

Developers can leverage The Things Stack in a number of ways, such as setting up a private LoRaWAN network, connecting IoT devices, collecting and processing data, integrating with other cloud-based services and building custom applications by leveraging the APIs and SDKs to manage devices, collect and process data, and trigger actions based on device data.

Overall, The Things Stack provides us a powerful platform for building our IoT solution, enabling us to quickly and easily develop, deploy, and scale our application through The Things Stack Console. The Console is the management application of The Things Stack for LoRaWAN. It is a web application that we are using to register our applications, end devices and gateways,

monitor network traffic, or configure network related options, among other things. Additionally, it does provide a command-line interface (CLI) tool that can be installed locally and used to interact with The Things Stack via the command line. The CLI tool is called "ttn-lw-cli" and can be installed on Windows, Linux, and macOS. It provides a range of commands for managing LoRaWAN networks and devices, such as creating applications, registering devices, and managing network settings.

To initialize The Things Stack (TTS) for LoRaWAN network deployment, we first need to create an account on the TTS website and set up a new application. Then, we need to add your LoRaWAN gateways and devices to the application by specifying their unique identifiers and network parameters. Once the devices are added based on our network design methodology, we can start receiving and sending data between the devices and the TTS network. Additionally, we can use the TTS APIs and integrations to connect our LoRaWAN devices to our IoT platform. Finally, we can use the TTS console to monitor and manage our LoRaWAN network, including viewing device activity, managing network settings, and configuring integrations.

### *LoRaWAN network design methodology*

When we can configure a LoRaWAN network of thousands of devices over a range of several km with the least power consumption, we can define some different application oriented demands named, data acquisition assurance, data acquisition frequency, and data size, which all are different aspects of Quality of service (QoS), as long as a hybrid solution for serving applications based on data-level requirements such as the distinction of data and so in EDs level QoS. The level of data acquisition assurance is not directly connected with ACK and retransmissions, as it could, but to the PDR cut-off which is acceptable for our needs. For example this cut-off could be 0.50 if our application is not precise centric and we don’t care if we lose one out of two messages. [55]

The clustering method we have followed can be described as:

1. Count the overall EDs that needed
2. Deploy as many GW are needed to avoid the DR0 allocation that could be necessary for far deployed nodes.

3. Separate the EDs in several clusters based on the distance from the closer gateway. The clusters should be in donut shape with the inner cycle being the shortest distance and the outer being the furthest.
4. If there is not an application wide QoS level, further distribute the EDs in each distance cluster to several new clusters based on QoS aspects (data acquisition assurance, data acquisition frequency and data size).
5. Distribute the EDs for each of the previous clusters to DR-TX groups based on the following criteria:
    a. As bigger is the distance and data acquisition frequency that bigger is the DR that should be allocated, which are proportional volumes to the acknowledgment and amount of retransmissions.
    b. As high is the DR that high should be the TX power to simultaneously achieve the best trade-off between coverage and power consumption.
    c. As more EDs are in the same cluster, the higher the DR allocation should be.
6. Exhaust the distance limits of each DR cluster in order to use as high DRs as possible, while DR5 cluster should configured to around 64% of the total EDs [51]
7. Lint the distance based clusters to equalise as possible the maintenance intervals, or distribute final clusters in power needs clusters to determine which set of power capacity configuration should be used in each case (if this is a possible capability for the deployment).

### *Interoperability and MQTT Broker*

To achieve interoperability between the cloud core application and TTS we can either set up Webhooks or an MQTT broker. Webhooks, which is a technology based on Hypertext Transfer Protocol (HTTP), is a way that applications can send automated messages to other apps. When events like getting uplink payloads occur, TTS makes an HTTP request to the URL configured for the webhook. Nevertheless, our choice is to use the MQTT protocol instead, because it uses a reduced overhead related to HTTP. This feature offers better energy efficiency which is critical in IoT applications where the number of connected IoT devices increases

considerably [30]. RabbitMQ, which has been selected as a broker component, is an open source message broker that uses a binary application layer protocol, designed to efficiently support a wide variety of messaging applications and communication patterns. It facilitates the development of distributed, fault-tolerant and asynchronous applications. From a security point of view, RabbitMQ uses Simple Authentication and Security Layer (SASL) for authentication and data security in Internet protocols. RabbitMQ MQTT offers a wide range of MQTT clients and adds the possibility to interoperate with AMQP and STOMP clients. RabbitMQ operates on top of two core protocol entities. These two entities are called exchanges and queues. Messages published to MQTT topics use a topic exchange internally. Subscribers consume from RabbitMQ queues bound to the topic exchange. These features allow interoperability with different protocols and make it feasible to use a unique management module to examine queue sizes or message rates. [30]

## Processing layer

### *Cloud core application*

On the Cloud core application, all IoT devices are distributed to virtual subnets that are handled by functionally independent resource managers. Each running resource manager is the root of the application layer of an IoT subsystem with individualized configuration, operational settings, and spatial data, while corresponding to a specific stakeholder (user or operator) on the business logic. Additionally, resource managers are using a decoupled service to load data from the DB, including the device map which includes the end devices of the IoT network and the operation scenario which in turn is a specification containing end user preferences and/or decision models. The core application service includes the physical capabilities of the edge devices as digital representations at the software level. This creates a virtual ecosystem between end users and remote end devices (IoT). In this way the workflow and the relationships between the end devices and their data are integrated into the cloud. Each edge node consists of three virtual objects that define what essentially constitutes the context of the node. The Microcontroller object is an emulation of a physical SBC placed at a spatial point that controls the software it runs, the communication interface, any sensors or actuators that are connected, and any subnets. The Communication Interface object represents the physical communication interface (modem) of each point that is responsible for communicating with the Cloud through

the network layer of LoRaWAN. The Device object is reflected in an end device receiving data or implementing actions and operating based on the device model specification and the instructions given by the Cloud to the Microcontroller that controls it. This end device can be a sensor, actuator or a subnet gateway (WSN gateway) forwarding data to a possible WSN sensor network. Each type of sensor or actuator corresponds to a Device Specification schema. It consists of some rules and options related to the capabilities of the physical devices, in order to facilitate the easiness and adaptiveness of adaptation and deployment of the system. In this way, there is no need to make custom settings by a qualified technician for each device. Instead the user can include devices from a range of predefined device properties for which the system has all the necessary information to start them up, adjust their operation, communicate with them and read their data. The Cloud composes a profile with the operating instructions of the edge nodes, which are thus not responsible for knowing exactly what to do. With this method all SBC execute a basic core of commands that are shipped with the device model specification, combined with the operating profile received from the Cloud. The operation composition service includes operation profiling based on system and device specification schemas, system-wide operation settings and manager-wide operation scenarios. Specifically, operation profiles are workflow descriptions for edge nodes and the data they consist of are sensors and actuators enumerations and parameters for performing tasks. Also includes the ID of their serial connection, and especially for sensors, the measurement frequency as well as the uplink frequency. In such a way we can change our device behaviour dynamically, like changing the reporting intervals or control the workflow of complicated edge node modules such as edge processing nodes or even agents such as unmanned aerial vehicles (UAV) or unmanned ground vehicles (UGV). Finally, the Cloud also includes a decision support service (DSS) which operates based on the edge node data and the manager-wide data processing models that are defined. This system produces commands for alerting and actuating on the edge. Alerts could be generated to notify the end user or to feed other services within the cloud (e.g. logging), while actions commands are triggering events on the edge devices. Our cloud core application consists of a Node.js Express Docker container as the server framework / controller layer, one MongoDB Docker container as the database layer and an NGINX server Docker container as a proxy / content-caching layer. Its installation is straightforward in contradiction with The Things Stack. The application basically consisted of models, classes, services, handlers and routers. Models are

Mongoose schema definitions and their associated models, while classes define these models in an application oriented way. Services contain the main application logic while handlers and routers utilize Express.js modules to expose the RESTful API.

### *Application Classes*

The cloud core application consists of several services to function. The most important of them are listed and briefly described below:

- **Orchestrator**: Orchestrate the cloud by utilizing individual services/modules in order to emulate network devices, manage the data they produce or receive, and implement the communication protocol. This service can simultaneously support the management of many different IoT deviceσ and data management subsystems of different operators that operating with their custom setup and management models. The purpose of this orchestration is the processing of the data to be collected that leads to decision making.
- **Resource Manager**: Resource managers are operation independent systems that support the management of an IoT network of devices in a cloud way. This is the root instance, corresponding to an operator user role, that consisted of service class instances, specification maps (system configuration, operation setup, device map, recipe) and spatial data. The resource manager additionally uses a decoupled service to load database models (Model Accessor), including the device map which includes the end devices of the IoT network and the operation scenario which is a specification map containing end user preferences and/or decision models.
- **Device Controller**: The device manager is the core service of the application because it integrates the end devices in the field including their physical capabilities at the software level creating a virtual ecosystem between end users and remote end devices (IoT). In this way the workflow and the relationships between the end devices are integrated into the computing cloud. Also, the device manager takes over all the communication processes and acts as a data input / output router to and from the end devices by exchanging messages with the LoRaWAN server.

- **Network Gateway**: Service for handling the interoperability between the application core of the cloud and The Things Stack (TTS) which is the LoRaWAN server.
- **Data Handler**: Service that interacts with database storage. Handles the modification of the data coming from the communication with the field and stores or serves it.
- **Operation Composer**: Service that configures the operating instructions of the end devices, which are thus not responsible for knowing exactly what to do. Operation synthesis involves building operation profiles based on system and device specification maps, operation settings, and operation scenarios.
- **Decision Processor**: A decision support system (DSS) that operates based on input data and management models defined for each resource manager.

Additionally there are several object classes that used to accommodate the data management on the cloud. Several of them are listed below:

- **Microcontroller**: Emulation of a physical microprocessor (e.g. Raspberry Pi) placed at a spatial point that controls the software it runs, the communication interface, any sensors or actuators that are connected, and any subnets.
- **Communication Interface**: It represents the physical communication interface of each point that is responsible for communicating with the cloud computing through the LoRaWAN network layer or for communicating between some subnet.
- **Sensor**/ **Actuator** - **Device**: It is reflected in an end device receiving data or implementing actions that operates based on the device model specifications and the instructions given by the cloud computing to the microprocessor that drives it.
- **Point**: Represents a geographic point of interest in the cloud. It could be a ground station, a crop, a field, a water collector, a water course, a water resource. The water resource is directly connected to the water routes that are directly connected to the collectors. Collectors must include at least one field which should include one or more different crops which could include ground stations.

- **Uplink/ Downlink - Payload**: Represents an uplink or downlink load of the LoRaWAN server.
- **Measurement**: Ingest data at a single point in time from an end-to-end physical field device.
- **Event**: Event that occurred at a single time from a final physical field device.
- **Action**: It contains an action that must occur at a field end device and is the product of a decision made by the decision processing unit.
- **Management Model**: It contains scientific calculations related to the respective operating scenario of the resource manager and is used in decision making.

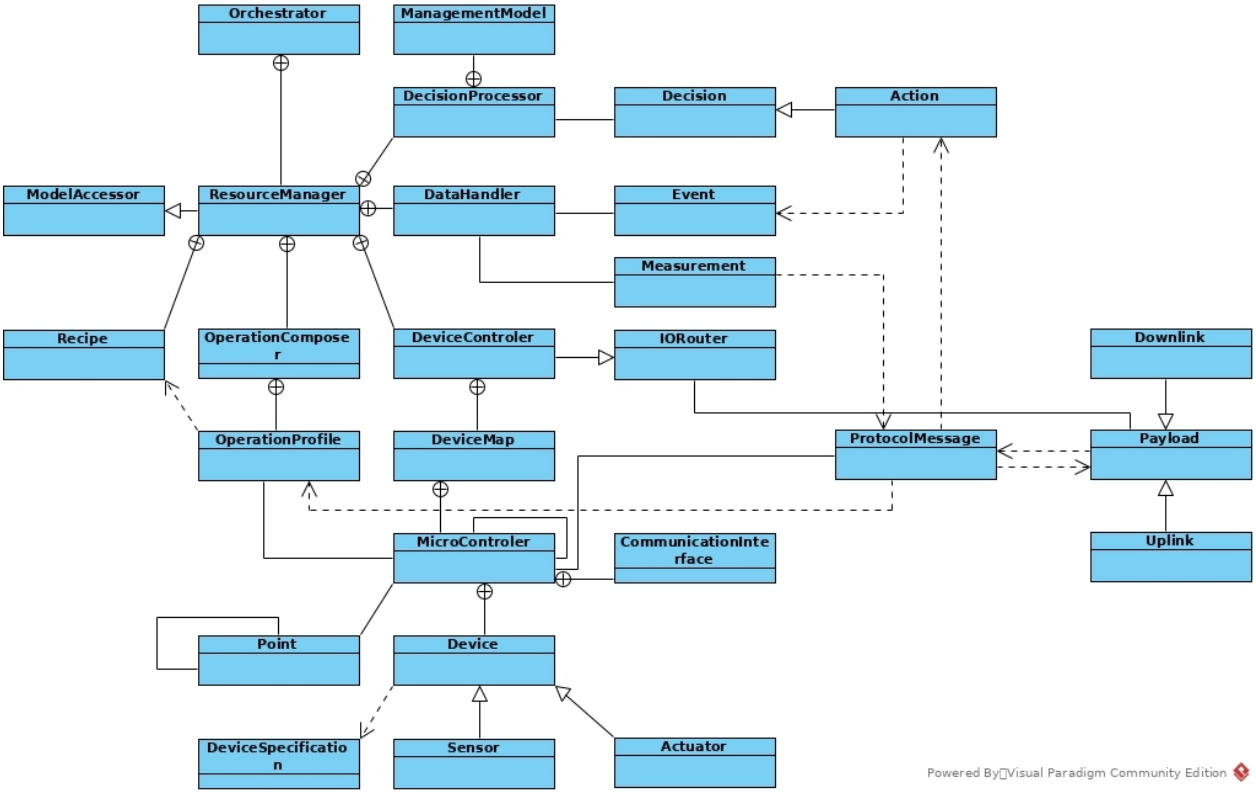


**Figure 4.1.** Cloud application core class diagram

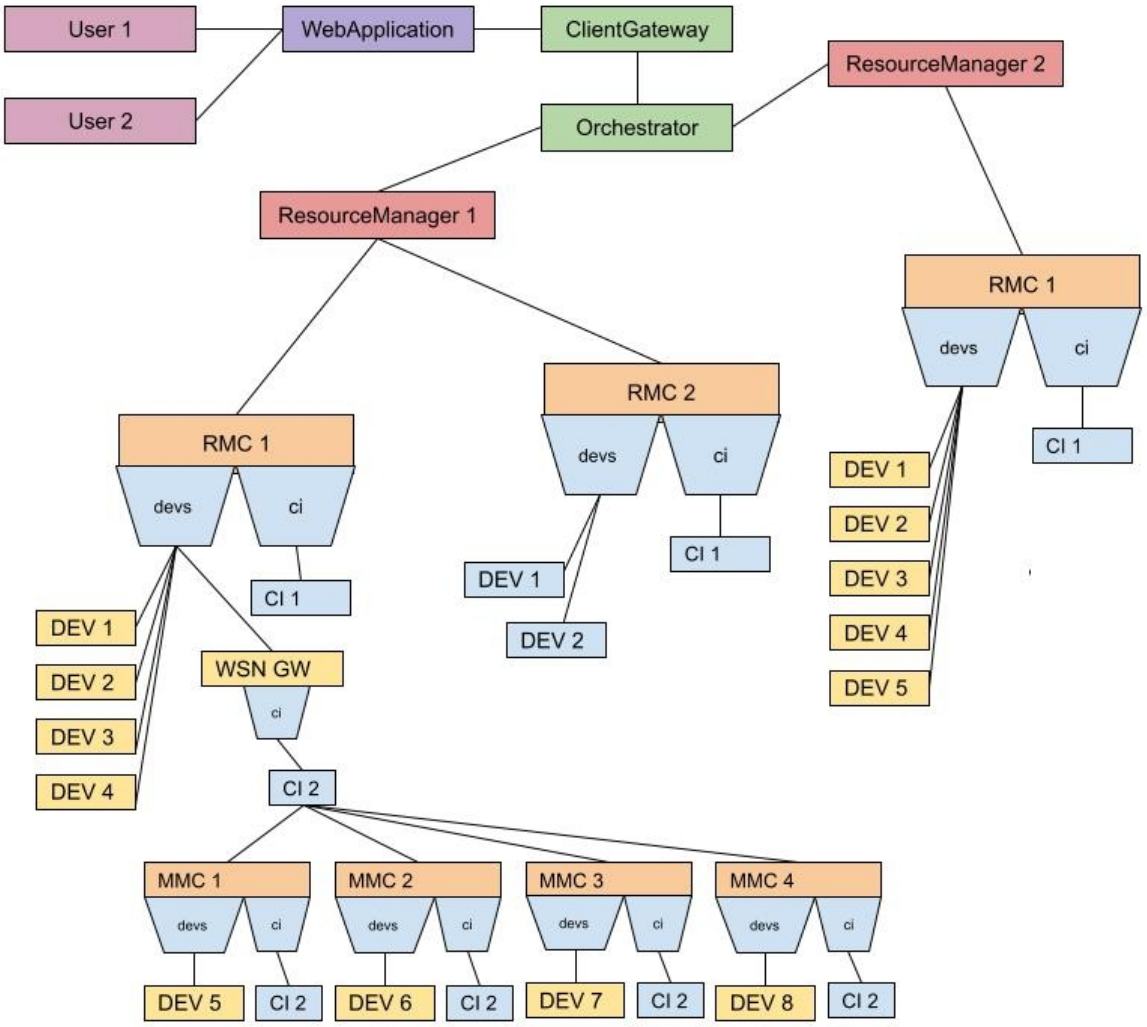


**Figure 4.2.** Diagram represents application models and their relations and inheritance

### *Workflow*

The system is distributed so that it can start working automatically after the end devices are installed and registered in the management interface, which will be developed for this purpose later, in combination with the registration of some preferences according to the operating scenario. Thus, after the aforementioned are registered in the cloud computing database, the orchestrator starts the function composition, device management and data manipulation units (Figure 4.3).

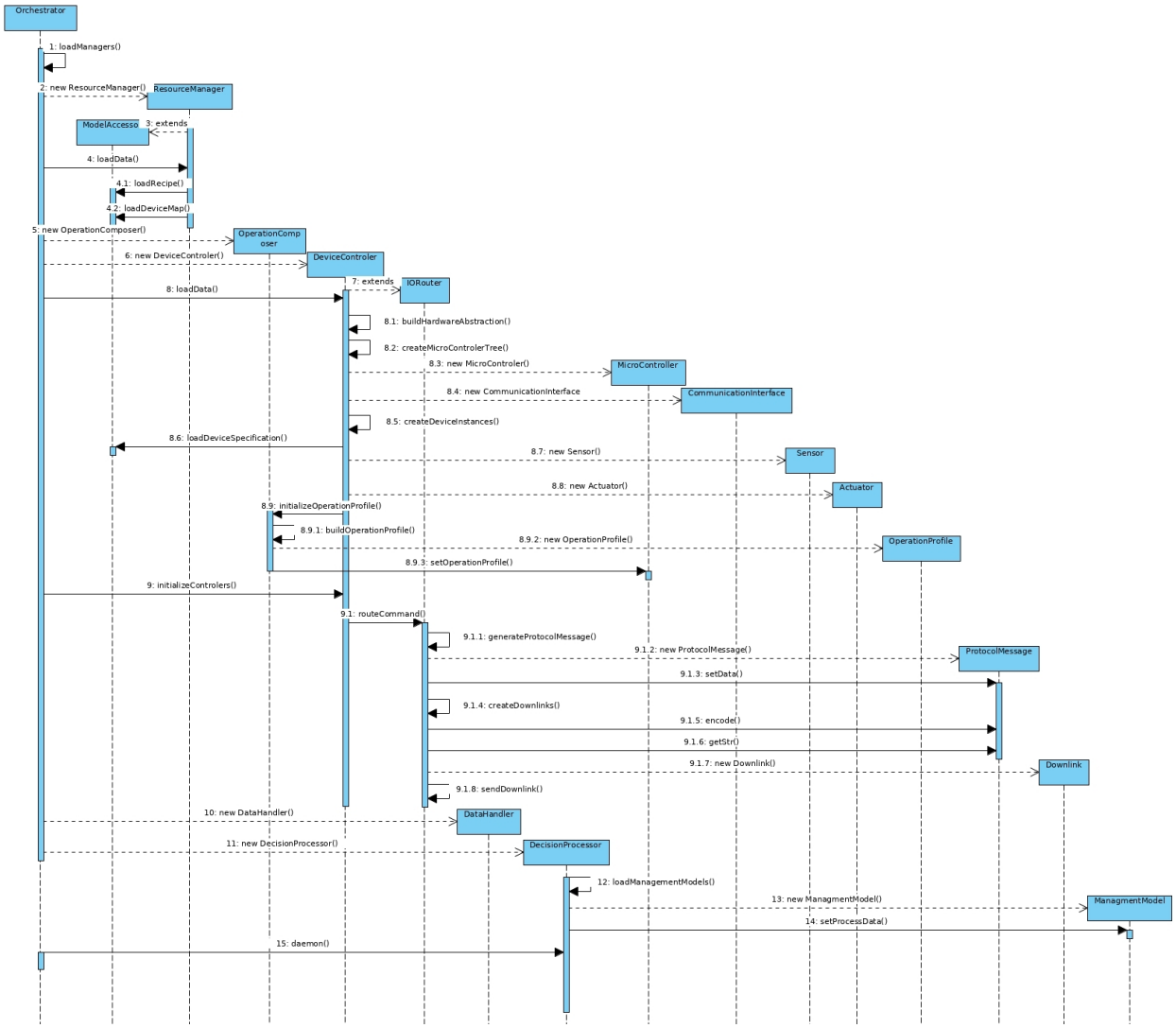


**Figure 4.3.** Initialization flow diagram of cloud computing core application modules

The device manager loads all connected devices (microcontrollers, communication interfaces, sensors, actuators) in tree form into memory from the database, specifying their capabilities from the database. This is done by associating the root microcontroller devices contained in the respective resource manager's device map. (Code Snippet 1) The generated tree is structured by repeating the communication interfaces associated with the microcontroller and the subnet nodes that are also loaded (microcontrollers, communication interfaces and devices).

**Snippet 4.2.** Individually load devices into memory per resource manager

```
await Promise.all(deviceMap.nodes.map(async (node) => {
        const rmcId = node.rmc_id;
        await self.createMicroControlerTree(rmcId).then(async(rmc) => {
                self.map[rmc.id] = rmc;
                return self.map;
        })
}));
```

The operation composition module then creates an operation profile for each microprocessor based on the operation scenario preferences of the respective resource manager. The operation profile is a map of instructions, options, and device mappings that depend on the operation of the physical microcontrollers (Code Snippet 2). Because microcontrollers know nothing more than what happened locally in the field, we must have a way to achieve mapping of sensors and data, as well as actuators and actions. Operation profiles also define the behavior and routing of data managed by the microcontroller. These operating profiles are then via the device management module encoded into messages and forwarded to the end devices.

**Snippet 4.3.** Enumerate microcontroller devices and map them to models in memory

```
Object.keys(mc.devs).forEach((d)=>{
        let dev = mc.devs[d];
        opMap.cis.enum.push(dev.id);  // enumerate from now the root mc connected devices
        let specs = config.specifications;
        opMap.devs[dev.id] = {};
        let specDevType = Object.keys(specs).find((k) => {
                return specs[k] === dev.specification.type;
        });
        let task = mc.devGroup.task;
        opMap.devs[dev.id]['io'] = dev.io;
        opMap.devs[dev.id][dev.role] = setup[task][dev.role][specDevType];
});
```

Following this sequence the application core is ready to receive messages and coordinate actions towards the end devices. When an uplink payload arrives from the LoRaWAN server the application associates the receiver with the resource manager and then associates the data with the corresponding device and stores it in the database (Code Snippet 3). This is achieved by the device manager collecting the payloads and decoding the protocol messages they contain. The

data handling unit then maps each part of the message to the corresponding data type (count, event) and stores them in the database (Figure 4.4).

**Snippet 4.4.** Uplink payload collection from field

```
  const raw = payload.uplink_message.decoded_payload.raw;
          const     ci     =     rms.deviceControler.interfaces.find((i)     =>     i.id     ==
payload.end_device_ids.device_id);
  const timestamp = payload.received_at;
  const uplink = rms.deviceControler.gatherUplink(id, raw, timestamp, ci);
  const inputData = rms.dataHandler.gatherReportsFromDevices(uplink, timestamp);
  rms.dataHandler.storeData(inputData);
```

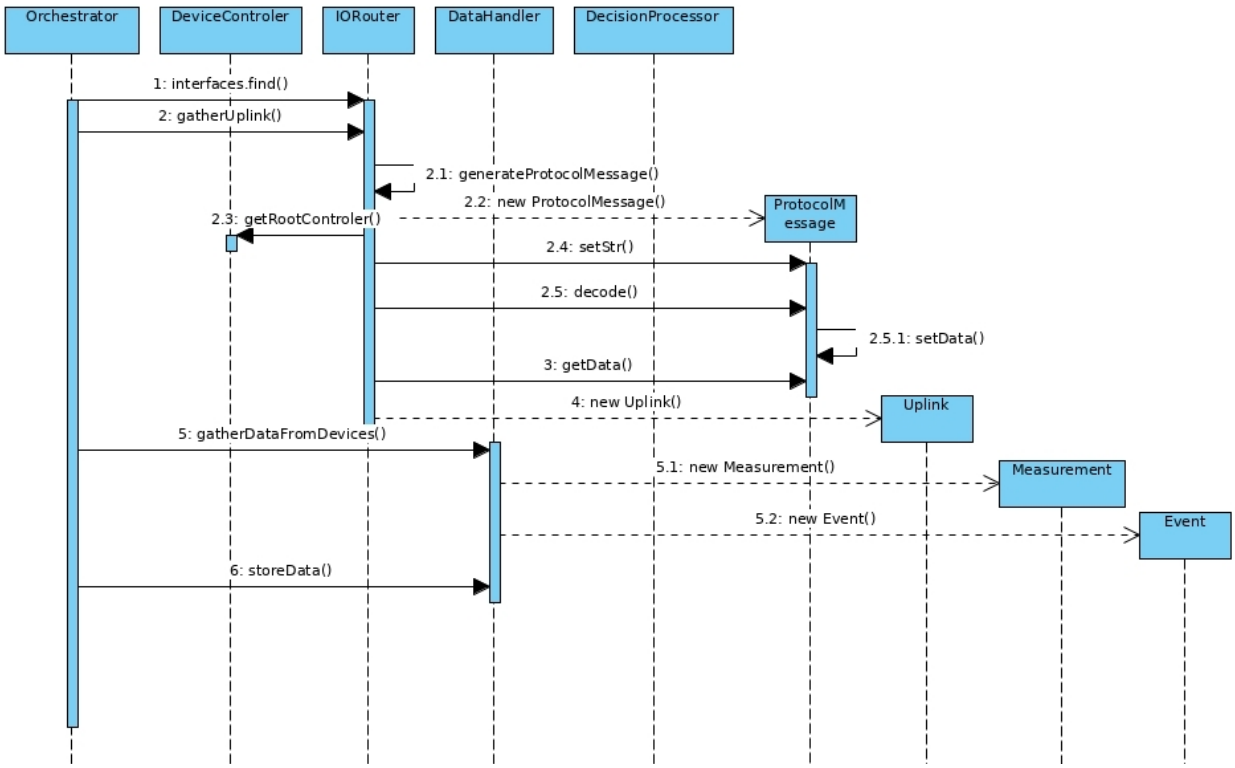


**Figure 4.4.** Uplink download flow diagram from LoRaWAN server

During cloud-to-field command initiation, the appropriate action objects are created and grouped by end device by the device management module, which then creates multiple routing processes for each node (communication interface) of the recipient list and then a data map. These data maps are then encoded into a protocol message based on the respective microcontroller's operating profile (Code Snippet 4), inserted into a downlink payload, and queued for sending to the field (Figure 4.5).

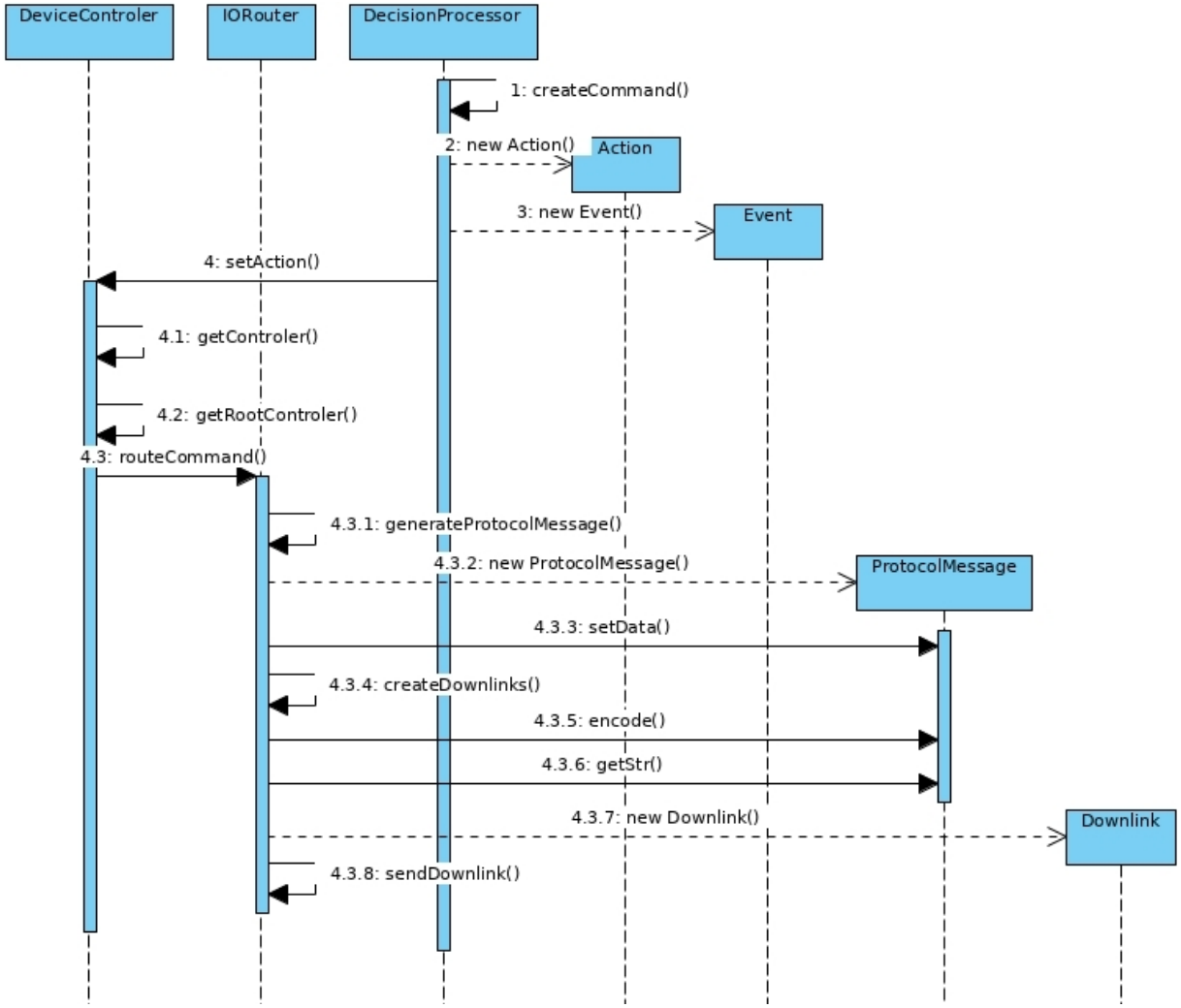


**Figure 4.5.** Decision and downlink flow diagram to the LoRaWAN server

## Application Layer

***Introduction of a novel autonomous resource management system for rural areas***

Decision making, which is done through adaptable computational models, uses data from one or more sensors that are distributed at the Edge nodes as the initial input. Later on, these data could be fused, depending on the use case with third-party data in order to enrich the application context and improve the decision efficiency. Additionally, in order to facilitate scheduled actions, decision computational models are able to extract data on specific time periods based on cron jobs. Generally, any action decided to be performed on the edge corresponds to one or more context or situational related actuators according to the inputs. Thus, in order for the system to

automatically associate the recipients of an action command, we are correlating devices to groups for the decision making (computational model feeding). To do so we have implemented dependency schemes for each Device group, which can be described as both the input (dependee) and output (dependent) of a decision computation from a model. We are using this concept to know that data processing for an input/s has some certain corresponding outputs for any action that will happen. Furthermore, because on each LoRaWAN based system we are having the limitation of message size in order to be scalable in rural areas we have implemented a queueing system. This is working by placing every potential action command to a timestamped ordered queue and shift from there whenever there is a downlink window open on the edge node. The current workflow is driven by the LoRaWAN specification, which includes three different classes of devices with different power consumption properties. By default we are using class A devices, which implements availability for downlinks once an uplink is sent. So the shortest time that an action can be sent to an edge device is equal to the default interval of pings setted up on the Edge node operation profile. To facilitate this gap, the class C devices are also implemented, where all downlinks are sended immediately to the edge, offering real time availability. On the other hand, Quality of Service (QoS) in some cases is quite doubtful for LoRaWAN deployments and can be a serious constraint for an autonomous system's stability. To handle this issue whenever an action is triggered from the Cloud a corresponding event in pending condition is created. Events are related to actuator devices that have a state which is setted from the latest related event. Once the action is triggered on the edge the Edge node adds the corresponding event to the reports queue. When the Cloud gathers this event, it gets compared with the latest corresponding event that is fetched for this device. If the value is the same the event condition gets completed and the device states the event value (on/off). If the values mismatched the event’s condition set to canceled.

### *Restful Services*

Any data that gets stored or generated on the cloud is available through a POST request on the REST API endpoint and a Bearer token authentication workflow. The requests and responses are encoded in a JSON data format.

- **Available Read Services**: read_rm, read_points, read_mcs, read_cis, read_devs, read_ops, read_events, read_notifications, read_devs_reports, read_devs_actions, read_cis_payloads, read_user
- **Available Write Services**: create_rm, create_points, create_mcs, create_actions, update_points, delete_rm, delete_points, delete_mcs

The overall logic of the routing mechanism is based on easy establishment of new endpoints based on a list of available routes that are directly pointing to an orchestrator function. The workflow consists of the initialization of the functions that will be used as middleware functions and the setup of their context of responding.

***Support for value-added services***

The application layer of the platform is responsible for providing value-added services and applications that can be offered in the agriculture domain by leveraging the data collected from the IoT devices. Indicatively the system is built with the vision to support several types of end-user applications:

- **Crop management**: This includes real-time monitoring of crops, providing insights into crop growth patterns, soil moisture content, nutrient levels, and other important parameters that can help farmers optimize crop yields and reduce waste.
- **Predictive analytics**: IoT solutions can provide data analytics capabilities that enable farmers to make data-driven decisions about crop management, pest control, and irrigation.
- **Livestock management**: IoT solutions can be used to monitor the health and well-being of livestock, including tracking their location, feed consumption, and overall health.
- **Supply chain management**: IoT solutions can help farmers and distributors monitor the temperature and humidity of goods in transit, ensure product quality and safety, and optimize supply chain efficiency.
- **Environmental monitoring**: IoT solutions can be used to monitor the environment, including temperature, humidity, air quality, and other factors that can impact crop growth and yield.

- **Precision agriculture**: IoT solutions can enable farmers to use data to make more precise decisions about planting, fertilization, irrigation, and other key aspects of crop management.
- **Decision support systems**: IoT solutions can provide farmers with real-time data and insights that can be used to make informed decisions about crop management, pest control, and other key factors that impact crop yields and quality.

# Chapter 5 - Use cases and results

Many agricultural systems focus on addressing specific aspects of the cultivation process, rather than integrating the entire lifecycle from planting to harvesting. An example of this is irrigation systems, which use IoT solutions to optimize the use of water resources in agriculture. These systems can help prevent soil diseases caused by excessive watering by measuring soil moisture levels through sensors and using this data to control the irrigation source. Alternatively, more sophisticated systems can combine humidity data with weather datasets to determine the appropriate amount of water required during irrigation.

The development of the current IoT platform has been conducted throughout the master degree of the author. In parallel, in order to assess the feasibility of utilizing our IoT platform for precision agriculture we have tested and evaluated it based on two water resource management scenarios throughout the two related scientific field projects that the author participated ("Action for Research in the Agri-Food Sector of Crete" and “Integrated irrigation network monitoring system”). The developed platform complements the server on the cloud infrastructure. Thus, this thesis includes some shared research and evaluation that was conducted and a network and cloud infrastructure on top of which modules have been developed with the aforementioned projects. Specifically, the LoRaWAN infrastructure has been used to make assessments on communication and project use cases have been used to design and evaluate the IoT platform. Additionally, three more subsystems, that have been developed throughout the aforementioned projects, are required to complete the evaluation: a Mesh WSN network, an Arduino-based node with smart flowmeters and an end user application based on cloud web sockets services. However, the details of these additional subsystems fall outside the scope of this study.

Because the system is dynamic the usage scenarios differ only at the installation/hardware level and management model (as long as data processing is necessary). The usage scenarios we have currently implemented, installed and tested are two. The first covers the management of water resources of multiple fields at the point of distribution (collector) and the second covers the autonomous management of the irrigation of a farm consisting of several hectares of olive crops.

## Scenario of integrated management of irrigation water resources

In order to implement the specific scenario, we installed a "collector" node in the HMU experimental olive grove. The goal of the node is to manage water resources by measuring water consumption and controlling the supply through solenoid valves. More generally, the goal of irrigation water collectors is to measure the consumption of multiple irrigation supplies at the point of distribution. To test this scenario we proceed with a customized solution where at each output of the collector instead of a field we have a row of olive trees, while in total we use 2 outputs. In this way, the members of the arboriculture department of HMU will be able to control the amount of watering for each row separately through our platform. Although this scenario is hybrid, it still remains suitable for us in the case of a single collector and the supplies are many and the quantities of water quite large, comparable to a normal irrigation network collector.

The installation consists of a specially designed water collector and the electronic equipment. The electronic equipment includes the node, the sensors, which are flowmeters, pressure gauge and thermometer, and the actuators which are the electrovalves (Figure 5.1). All sensors and actuators were connected according to the "collector" device group standard to be compatible with the platform. Also, integrated in the collector are the power supply circuit supplied by photovoltaic (12V 20W) with dimensions 44*35 cm and a lead battery (12V 7AH).

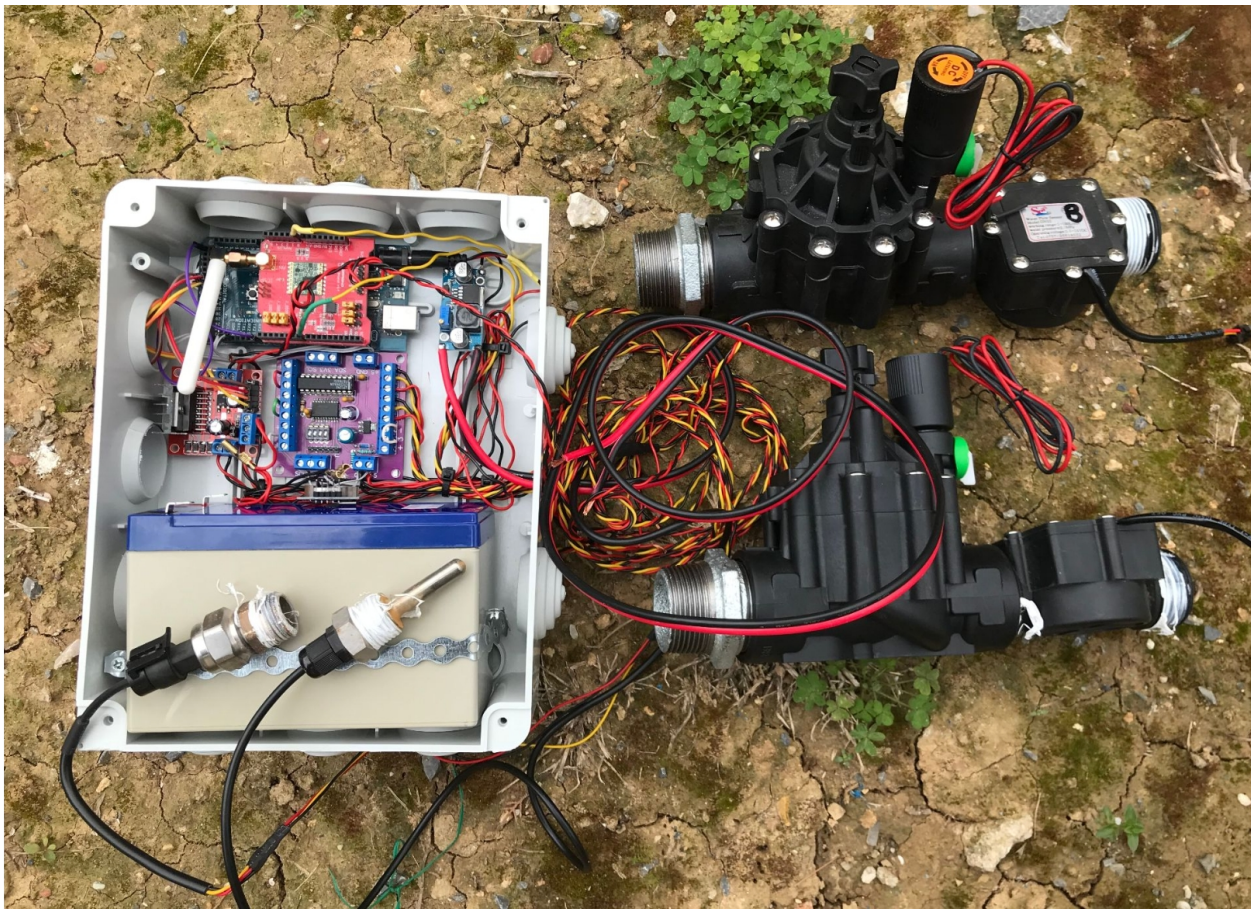

**Figure 5.1.** Collector node electronic circuit

The usage process starts with device initialization. We open the application and run the resource manager initialization wizard. To create the resource manager and start the system we add the spatial points where the node is located, select the type of the node, as well as the operation scenario. The cloud computing then processes the request and creates a new resource manager. We then run the end node which is configured to send flow measurements to the cloud every 10 minutes. Cloud computing stores the metrics in the database and pushes them to the application when requested by the user. The user has the ability to access all devices included in a field end node, along with graphs depicting reference values in the time domain, and can open and close any solenoid valve from the same device list.

### Irrigation scenario depending on microclimate and soil conditions

To test the irrigation scenario of a farm consisting of several acres of olive trees based on soil moisture, we decided to add an "extended irrigator node" type node (based on our Hybrid IoT edge node) to an output of the collector node of the HMU experimental olive grove. This node is completely separate from the collector node as the only thing they share is a water line. The extended sprinkler hub consists of a solenoid valve mounted on the inlet line and the hub with 2 lithium batteries (18650 3.7V) (Figure 5.2). Also, the term extended means that there is also a wireless sub-network of sensors (Mesh WSN RFM69) which forwards data to the central node. In this case, we installed a wireless soil moisture sensor placed at a point in the row of the olive grove that we control in this particular scenario. This node is powered by a lithium battery (18650 3.7V). All devices were connected according to the device group standard to be compatible with the platform.

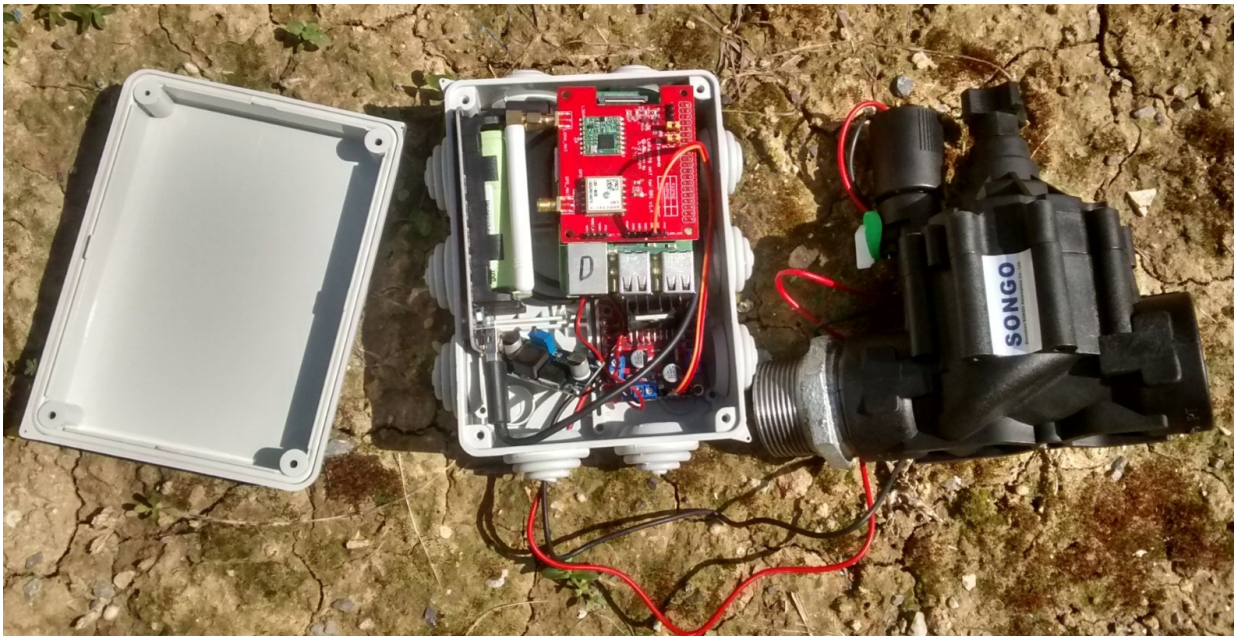


**Figure 5.2.** Electronic circuit of an extended irrigator node

The usage process starts with device initialization. We open the application and run the resource manager initialization wizard. To create the resource manager and start the system we add the spatial points where the nodes are located, we select the node type, as well as the operation scenario. The cloud computing then processes the request and creates a new resource

manager. Next we run the end nodes. The central endpoint starts the self-adaptation process. The node sends an initialization message to the cloud computing and it responds with the operating profile it creates based on our choices. Then the node receives the operation profile, defines the parameters and starts its operation. The node of the particular installation is set to send measurements every 5 minutes which include the soil moisture measurements forwarded by the subnet. The end nodes throughout their operation collect and send data from the sensors. Cloud computing stores them in the database and pushes them to the application when requested by the user. The user has the ability to access all devices included in a field endpoint, along with graphs depicting reference values in the time domain. Actuators in the field are activated according to the operating scenario or manually on user demand. In this scenario, we chose the Drip API as the decision-making model and thus during the initialization it was defined that scheduled executions of the computational model would be done, once a day, collecting data from the node with the soil moisture sensor. The model decides if actions need to be programmed, which are limited to opening and closing the solenoid valve based on the desired watering time. Finally, from the list of devices of a node, the user can open and close any solenoid valve.

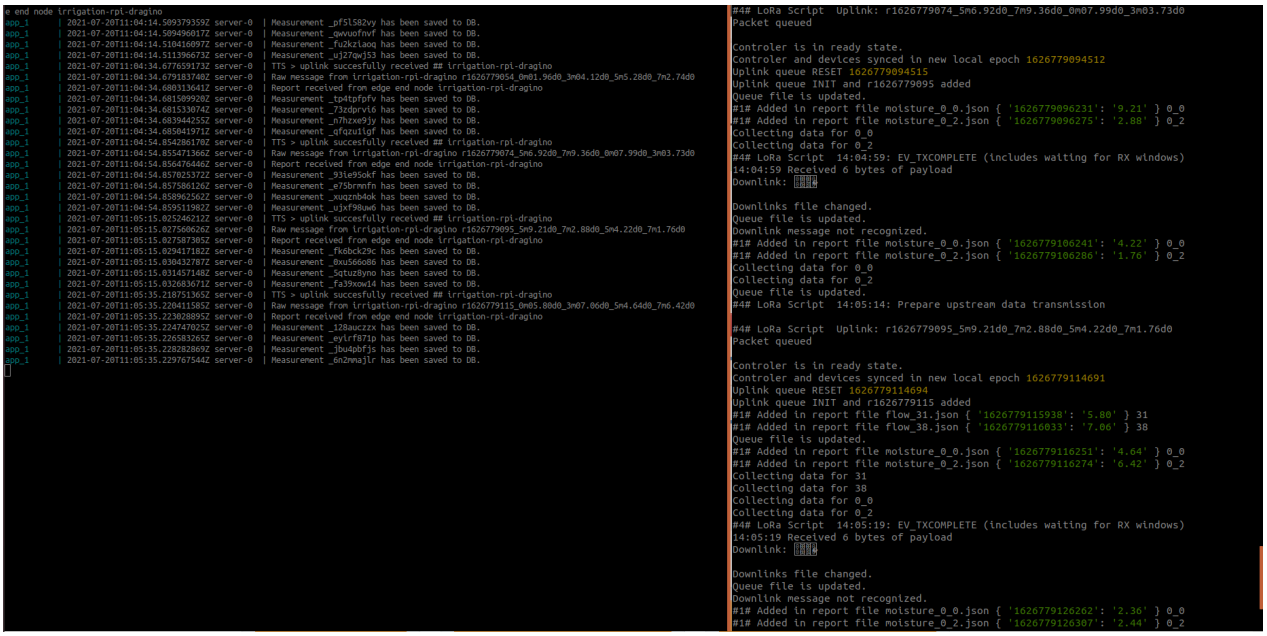

**Figure 5.3.** Excerpt from the log file from the field end node (right) and from the cloud computing (left) when collecting, sending and storing measurements.

# Chapter 6 - Outcomes & Conclusions

The tests showed that the core of the application running in the cloud responds satisfactorily to the cases of heavy workload (multiple resource managers, many users, etc.) that meets the scalability necessities of such a system. Furthermore, the flexibility and the easy customization showed that a developer or a user can tailor his/her specific needs with a small learning curve.

Achieving data and device diversity is especially important in the context of agriculture, where a variety of sensors and devices are used to monitor different aspects of the farming process. The presented IoT platform supports diverse data types, such as soil moisture, temperature, and humidity, and is suitable for providing farmers with a more comprehensive understanding of their crops and soil. Additionally, the current IoT platform can support a variety of devices, such as low-power sensors, drones, and cameras, which provide farmers a range of tools to monitor their fields and crops. By achieving data and device diversity in an agriculture-focused IoT platform, farmers can gain valuable insights into their farming practices, optimize their resource usage, and improve their crop yields. The data diversity that we have achieved can help to ensure that the platform can be used in a variety of value-added services within agriculture. Also the device diversity nature of the platform provides more options for users and developers, allowing them to choose the most suitable devices for their specific application. It is essential that the IoT platform could support both LoRaWAN and Wi-Fi because in such a way it can support a wider range of devices, such as low-power sensors and high-bandwidth cameras. Finally, our main objective from the beginning that has been lastly met was to achieve hardware agnosticism in the platform. An effective platform should be capable of supporting a wide range of devices, ranging from low-capacity devices to full-fledged computers with an operating system, as well as both wired and wireless connections. It is up to the user or expert designing the IoT product to decide which devices to use. Ultimately, when the previous summarization concatenated with the low cost of the supported hardware along with the integration with LoRaWAN solutions that provided make the proposed solution a well based IoT platform able to offer data and device diversity for Agriculture 4.0 scenarios.

# Chapter 7 - Future work

Our desire is to extend the existing IoT platform with modules and algorithms in order to create a novel unit-level granularity Digital Twin platform for Smart Farming. To do so we need to adapt our scalable and open architecture IoT platform to a Digital Twin technology ecosystem and facilitate it within a Fog computing infrastructure. Our thought is to get the full potential of Digital Twin technology is the Fog-based approach that efficiently bypasses the constraints of real-time synchronization and the seamless access to data within low-bandwidth channels. Digital Twins will detect errors, acquire new information and predict the behavior of each subsystem or component. These features are exploiting the edge, fog and cloud computing capabilities and point to the “softwarization” of physical objects [9]. More specifically, our novel approach on contextualizing the behavior of plants in a unit level is based on the fuse of spectral cognitions with several other environmental and electrochemical features. In this way we will handle a large variety of interrelated objects that create interdependencies between entities over different granularity levels. With this approach our research proposal goes beyond the state-of-the-art by increasing the granularity of Digital Twins while contextualizing other data sources that the PO itself [32, 34] and decode the physiological activities within a plant through its lifetime [14]. Additionally, our time series recollection machine learning prediction algorithm and the fuzzy logic DSS are advancing capabilities dealing with the literature concerns on prediction and prescription of Digital Twin [24]. While some projects have dealt with those concerns, their overall output is not yielding an evaluated platform ready to be used by a farmer on a specific use case [18, 34]. The assistance on critical decision-making that will finally affect the farmer’s choices is a pivotal objective for us and also is placed on the literature [30, 34]. Because our goal in that case is to research and develop a novel unit-level Digital Twin system we have to depend on non-vague schemes to form a real Digital Twin implementation. To have at least the basic implementation of the concept we need to ensure that the LO fully represents and behaves like the PO within its operational context. To do so Reflection, Entanglement, Virtualization, Representativeness and Spatiotemporal Contextualization are some key requirements we need to initial study and design [9]. Additionally, to extend and increase the intrinsic value of the Digital Twin relation we need to study about the ways to achieve concepts

such as Connectivity, Promptness, Association, Persistency, Predictability, Memorization, Augmentation and Servitization [9].

# References Or Bibliography


[1] Y. Liu, X. Ma, L. Shu, G. P. Hancke and A. M. Abu-Mahfouz, "From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research Challenges," in IEEE Transactions on Industrial Informatics, vol. 17, no. 6, pp. 4322-4334, June 2021, doi: 10.1109/TII.2020.3003910.

[2] Saiz-Rubio, V.; Rovira-Más, F. From Smart Farming towards Agriculture 5.0: A Review on Crop Data Management. Agronomy 2020, 10, 207. https://doi.org/10.3390/agronomy10020207

[3] Navarro, E.; Costa, N.; Pereira, A. A Systematic Review of IoT Solutions for Smart Farming. Sensors 2020, 20, 4231. https://doi.org/10.3390/s20154231

[4] M. S. Farooq, S. Riaz, A. Abid, K. Abid and M. A. Naeem, "A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming," in IEEE Access, vol. 7, pp. 156237-156271, 2019, doi: 10.1109/ACCESS.2019.2949703.

[5] Glaroudis, Dimitrios & Iossifides, Athanasios & Chatzimisios, Periklis. (2019). Survey, Comparison and Research Challenges of IoT Application Protocols for Smart Farming. Computer Networks. 107037. 10.1016/j.comnet.2019.107037.

[6] Achilles D. Boursianis, Maria S. Papadopoulou, Panagiotis Diamantoulakis, Aglaia Liopa-Tsakalidi, Pantelis Barouchas, George Salahas, George Karagiannidis, Shaohua Wan, Sotirios K. Goudos, Internet of Things (IoT) and Agricultural Unmanned Aerial Vehicles (UAVs) in smart farming: A comprehensive review, Internet of Things, Volume 18, 2022, 100187, https://doi.org/10.1016/j.iot.2020.100187

[7] Angin, P., Anisi, M.H., Göksel, F., Gürsoy, C., & Büyükgülcü, A. (2020). AgriLoRa: A Digital Twin Framework for Smart Agriculture. J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl., 11, 77-96.

[8] Cor Verdouw, Bedir Tekinerdogan, Adrie Beulens, Sjaak Wolfert, Digital twins in smart farming, Agricultural Systems, Volume 189, 2021, 103046, ISSN 0308-521X, https://doi.org/10.1016/j.agsy.2020.103046.

[9] R. Minerva, G. M. Lee and N. Crespi, "Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models," in Proceedings of the IEEE, vol. 108, no. 10, pp. 1785-1824, Oct. 2020, doi: 10.1109/JPROC.2020.2998530.

[10] Jacoby M, Usländer T. Digital Twin and Internet of Things—Current Standards Landscape. Applied Sciences. 2020; 10(18):6519. https://doi.org/10.3390/app10186519

[11] D. P. Proos and N. Carlsson, "Performance Comparison of Messaging Protocols and Serialization Formats for Digital Twins in IoV," 2020 IFIP Networking Conference (Networking), Paris, France, 2020, pp. 10-18.

[12] T. R. Sreedevi and M. B. Santosh Kumar, "Digital Twin in Smart Farming: A Categorical Literature Review and Exploring Possibilities in Hydroponics," 2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA), Cochin, India, 2020, pp. 120-124, doi: 10.1109/ACCTHPA49271.2020.9213235.

[13] J. Monteiro, J. Barata, M. Veloso, L. Veloso and J. Nunes, "Towards Sustainable Digital Twins for Vertical Farming," 2018 Thirteenth International Conference on Digital Information Management (ICDIM), Berlin, Germany, 2018, pp. 234-239, doi: 10.1109/ICDIM.2018.8847169.

[14] Idoje, Godwin, Tasos Dagiuklas and Muddesar Iqbal. “Survey for smart farming technologies: Challenges and issues.” Comput. Electr. Eng. 92 (2021): 107104.

[15] Vasileios Moysiadis, Panagiotis Sarigiannidis, Vasileios Vitsas, Adel Khelifi, Smart Farming in Europe, Computer Science Review, Volume 39, 2021, 100345, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2020.100345.

[16] Sen, A.A.A., Yamin, M. Advantages of using fog in IoT applications. Int. j. inf. tecnol. 13, 829–837 (2021). https://doi.org/10.1007/s41870-020-00514-9

[17] Qureshi, R., Mehboob, S.H. & Aamir, M. Sustainable Green Fog Computing for Smart Agriculture. Wireless Pers Commun 121, 1379–1390 (2021). https://doi.org/10.1007/s11277-021-09059-x

[18] N. Ahmed, D. De and I. Hussain, "Internet of Things (IoT) for Smart Precision Agriculture and Farming in Rural Areas," in IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4890-4899, Dec. 2018, doi: 10.1109/JIOT.2018.2879579.

[19] Guadalupe Ortiz, Meftah Zouai, Okba Kazar, Alfonso Garcia-de-Prado, Juan Boubeta-Puig, Atmosphere: Context and situational-aware collaborative IoT architecture for edge-fog-cloud computing, Computer Standards & Interfaces, Volume 79, 2022, 103550, ISSN 0920-5489, https://doi.org/10.1016/j.csi.2021.103550.

[20] Kalyani Y, Collier R. A Systematic Survey on the Role of Cloud, Fog, and Edge Computing Combination in Smart Agriculture. Sensors. 2021; 21(17):5922. https://doi.org/10.3390/s21175922

[21] Baghrous, M., Ezzouhairi, A., Benamar, N. (2020). Smart Farming System Based on Fog Computing and LoRa Technology. In: Bhateja, V., Satapathy, S., Satori, H. (eds) Embedded Systems and Artificial Intelligence. Advances in Intelligent Systems and Computing, vol 1076. Springer, Singapore. https://doi.org/10.1007/978-981-15-0947-6_21

[22] T. N. Gia, L. Qingqing, J. P. Queralta, Z. Zou, H. Tenhunen and T. Westerlund, "Edge AI in Smart Farming IoT: CNNs at the Edge and Fog Computing with LoRa," 2019 IEEE AFRICON, Accra, Ghana, 2019, pp. 1-6, doi: 10.1109/AFRICON46755.2019.9134049.

[23] N. Ahmed, D. De and I. Hussain, "Internet of Things (IoT) for Smart Precision Agriculture and Farming in Rural Areas," in IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4890-4899, Dec. 2018, doi: 10.1109/JIOT.2018.2879579.

[24] Natasja Ariesen-Verschuur, Cor Verdouw, Bedir Tekinerdogan, Digital Twins in greenhouse horticulture: A review, Computers and Electronics in Agriculture, Volume 199, 2022, 107183, ISSN 0168-1699, https://doi.org/10.1016/j.compag.2022.107183.

[25] Fernández-Ahumada LM, Ramírez-Faz J, Torres-Romero M, López-Luque R. Proposal for the Design of Monitoring and Operating Irrigation Networks Based on IoT, Cloud Computing and Free Hardware Technologies. Sensors. 2019; 19(10):2318. https://doi.org/10.3390/s19102318

[26] Symeonaki E, Arvanitis K, Piromalis D. A Context-Aware Middleware Cloud Approach for Integrating Precision Farming Facilities into the IoT toward Agriculture 4.0. Applied Sciences. 2020; 10(3):813. https://doi.org/10.3390/app10030813

[27] Luis Bustamante A, Patricio MA, Molina JM. Thinger.io: An Open Source Platform for Deploying Data Fusion Applications in IoT Environments. Sensors. 2019; 19(5):1044. https://doi.org/10.3390/s19051044

[28] Partha Pratim Ray, A survey of IoT cloud platforms, Future Computing and Informatics Journal, Volume 1, Issues 1–2, 2016, Pages 35-46, ISSN 2314-7288, https://doi.org/10.1016/j.fcij.2017.02.001.

[29] M. Asemani, F. Abdollahei and F. Jabbari, "Understanding IoT Platforms : Towards a comprehensive definition and main characteristic description," 2019 5th International Conference on Web Research (ICWR), Tehran, Iran, 2019, pp. 172-177, doi: 10.1109/ICWR.2019.8765259.

[30] Trilles S, González-Pérez A, Huerta J. An IoT Platform Based on Microservices and Serverless Paradigms for Smart Farming Purposes. Sensors. 2020; 20(8):2418. https://doi.org/10.3390/s20082418

[31] Akpolat, C., Şahinel, D., Sivrikaya, F., Lehmann, G., & Albayrak, S. (2017). CHARIOT: An IoT Middleware for the Integration of Heterogeneous Entities in a Smart Urban Factory. Conference on Computer Science and Information Systems.

[32] Adam Wolnikowski, Stephen Ibanez, Jonathan Stone, Changhoon Kim, Rajit Manohar, and Robert Soulé. 2021. Zerializer: towards zero-copy serialization. In Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS '21).

Association for Computing Machinery, New York, NY, USA, 206–212. https://doi.org/10.1145/3458336.3465283

[33] Amrit Kumar Biswal, Obada Almallah, Marco Brambilla, Analytical Assessment of Binary Data Serialization Techniques in IoT Context, Dipartimento di Elettronica, Informazione e Bioingegneria, 2019

[34] Zhai, Z., Martínez, J., Beltran, V., & Martínez, N.L. (2020). Decision support systems for agriculture 4.0: Survey and challenges. Comput. Electron. Agric., 170, 105256.

[35] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui and T. Watteyne, "Understanding the Limits of LoRaWAN," in IEEE Communications Magazine, vol. 55, no. 9, pp. 34-40, Sept. 2017, doi: 10.1109/MCOM.2017.1600613.

[36] Mekki, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. ICT Express, 5, 1-7.

[37] A. Lavric and A. I. Petrariu, "LoRaWAN communication protocol: The new era of IoT," 2018 International Conference on Development and Application Systems (DAS), Suceava, Romania, 2018, pp. 74-77, doi: 10.1109/DAAS.2018.8396074.

[38] A. Grunwald, M. Schaarschmidt and C. Westerkamp, "LoRaWAN in a rural context: Use cases and opportunities for agricultural businesses," Mobile Communication - Technologies and Applications; 24. ITG-Symposium, Osnabrueck, Germany, 2019, pp. 1-6.

[39] A. Pötsch and F. Haslhofer, "Practical limitations for deployment of LoRa gateways," 2017 IEEE International Workshop on Measurement and Networking (M&N), Naples, Italy, 2017, pp. 1-6, doi: 10.1109/IWMN.2017.8078360.

[40] F. Cuomo, M. Campo, A. Caponi, G. Bianchi, G. Rossini and P. Pisani, "EXPLoRa: Extending the performance of LoRa by suitable spreading factor allocations," 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Rome, Italy, 2017, pp. 1-8, doi: 10.1109/WiMOB.2017.8115779.

[41] T. Elshabrawy and J. Robert, "Capacity Planning of LoRa Networks With Joint Noise-Limited and Interference-Limited Coverage Considerations," in IEEE Sensors Journal, vol. 19, no. 11, pp. 4340-4348, 1 June1, 2019, doi: 10.1109/JSEN.2019.2897156.

[42] J. -T. Lim and Y. Han, "Spreading Factor Allocation for Massive Connectivity in LoRa Systems," in IEEE Communications Letters, vol. 22, no. 4, pp. 800-803, April 2018, doi: 10.1109/LCOMM.2018.2797274.

[43] M. Bor and U. Roedig, "LoRa Transmission Parameter Selection," 2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS), Ottawa, ON, Canada, 2017, pp. 27-34, doi: 10.1109/DCOSS.2017.10.

[44] A. -I. Pop, U. Raza, P. Kulkarni and M. Sooriyabandara, "Does Bidirectional Traffic Do More Harm Than Good in LoRaWAN Based LPWA Networks?," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, 2017, pp. 1-6, doi: 10.1109/GLOCOM.2017.8254509.

[45] P. S. Cheong, J. Bergs, C. Hawinkel and J. Famaey, "Comparison of LoRaWAN classes and their power consumption," 2017 IEEE Symposium on Communications and Vehicular Technology (SCVT), Leuven, Belgium, 2017, pp. 1-6, doi: 10.1109/SCVT.2017.8240313.

[46] J. M. Marais, R. Malekian and A. M. Abu-Mahfouz, "LoRa and LoRaWAN testbeds: A review," 2017 IEEE AFRICON, Cape Town, South Africa, 2017, pp. 1496-1501, doi: 10.1109/AFRCON.2017.8095703.

[47] Marais, J.M., Malekian, R., & Abu-Mahfouz, A.M. (2019). Evaluating the LoRaWAN Protocol Using a Permanent Outdoor Testbed. IEEE Sensors Journal, 19, 4726-4733.

[48] B. Vejlgaard, M. Lauridsen, H. Nguyen, I. Z. Kovacs, P. Mogensen and M. Sorensen, "Interference Impact on Coverage and Capacity for Low Power Wide Area IoT Networks," 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 2017, pp. 1-6, doi: 10.1109/WCNC.2017.7925510.

[49] Martínez, B., Adelantado, F., Bartoli, A., & Vilajosana, X. (2018). Exploring the Performance Boundaries of NB-IoT. IEEE Internet of Things Journal, 6, 5702-5712.

[50] Sinha, R.S., Wei, Y., & Hwang, S. (2017). A survey on LPWA technology: LoRa and NB-IoT. ICT Express, 3, 14-21.

[51] S. Aggarwal and A. Nasipuri, "Improving Scalability of LoRaWAN Networks by Spreading Factor Distribution," SoutheastCon 2021, Atlanta, GA, USA, 2021, pp. 1-7, doi: 10.1109/SoutheastCon45413.2021.9401855.

[52] Lenders, M.S., Kietzmann, P., Hahm, O., Petersen, H., Gündoğan, C., Baccelli, E., Schleiser, K., Schmidt, T.C., & Wählisch, M. (2018). Connecting the World of Embedded Mobiles: The RIOT Approach to Ubiquitous Networking for the Internet of Things. ArXiv, abs/1801.02833.

[53] Ji, S., Xu, W., Yang, M., & Yu, K. (2018). 3D convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(5), 1138-1152.

[54] A comprehensive reference for AI can be found in the textbook "Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig.

[55] Fragkopoulos, M.; Panagiotakis, S.; Kostakis, M.; Markakis, E.K.; Astyrakakis, N.; Malamos, A. Experimental Assessment of Common Crucial Factors That Affect LoRaWAN Performance on Suburban and Rural Area Deployments. Sensors 2023, 23, 1316. https://doi.org/10.3390/s23031316

# Appendix A - Άδεια χρήσης Creative Commons

Attribution - Αναφορά Δημιουργού 4.0 (CC BY)