UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXIV

Design and Development of IoT Architectures for Smart Agriculture Systems

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutore:
Chiar.mo Prof. Gianluigi Ferrari

Dottoranda: Gaia Codeluppi

Anni 2018/2021

A chi esplora il mondo con curiosità ed è protagonista della vita.
A chi sogna senza riserve e combatte instancabilmente,
con perseveranza, strategia e pragmatismo,
per realizzare le proprie idee.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
AIOTI	Alliance for Internet of Things Innovation WG03
AMQP	Advanced Message Queuing Protocol
ANNs	Artificial Neural Networks
AP	Access Point
APIs	Application Programming Interfaces
AS	Application Server
BLE	Bluetooth Low Energy
BP	Back-Propagation
CC	CC3200-LAUNCHXL
CoAP	Constrained Application Protocol
DDS	Data Distribution Service
ENs	End Nodes
FAO	Food and Agriculture Organization
FMS	Farm Management System
GPS	Global Positioning System
GW	gateway
HL	High-Level
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol

ISM	Industrial, Scientific and Medical
ISO/OSI	International Organization for Standardization Open Systems Interconnection
ITU-T	Telecommunication Standardization sector of International Telecommunication Union
LL	Low-Level
LSTM	Long Short-Term Memory
MAC	Multiply–ACcumulate
MAPE	Mean Absolute Percentage Error
MCU	Micro Controller Unit
ML	Machine Learning
MLP	Multi-Layer Perceptron
mp-GW	multi-protocol Gateway
MQTT	Message Queuing Telemetry Transport
MW	MiddleWare
NB-IoT	Narrowband IoT
NIR	Near InfraRed
NN	Neural Networks
NS	Network Server
OSs	Operating Systems
PaaS	Platform as a Service
QoS	Quality of Service
$\mathbf{R}^2$	Coefficient of determination
REST	REpresentational State Transfer
RMSE	Root Mean Squared Error
RNNs	Recurrent Neural Networks
RPi	Raspberry Pi
RSSI	Received Signal Strength Indicator
SA	Smart Agriculture

SDN	Software-Defined Networking
SF	Smart Farming
SNs	Sensor Nodes
SOA	Service Oriented Architecture
SOs	Smart Objects
UAV	Unmanned Aerial Vehicles
UGV	Unmanned Grounded Vehicles
WSNs	Wireless Sensor Networks
XMPP	Extensible Messaging and Presence Protocol

Acknowledgment

Un percorso di dottorato può essere spesso un viaggio solitario, nel quale è facile perdere di vista l'obiettivo finale e la propria motivazione nel raggiungerlo. A volte, infatti, si percorre una strada ben segnata, altre volte si procede a passi incerti in un sentiero buio, molte altre non si hanno elementi sufficienti per decidere quale direzione scegliere tra le tante possibili. Ci si può sentire terribilmente persi, galvanizzati da una nuova idea da esplorare, o (anche) semplicemente annoiati. Nonostante si tratti di un percorso molto solitario, durante quest'ultimo si incontrano persone che in un modo o nell'altro ti accompagnano. C'è chi lo fa aiutandoti a superare i momenti difficili, supportandoti nel lavoro o semplicemente prestandoti un orecchio per ascoltare i tuoi dubbi e pensieri, chi condividendo con te i tuoi successi, e chi fa un po' tutte queste cose. È anche grazie a queste persone se sono arrivata fino a qui: per questo motivo ora vorrei ringraziarle.

Innanzi tutto, ringrazio il mio relatore, il prof. Gianluigi Ferrari, per avermi dato la possibilità d'intraprendere questo percorso e avermi guidato in esso. Ringrazio il mio super collega Luca per la sua sempre presente disponibilità nell'aiutarmi a risolvere qualunque problematica accademica (e anche per essere un simpaticissimo compagno di trasferte). Ringrazio il mio collega Gabriele per aver reso ogni giorno passato insieme in lab divertente e non ordinario (peccato averne passati così pochi insieme e non essere potuti tornare nella piscina estiva del campus causa Covid).

Ringrazio la Regione Emilia-Romagna e l'Unione Europea per aver finanziato le mie attività di dottorato, rispettivamente mediante una borsa di dottorato e i fondi del progetto Europeo AFarCloud, nel quale ho avuto modo di lavorare negli ultimi tre

anni. Ringrazio Niccolò, il proprietario della fattoria in cui sono stati testati i sistemi presentati in questa tesi, per la sua disponibilità e cordialità (romagnola). Ringrazio lo staff del Parco di San Rossore per la loro disponibilità nel supportarci durante le installazioni effettuate al parco e le relative attività del progetto AFarCloud.

Ringrazio la mia amica Debby per esserci sempre stata per ascoltarmi, supportarmi nel risolvere i momenti problematici in ogni sfera della mia vita, e gioire con me per i miei successi. Ringrazio il mio amico Dario per la sua longeva amicizia, da anni una delle (poche?) certezze nella mia vita.

Un enorme ringraziamento va a tutta la mia famiglia (Valeria, Asia, Flavio, Anna, Gianni ed Erio): il vostro amore e supporto incondizionato sono tra le cose più preziose che ho. Senza di voi questo traguardo (come molti altri) non sarebbe stato possibile.

Infine, ringrazio me stessa per non aver mollato nonostante i momenti difficili, essere arrivata fino a qui e avermi dimostrato (ancora una volta) che coltivare i propri sogni, lavorando sodo per raggiungerli, ripaga sempre!

Introduction

Agricultural production processes, which include a wide range of practices aiming at cultivating plants and livestock animals to produce foods, fibers, fuels, and raw materials, is nowadays undergoing noteworthy transformations, which will be even more massive in the future. Indeed, future trends such as global population growth, climate changing, intensification of natural hazards, and poor countries hunger, together with the challenges of improving food safety and quality, are forcing the present food system (and, thus, agriculture) to become more productive, sustainable, and inclusive [1]. This means that agricultural processes have to be deeply re-designed and transformed into a set of more optimized and environmental-friendly activities. Key enablers in leading the aforementioned evolution of the agricultural sector, which is usually identified by the terms Smart Agriculture (SA) and Smart Farming (SF), are digital technologies [2].

Indeed, a wide range of digital technologies can support the diffusion of SA. They include sensors, able to monitor environmental variables which are relevant for agricultural production (e.g., soil moisture), and autonomous vehicles, as smart tractors and drones, automatically managing operations such as spraying and plowing. Furthermore, additional involved technologies are Internet of Things (IoT) and analytical tools, such as Artificial Intelligence (AI) algorithms, proving sophisticated techniques to process data relevant for a farm's production (e.g., forecasting the occurrence of a plant disease starting from environmental data collected from sensors).

From an engineering perspective, there exists a big barrier limiting the diffusion

of SA and, thus, the development of future highly technological farms (namely, *high-tech* or *smart farms*). This obstacle is related to the definition of proper strategies aiming at integrating multiple heterogeneous SA-oriented technologies within a single system, usually denoted as Farm Management System (FMS), which is robust, flexible, and can automatically manage, in a simple and scalable way, the production activities of a farm. A crucial role in addressing this challenge is played by IoT technologies. In fact, the usage of IoT allows building systems (i.e., IoT/SA-oriented systems) able to collect, process, share, and store data relevant for agricultural processes, thus enabling the deployment of FMSs which can create new value from farm-related data. In detail, the above-mentioned relevant data, coming from physical devices installed in the farms (e.g., sensors, tractors, smart collars) or external data sources (e.g., historical databases), are managed by a networking and computing (IoT) infrastructure, which processing components of are distributed between local networks of devices placed in the farms and the Cloud (in other words, the infrastructure relies on both the edge and the Cloud computing paradigms).

The development of the above-mentioned IoT architecture has to deal with several challenges. Some of them naturally arise while working with IoT technologies (and include heterogeneity, modularity, scalability, and standardization), whereas some others become more incisive while dealing with agricultural scenarios (e.g., connectivity issues, harsh environment, and device power management).

Despite several IoT/SA-oriented systems proposed in literature [3–11] have significantly enhanced the management of sets of farm's activities, there is still the need to define methodologies to (i) properly handle all the challenges characterizing IoT/SA-oriented systems, and to (ii) develop general-purpose IoT systems targeting SA applications. Indeed, according to the best of this doctoral thesis's author's knowledge, although each IoT/SA-oriented architecture found in literature aims to support a reduced set of them, future smart farms will have to manage multiple and wider groups of farms' activities (in other terms, SA scenarios). This means that a modular IoT/SA-oriented architecture, targeting the management of a generic farm, regardless of its particular characteristics, but which can also be customized according to the farm's specific requirements, has not being already designed. For this

reason, one of the main goals of this doctoral thesis is to address the aforementioned research gap.

More exactly, the aim of this doctoral thesis is four-fold.

- *First*, to identify and discuss the main aspects, challenges, technologies, and requirements to be considered while designing and developing IoT systems targeting SA applications—also denoted as IoT/SA-oriented systems or architectures in this manuscript—(Chapter 1).

- *Second*, to define a theoretical architectural model (or framework) to be followed while developing general-purpose IoT/SA-oriented architectures, which are modular, scalable, able to easily integrate heterogeneous technologies and to accelerate the diffusion of smart farms. This will be carried out through theoretical and practical knowledge acquired from the literature reviews performed to reach the *first* goal and the on-field experience, respectively (Chapter 2).

- *Third*, to experimentally validate the proposed framework. This will be achieved by presenting (i) how two real IoT/SA-oriented systems, which target two different SA applications (namely, the management of small-sized crops [12] or larger generic farms [13]), have been developed following the presented architectural model; and (ii) the experimental results obtained with the two described systems (Chapter 3).

- *Fourth*, to provide some insights concerning the integration of AI tools (as NNs) within the proposed framework in order to develop data mining, analysis, and forecasting algorithms, enriching IoT/SA-oriented systems with intelligent capabilities. This will be achieved by describing the methodologies adopted to develop two forecasting NN-based algorithms, starting from data coming from one of the systems presented in the *third* objective and aiming at enhancing the management of a greenhouse [14, 15] (Chapter 4).

Thesis Structure

The thesis is structured as follows. In Chapter 1, a background related to SA, IoT, and AI is provided. In particular, in Section 1.1, future trends, challenges, and enabling technologies for SA, together with the key features of future smart farms, are presented. Then, in Section 1.2, reference models for the deployment of an IoT system are discussed, together with their building components, technologies to adopt, and open challenges to address. In Section 1.3, a literature review of IoT/SA-oriented systems is presented together with some open research gaps. Finally, in Section 1.4, AI algorithms are briefly introduced with a focus on Neural Networks (NN) and the execution of AI algorithms on constrained IoT edge devices. This is an interesting recent research trend (denoted as *EdgeAI*), which can be applied also to SA scenarios.

In Chapter 2, the proposed architectural model for IoT/SA-oriented system, in terms of building components and interactions among them, is presented. Furthermore, a few examples of relevant hardware, software, and communication technologies and tools to adopt to build such systems are discussed.

In Chapter 3, two experimental IoT/SA-oriented systems for future smart farms (namely, VegIoT Garden system [12] and LoRaFarM system [13]), based on the architectural model proposed in Chapter 2, are presented.

In Chapter 4, the two methodologies followed to develop two air temperature forecasting algorithms aiming at simplifying the management of a case of study greenhouse are discussed. Moreover, an approach (based on EdgeAI) to integrate the developed algorithms within the LoRaFarM system (presented in Chapter 2) is also outlined.

Finally, in the Conclusions, the main contributions and future research directions of this manuscript are summarized.

Chapter 1

Background and State of the Art

This chapter covers the following three main topics. *First*, the field of Smart Agriculture (SA) is introduced (Section 1.1). In detail, today's agriculture (Sub-section 1.1.1), future trends and challenges agriculture will have to handle (Sub-section 1.1.2), key enabling technologies for SA applications (Sub-section 1.1.3), and main challenges of future smart farms (Sub-section 1.1.4) are presented.

Second, Internet of Things (IoT) technologies are discussed (Section 1.2). In detail, an overview on IoT (Sub-section 1.2.1), representative reference models of an IoT system (Sub-section 1.2.2), its building components and technologies (Sub-section 1.2.3), and a reduced set of relevant open challenges (Sub-section 1.2.4) of IoT, are provided. Moreover, a literature review concerning IoT/SA-oriented systems (Sub-section 1.3.1) and related open research gaps (Sub-section 1.3.2) are presented.

Finally, main concepts concerning AI algorithms are introduced (Section 1.4), providing a general overview on the topic (Sub-section 1.4.1), some more concepts related to NNs (Sub-section 1.4.2), a brief literature review regarding their adoption to support the management of a particular agricultural scenario, namely greenhouses (Sub-section 1.4.3), and main aspects related to EdgeAI (Sub-section 1.4.4).

1.1 Smart Agriculture

1.1.1 Today's Agriculture

Agriculture (or farming) is usually defined as the practice of cultivating plants and livestock to produce foods, fibers, fuels, or raw materials. More precisely, agricultural production processes include a wide and heterogeneous set of activities: (i) the growing, harvesting, and primary processing of different type of crops; (ii) the breeding, raising, and caring of livestock animals; and (iii) the horticultural, floricultural, and nursery production [16].

From the cultivation of small-sized crops (based on manual labor) and the growing of few heads of livestock, up to nowadays, the dimension of the global agricultural production and the complexity of associated processes have significantly increased. This is mainly due to the introduction of technological innovations and novel production techniques, such as, just to name few examples, plant breeding and agro-chemicals (e.g., pesticides and fertilizers), aiming at improving agricultural processes. Although the above-mentioned contemporary (and often intensive) practices have allowed to considerably boost agricultural production, they have also raised concerns about the impact of agricultural processes on the environment and, nonetheless, livestock animal welfare. Indeed, nowadays, it is well-known that farming contributes to undesirable trends as global warming, deforestation, depletion of water and natural resources, desertification, soil degradation, antibiotic resistance, and the increase in growth hormones in meat industry. The just presented consuming trends which, as a main drawback (along with many others), will make agricultural production less effective in the future, will be even more serious in the next decades (as discussed in the following section), thus, remarking the need to reach a novel, more sustainable, optimized, and respectful agri-food system.

1.1.2 Future Trends and Challenges for Agriculture

In a report published in 2017, but still relevant at the time of this doctoral thesis's submission (namely, October 2021), the Food and Agriculture Organization (FAO)

ID	Trend	ID	Trend
1	Population growth, urbanization, and ageing	9	Nutrition and health
2	Global economic growth, investment, and trade	10	Structural change and employment
3	Increasing competition for natural resources	11	Migration and agriculture
4	Climate change	12	Changing food systems
5	Agricultural productivity and innovation	13	Food losses and waste
6	Transboundary pests and diseases	14	Governance for food security and nutrition
7	Conflicts, crises, and natural disasters	15	Development finance
8	Poverty, inequality, and food insecurity		

Table 1.1: Future trends which, according to the FAO, will influence agricultural production in the future.

highlights 15 trends influencing future agriculture and 10 challenges agriculture will have to face to properly handle them [1]. The 15 future trends outlined by the FAO are summarized in Table 1.1. Moreover, future challenges targeting agriculture together with the trends they intend to address are summarized in Table 1.2. A reduced set of the trends outlined in Table 1.1 are discussed in the following paragraphs.

One of the major trend, leading to significant changes in agricultural processes, is the global population growth (Trend 1). Indeed, even if this trend is slowing down, with respect to the last decades, in some countries (mainly in Africa and Asia), population will continue to expand. Indeed, the FAO estimates the earth population will touch 9.73 billion by 2050 and 11.2 billion by 2100 [1] (the current is 7.8 billion [17]). In addition to the increase in food demand due to a broader population, other relevant trends include urbanization and agricultural labor force aging (Trend 1). These last two phenomenons result in having less and older labors in the rural area, thus, requiring to significantly re-think food production processes [18]. Furthermore, climatic changing and dietary patterns transformation (Trend 4) will intensify natural resources competition (Trend 3). This negatively impacts on nature, exacerbating phenomenons such as water scarcities, land degradation, and deforestation. Additional issues to consider are related to food safety (Trend 8) and quality (Trend 9 and Trend 14). In fact, intensive usage of natural resources and climate changing factors (Trend 4), such as hottest temperature and more extreme

Challenge		Trends (ID)														
ID	**Description**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	Sustainably improving agricultural productivity to meet increasing demand	•	•	•		•										•
2	Ensuring a sustainable natural resource base			•	•											
3	Addressing climate change and intensification of natural hazards				•			•		•						
4	Eradicating extreme poverty and reducing inequality				•				•			•				•
5	Ending hunger and all forms of malnutrition	•			•					•						
6	Making food systems more efficient, inclusive and resilient										•		•	•		
7	Improving income earning opportunities in rural areas and addressing the root causes of migration								•		•	•				
8	Building resilience to protracted crises, disasters, and conflicts							•				•				•
9	Preventing transboundary and emerging agriculture and food system threats				•		•									
10	Addressing the need for coherent and effective national and international governance														•	•

Table 1.2: Challenges targeting the agricultural sector identified by the FAO compared with future trends they intend to handle.

weather conditions, will make food less nutrient (i.e., wheat with lower protein ad micro nutrients content) and will create favorable conditions for the spreading of food/water-borne pathogens, increasing the need of using pesticides and the risks for the health of food consumers.

As said, the presented trends will force agriculture to face new challenges. *First*, agricultural processes have to become more sustainable to satisfy the growing food

demand without negatively affecting the environment. Indeed, the request of food products is estimated to increase of 50% by 2050 (Challenge 1 and Challenge 2). *Second*, climate changing and intensification of natural hazards (Challenge 3) have to be accurately handled since they directly impact on food safety and productiveness. *Third*, hunger and malnutrition need to be overcome supporting poor countries and producing good quality, nutrient food (Challenge 4 and Challenge 5). *Lastly*, food systems efficiency, inclusiveness, and resilience have to be improved (Challenge 6), strengthening the relationships existing between farmers, markets and customers, for example.

As already mentioned, future trends and challenges will require agricultural processes to be deeply transformed and re-engineered. The introduction of a set of heterogeneous technologies (within the agricultural sector) will enable this transformation, moreover, optimizing food production, reducing unnecessary inputs and improving also sustainability [2]. As said in the Introduction, this trend of technological transformation of the agricultural sector is usually denoted with the words Smart Agriculture and Smart Farming.

1.1.3 Key Technological Enablers

According to [19–23], the major technologies aiming at supporting the digital transformation of agriculture can be divided into the following four groups: (i) sensing and actuating technologies—including Wireless Sensor Networks (WSNs); (ii) Unmanned Grounded Vehicles (UGV) and Unmanned Aerial Vehicles (UAV); (iii) IoT; and (iv) data analysis tools, as AI algorithms. Although the exhaustive description of these four groups of technologies is out of this manuscript's scope, a few representative applications adopting the mentioned technologies are reported in Table 1.3, together with a set of relevant surveys concerning the adoption of these technologies in SA applications.

As can be seen by Table 1.3, farm activities can be enhanced by several and very heterogeneous technologies. In detail, in [24, 25] literature concerning the usage of remote sensing technologies, namely sensors and WSNs, aiming at supporting agricultural productive processes of various farm's scenarios is surveyed. According

Tech-based group	Examples of applications	Reference survey(s)
Sensors & Actuators	• Controlled environments relevant variables monitoring and control • Water-related tasks management and irrigation • Pest control and weed damage monitoring • Seed germination monitoring and determination • Smart collars for the monitoring of livestock animals' position, behavior, and health	[24, 25]
UAV	• Weed mapping and management • Vegetation growth monitoring and yield estimation • Vegetation health monitoring and diseases detection • Irrigation management • Autonomous UAV-based spraying system for precision agriculture (e.g., pesticides, fertilizers) • UAV-based spraying optimization	[26]
UGV	• Management of crop activities based on autonomous driving farm machines (e.g., harvesting, seeding, fertilization) • Robots-based activities in orchards and greenhouse (e.g., to pick fruits and small products) • Robots-based weeding	[27]
IoT	• Collection, processing, and graphical representation of gathered farm data coming from sensors, tractors, smart collars, etc. • Warnings creation and decision support system • Food supply chain tracking • Vehicles and machinery control	[28, 29]
Data analysis tools	• Disease detection and forecasting • Plant classification • Land cover identification • Pest recognition • Phenotyping	[30]

Table 1.3: A set of representative farm activities which can be improved by the usage of digital technologies, grouped by the type of adopted technology.

to the aforementioned review, the gather of sensor data related to, for instance, soil, air, and animals, coming from sensing devices deployed in a farm (e.g., in stables, crops, and greenhouses), allows to, for instance, optimize irrigation tasks and the regulation of a greenhouse's internal micro climate; prevent plants from being infected by pests or be afflicted by other kind of diseases; define seeds germination; and monitor parameters relevant for animals' welfare (e.g., their habits, positions, and body temperature). In [26], 20 exemplary UAV-based applications targeting the aerial monitoring of crops or the automation of spraying operations are critically examined. A few examples of activities covered by works analyzed by the above-cited survey are: monitoring of vegetation growth and health; vegetation disease detection; and automation and optimization of UAV-based spraying operations (e.g., pesticides, fertilizers). In [27], a survey on relevant papers targeting the adoption of robots (as manipulators, ground vehicles, and aerial robots) to make automatic agricultural activities is proposed. These activities include planting, harvesting, environmental monitoring, and crop inspection and treatment. In [28, 29], the main IoT/SA-oriented technologies, their adoption in nowadays agricultural practices, their potential value for future farms, and correlated challenges are reviewed. The survey outlines that the adoption of IoT allows to deploy IoT/SA-oriented systems able to properly handle relevant data for agricultural processes; to support decision making; to create warning and alarms for farmers (e.g., to notify that a cow has an unusual behavior); and to connect other SA-oriented technologies (e.g., smart tractors, irrigation systems) among them and the Internet, and, thus, to remotely control them. In [30], the advantages of introducing AI algorithms and data analysis tools to support farm activities, together with their main fields of application, are presented on the basis of an extended literature review of 120 papers relevant for the subject. The main objectives the selected papers aim to achieve are to improve disease detection; plant classification; land cover identification; precision livestock farming; object, weed and pest recognition; phenotyping; and smart irrigation.

Furthermore, to enable the development of the so-called future *smart farms*, the above-mentioned technologies have to cooperate and, possibly, be integrated within the same FMS (as more deeply discussed in Sub-section 1.1.4). Key players in

effectively handling this challenge, which is, in other words, the development of an architecture able to integrate such wide range of technologies and to effectively manage agricultural meaningful data coming from different sources (e.g., sensors, agricultural machines, historical weather repositories), will be IoT technologies. Indeed, IoT allows to design and deploy systems (or, equivalently, architectures) able to collect, process, exchange, and store data which are relevant for agricultural processes, and to create new value from them. Due to these reasons, IoT, its prominence for future smart farms, and key challenges to consider while developing IoT systems (as heterogeneity, standardization, integration, and flexibility [31]), are covered in Section 1.2. Moreover, some more insights related to AI algorithms and their usage within IoT applications (with a focus on the SA domain) are provided in Section 1.4.

1.1.4 Future Smart Farms

As discussed in Sub-section 1.1.3, future farms are going to be smarter, more productive, and more environmental friendly thanks to the adoption of a wide range of digital technologies, as shown in Figure 1.1, displaying a possible future smart farm. Indeed, as depicted in Figure 1.1, autonomous tractors, smart collars, drones, sensors, and agribots will be integrated within the same FMS, thus, allowing to more easily manage farm activities and to reduce the number of tasks manually performed by farmworkers. Moreover, the above-mentioned FMS can be successfully implemented through the deployment of a modular and scalable architecture based on IoT technologies. This IoT architecture is, in addition, able to integrate different technologies and entities, ranging from physical devices deployed in the farm to software components, data analysis algorithms, and information (or services) coming from external data sources (e.g., historical databases, legacy systems). Several and significant challenges, including, just to name few examples, devices connectivity, agricultural harsh environment, design of easy-to-use and plug-and-play solutions, arise while creating such architecture. A brief overview of main future challenges, characterizing the deployment of IoT architectures in the SA domain, is provided in

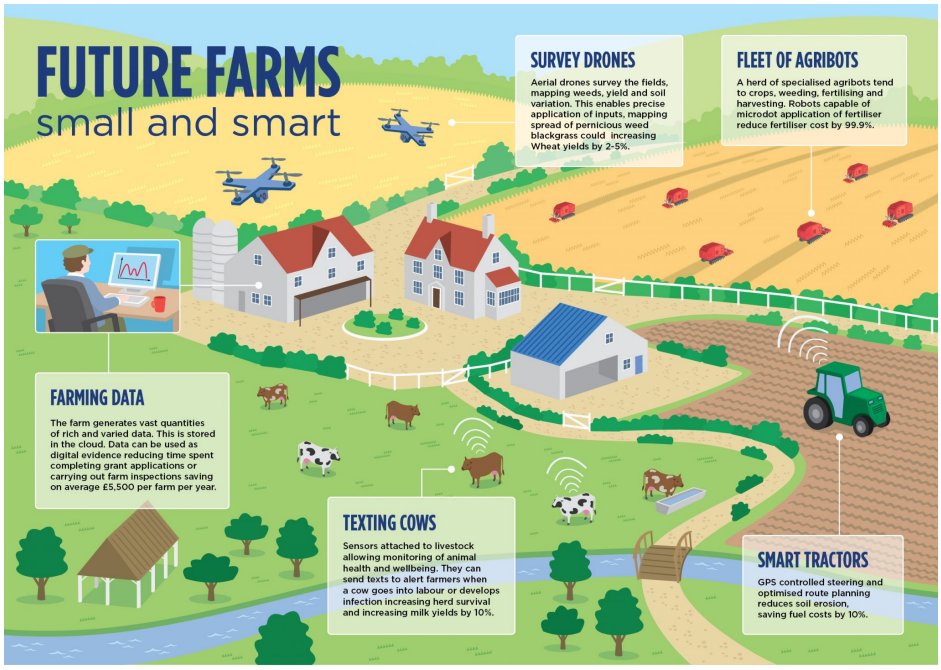


Figure 1.1: A possible future smart farm.

Table 1.4.

According to [28], the major challenges to deal with while deploying IoT systems within agricultural scenarios are related to the adopted hardware devices (which, for instance, has to be suitable to be placed in outdoor, agricultural, harsh environments); interoperability; networking (e.g., the quality of wireless communications is influenced by environmental variables as air humidity); security; and IoT stack (i.e., how data securely flow among the different components of an IoT system). In [32], the current and the potential application of IoT in arable farming is analytically surveyed and the main correlated issues to overcame highlighted. Compared with [28], in [32] the following additional challenges for IoT/SA-oriented applications are identified: revenue and affordability of adopting smart solutions for farmers (more details concerning this challenge are provided as a final remark in this sub-section); heterogeneity (of, e.g., devices, type of selected

Challenge	Description	Ref.
Connectivity	• Agricultural environments usually lack of a stable Internet connectivity • Introduction of (multiple and heterogeneous) communication technologies • (Mainly) wireless, robust, and reliable technologies	[28, 32, 33]
Heterogeneity	• Physical (devices): smart collars, tractors, drones, sensors, etc. • Devices, services, and contributions from different vendors and developers • Adopted technologies: IoT, data analytics tools, robotics, etc. • Farm's characteristics: activities to handle, size, location, etc.	[28, 32]
Interoperability & Integration	• Proper integration of heterogeneous components coming from different vendors and developers • Technical, syntactical, semantic, and organizational interoperability	[28, 31]
Harsh Environment	• Outdoor locations, presence of wild animals, bad weather, very high or very low temperatures • Farm devices have to be properly protected and encapsulated in water resistant enclosures	[28, 32]
Modularity & Scalability	• To simplify the introduction, removal, and update of new components and functionalities inside the FMS • Based on the concept of modules	[13, 32]
Standardization	• Development and usage of open standards • Adoption of a shared architectural model and open technologies	[31]
Easy-to-use	• For farmers, which may not have expertise in using digital technologies • For developers of new components	[32]
Plug-and-Play Solutions	• Installation of farm devices should requires few (or none) operations in order to make them work • Ideally, devices are physically installed in the deployment scenarios and start to work.	[32]
Power Source & Management	• Lack of a stable power source for farm devices • Devices need to work for years powered by batteries and/or power scavenging techniques • Device equipped with low-power modes • Usage of low-power communication protocols	[28, 31–33]
Security & Privacy	• Authentication, confidentiality, and access control	[28, 31–34]

Table 1.4: Main challenges to overcome in the deployment of IoT architectures targeting future smart farms.

IoT technologies, agricultural relevant data and scenarios); and scalability. In [33], issues and challenges for IoT and SA, which are more linked to network and open source software than those highlighted in [28, 32], are presented. Moreover, in [34] security and privacy issues targeting distributed SA-oriented cyber-physical systems are deeply studied, moreover, open research challenges and future directions for this subject investigated. In detail, according to [34], security challenges can be grouped into two main classes: (i) security and privacy issues, namely data security and privacy, authorization and trust, compliance and regulations, authentication, and secure communications; and (ii) potential cyber attacks (e.g., insider data leakage). In addition, in [31], an extensive and comprehensive review on open issues and challenges for generic IoT systems, from a technological point of view, is provided. A reduced set of the major open challenges highlighted by this last work (namely, standardization, interoperability, integration, scalability, power source and consumption, and security) and, thus, currently targeting IoT applications, are more thoroughly discussed in Sub-section 1.2.4.

As a final remark, an additional relevant aspect to consider, in order to enable a pervasive diffusion of smart farms, is related to farmers and their skeptical attitude towards the adoption of digital technologies to enhance their farms' management. Indeed, as an example, two of the main barriers, dissuading (especially small) European farmers to adopt new technologies, concerns: (i) the expensiveness of SA-oriented solutions; and (ii) the farmworkers' fear of not being able to learn how to use them [32, 35]. Governmental and institutional funding programs, financing the purchase of SA-oriented technologies and the technological training of farmers, are valuable solutions to overcome these limitations [35]. In the case of Italy, the trend is even more noteworthy [36–38]: as an example, the digital growth of the Italian agricultural sector is markedly slower compared with North Europe countries [35].

1.2 Internet of Things

1.2.1 Overview

The term IoT generally identifies a heterogeneous set of technologies, including hardware, software, and networking technologies, which enable real-world objects (i) to gather information from an environment; (ii) exchange them among each other and the Internet; and (iii) to exploit this information to build pervasive applications [28, 32, 39].

In detail, IoT-oriented systems or, equivalently, IoT architectures (namely, architectures built around IoT technologies) are usually composed of a heterogeneous set of physical devices, or Smart Objects (SOs), interacting with the surrounding environment and organized in local (or edge) networks. Thanks to, generally, both an available Internet Access Point (AP) and an intermediary device called gateway (GW), information coming from edge networks can be forwarded outside (and vice versa). More precisely, data are transmitted to entities (belonging to the same IoT system) which can be placed in the Cloud, e.g., Cloud platforms and analytical tools, as shown in Figure 1.2. Then, at Cloud-level, information are stored, visualized, analyzed, and eventually integrated with data retrieved from sources external to the IoT system.

Since the deployment of IoT architectures enhances the management of data relevant for an application, several fields of nowadays society (as home and building automation, Industry 4.0, and healthcare sector) can benefit from the adoption of IoT. As previously anticipated in Sub-section 1.1.3 and further discussed in Sub-section 1.3, SA is, definitely, included in this group.

1.2.2 Reference Models

The internal structure of an IoT system (i.e., its architecture) can be designed according to a reference architectural model. This, in general, allows to more clearly define the main building components of an IoT system, the interactions existing among these components, their functionalities, and the data flow of the overall system. Several reference models for IoT architectures have been proposed by not only scholars [40–43], but also standard institutes, e.g., the Telecommunication

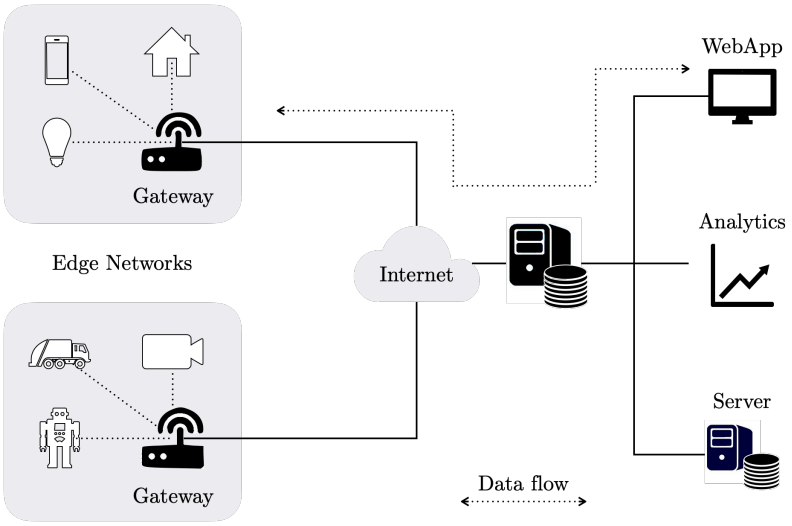


Figure 1.2: Main building components of an IoT system.

Standardization sector of International Telecommunication Union (ITU-T) [44], and associations, e.g., Alliance for Internet of Things Innovation WG03 (AIOTI) [45].

Generally, IoT reference models rely on a layered structure based on a variable number of vertically stacked layers (as models shown in Figure 1.3), ranging from three [28] to seven [41], according to the characteristics of the target application and its complexity. As an example, low-complexity applications, normally, lean on a reference model with a lower number of layers than those of more complex applications (e.g., with a higher number of entities to integrate and manage).

The most elementary IoT reference model, displayed in Figure 1.3a, is based on three layers. They are: (i) *perception layer*, composed of SOs sensing the surrounding environment and collecting data from it; (ii) *network layer*, enabling data exchange among devices and the Internet, eventually implementing local processing; and (iii) *application layer*, allowing end-users to access to the application's functionalities [28, 32, 43].

In the 4-layers IoT reference model, detailed in Figure 1.3b, an additional layer, namely, *MiddleWare (MW) layer*, is introduced (with respect to the 3-layers-based

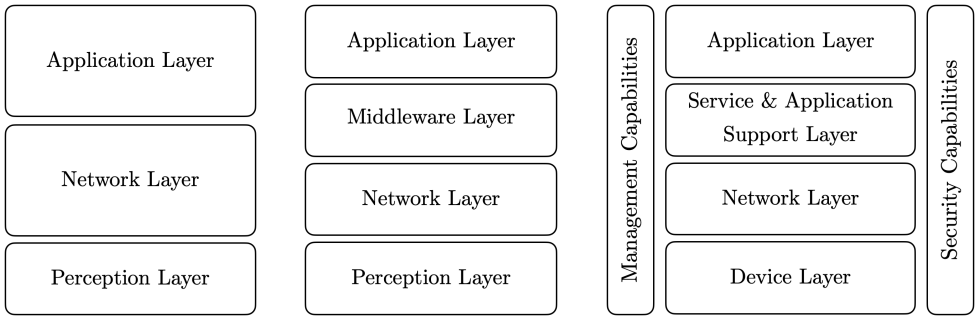


Figure 1.3: Three examples of possible IoT reference models: (a) 3-layers-based; (b) 4-layers-based; and (c) ITU-T Y.2060.

model) to emphasize the fact that the network layer and the application layer are decoupled. Indeed, the MW layer acts as an intermediary layer between the two, thus, seamlessly enabling and simplifying communications between physical devices (of the perception layer) and software applications (of the application layer), thus, hiding their intrinsic heterogeneity [46]. Moreover, the MW layer is usually in charge of managing information related to perception layer's devices, storing data coming from them, and made them straightforwardly available to application layer's entities. This allows to build modular and scalable IoT systems, since (i) SOs producing data, (ii) collected data, and (iii) software applications (or, in general, entities using these data) are all decoupled. A possible solution to simplify the design and deployment of a MW layer and the implementation of its functionalities, is relying on an IoT platform [42, 46] (as furthermore discussed in Sub-section 1.2.3). More details concerning main technologies and elements composing the four layers of the just described model are discussed in the following sub-section (Sub-section 1.2.3).

To conclude, a last relevant model is the *ITU-T Y.2060 IoT reference model* [44], provided by the ITU-T and shown in Figure 1.3c. As detailed by Figure 1.3c, the ITU-T Y.2060 IoT reference model is based on four vertical layers and two horizontal layers. More specifically, the functionalities implemented by the four vertical layers are similar to those described for the four layers composing the 4-layers-based

Device layer
• Device capabilities – Devices may be able to gather information and exchange them with the communication network directly or indirectly (namely, thanks to other devices, such as gateways) – Devices may support *sleep* and *active* modes and other mechanisms to optimize power consumption • Gateway capabilities – Support for multiple and diverse communication interfaces – Capabilities of converting different protocols
Network layer
Network and transport capabilities, such as mobility management; authentication; authorization and accounting (AAA); transport of IoT-related control and management information.
Service & Application Support layer
• Generic support capabilities, shared among different IoT applications, e.g., data processing and data storage • Specific support capabilities for a target IoT application
Application layer
The application layer is composed of end user software applications; data analytical; sophisticated processing; and forecasting tools.
Security capabilities
Proper management of security aspects at different layers of an IoT architecture, namely application, network, and device layer. Authorization; authentication; and data confidentiality at all the three layers; in addition: • at the application layer: integrity and privacy protection, and anti-visrus • at the network layer: signaling data confidentiality and integrity protection • at the device layer: device integrity validation, access control, and integrity protection
Management capabilities
Devices management; device firmware updates; local network topology; traffic and congestion management; accounting and performance management.

Table 1.5: Some examples of capabilities implemented by the layers of the ITU-T Y.2060 IoT reference model [44].

model shown in Figure 1.3b. Conversely, the two horizontal layers handle cross layers capabilities related to security and management. More details, concerning key functionalities and building components of the layers characterizing the ITU-T Y.2060 model, are provided in Table 1.5.

1.2.3 Main Technologies and Building Components

Main technologies and building components of a general IoT system, according to the 4-layers-based model displayed in Figure 1.3b, are concisely investigated in this sub-section. Moreover, technologies which are relevant for the deployment of IoT/SA-oriented systems are more deeply discussed in Chapter 2.

Physical Devices

The set of SOs which can be integrated within an IoT system is wide and heterogeneous. Indeed, it can include, for instance, daily life appliances, smartphones, vehicles, industrial machineries, microcontrollers, development boards, sensors, and actuators. Despite their great degree of heterogeneity, physical devices are, generally, equipped with (i) *computing capabilities*, or, namely, a Micro Controller Unit (MCU); (ii) *sensors* and *actuators*, to sense and act into the real world; (iii) a *communication interface*, which enables SOs to exchange information with other devices and the Internet network, using one or more communication protocols; and (iv) a *power source* [28, 39].

Micro-controllers and development boards (such as Arduino [47], Raspberry Pi (RPi) [48], and Pycom [49] boards) are attractive options to build devices which can be programmed according to the requirements of a specific IoT application. Indeed, they can be easily enriched with sensors/actuators and, frequently, are natively equipped with one or more network interfaces (thus allowing them to join an IoT network without requiring additional hardware to be integrated). In particular, since general purpose development boards allows to easily develop prototypical devices (or, simply, prototypes), they are popular solutions for both academic research projects and R&D commercial projects. This is notable because prototypes can be

used to perform feasibility studies aiming at understanding if an IoT system can be realized and, then, to more rapidly develop first versions of the same system. Indeed, since, usually, their realization helps to define the specifications of a real product and to highlight issues which may arise before and after its deployment, prototypes are frequently adopted as validation tools in multiple development's stages of commercial solutions (based on custom hardware, as IoT edge devices).

As a final remark, some examples of major challenges targeting SOs are related to their power consumption (which has to be efficiently managed especially when devices are battery-powered); their size; computing capabilities (which sometimes can be limited); and costs (which becomes more relevant while deploying or producing devices over a large scale) [50].

Wireless Sensor Networks

As said in Sub-section 1.2.1, physical devices at perception layer, usually equipped with sensors and actuators, are organized in edge networks. These networks, which are composed of spatially distributed nodes with sensing capabilities, can be, practically, defined as WSNs since they aim to monitor and record environmental conditions.

More specifically, in a WSN, a variable number of nodes, called SNs, potentially equipped with multiple sensors, measure environmental one or more variables (such as temperature, humidity, air quality, wind speed) and record them. Gathered data are, in general, forwarded to a collector, usually denoted as *sink* or *gateway*, which stores and can exchange them with entities outside the WSN.

More precisely, the sink connects the WSN with the external world, using a communication protocol which can differ from the protocol enabling SNs to exchange information within the WSN. Indeed, communications between SNs and the rest of the IoT system's components are frequently enabled by the integration of long-range (such as, cellular LTE, Sigfox, and LoRaWAN) and short/medium-range (e.g., Bluetooth Low Energy (BLE) or IEEE 802.11) networking technologies.

IoT Gateways

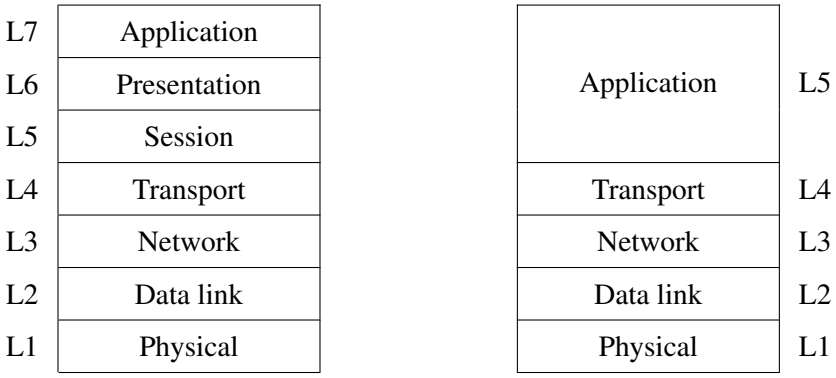
In an IoT application, an IoT GW implements some functionalities typically provided by a sink node of a WSN. This means that IoT GWs allows physical devices, belonging to an edge network, to join another network, such as the Internet, thus, enabling information exchange between them and the Cloud, as displayed in Figure 1.2 [31]. Moreover, GWs can provide support for multiple and diverse communication interfaces, and for translation of packets coming from edge devices or the Cloud among different communication protocols (as outlined in Table 1.5). To conclude, since they are usually more capable than other edge devices (e.g., in terms of power supply, computing, and storage resources), they can process received messages with, for instance, data fusion (or other pre-processing) techniques, together with providing other "smart" capabilities.

Network Layer

Networking technologies, such as communication protocols and infrastructures, are key components in enabling data flow among different entities of the same IoT system. More precisely, information exchange within an IoT system are usually built around one or more communication protocol stacks (in other words, groups of communication protocols), according to the International Organization for Standardization Open Systems Interconnection (ISO/OSI) model [51], shown in Figure 1.4a.

For what concern the protocol stack enabling edge devices to forward messages to the Cloud and vice versa, relying on a protocol suite with seven layers (i.e., classic ISO/OSI model) is not always a feasible solution. Indeed, since edge devices are usually constrained, meaning that they cannot support a full ISO/OSI protocol stack, the number of model's layers has to be reduced from seven to, for example, five, as described in Figure 1.4b.

Some examples of popular communication protocols, to include in the IoT protocol stack shown in Figure 1.4b, are presented in the following. A selected set of such networking technologies which are promising for SA applications are more



(a) ISO/OSI model. (b) IoT protocol stack.

Figure 1.4: IoT protocol stack: the ISO/OSI stack model (a) can be reduced to five layers (b).

deeply discussed in Sub-section 2.3.1. Wireless communication protocols widely adopted in IoT applications, defining specifications for the first two layers of the ISO/OSI model, significantly differ in terms of transmission range, data rate, power requirements, and maximum amount of data transferred within a single packet (or payload size). A few representative examples of these protocols are listed in Table 1.6, together with their operating frequency, power, maximum transmission range, and data rate [28].

Regarding the application layer (which is the higher level of both the stacks presented in Figure 1.4), a few representative examples of IoT protocols [52, 53] are: Hypertext Transfer Protocol (HTTP) [54], Message Queuing Telemetry Transport (MQTT) [55], Data Distribution Service (DDS) [56], Extensible Messaging and Presence Protocol (XMPP) [57], Advanced Message Queuing Protocol (AMQP) [58], and Constrained Application Protocol (CoAP) [59].

Protocol	Operating frequency	Max range	Max data rate	Power [mW]
RFID	13.56 MHz	1 m	423 kbps	1
Zigbee	2400-2483.5 MHz	10 m	250 kbps	1
WiFi	2.4, 3.6, 5 GHz	100 m	6-780 Mbps, 6.75 Gbps at 60 GHz	1
BLE	2400-2483.5 MHz	100 m	1 Mbit/s	10–500
6LowPAN	908.42 MHz, 2400-2483.5 MHz	100 m	250 kbps	1
Sigfox	908.42 MHz	30-50 km	10-1000 bps	N/A
LoRaWAN	Various	2-15 km	0.3-50 kbps	N/A
NB-IoT	200 kHz	10 km	250 kbps	N/A
LTE	700-2600 MHz	28-10 km	0.1-1 Gbps	5000-1000

Table 1.6: Example of popular wireless technologies adopted in IoT applications.

MW & IoT Platforms

As anticipated, a valuable solution to simply the integration of different devices and software components within the same IoT system, moreover, enabling data flow among them, is to lean on an intermediary entity called MW. Furthermore, the implementation of such layer can rely on an IoT platform. Indeed, although IoT platforms are usually less flexible compared to custom-developed MW (i.e., which are more customizable, developers can better control them), their usage relieve developers from building a MW starting from scratch, which is usually a very complex and time-consuming operation.

IoT platforms are software suites, or cloud services, namely, Platform as a Service (PaaS), providing developers an ecosystem and infrastructure capabilities which allow to rapidly and straightforwardly integrate devices, networks, and applications within a single IoT system. An IoT platform has to ideally satisfy an extensive set of functional requirements (e.g., resource, data, event, and code management) and non-functional requirements, such as scalability, interoperability, and real-time responsiveness, just to name few examples [46]. The range of possible IoT platforms to be selected is wide and heterogeneous, in terms of implemented functionalities

and satisfied requirements. Despite their diversity, a possible classification for IoT platforms is provided by [46], which categorizes them in three groups, according to their main goals: (i) device management-oriented; (ii) application development-oriented; and (iii) application enablement-oriented platforms.

Moreover, platforms available on the market can be open-source or commercial. The first type of platforms, usually, benefit from contributions coming from communities of developers. Moreover, they offer free plans which, as drawback, provide fewer guarantees on Quality of Service (QoS). Indeed, for instance, open-source platforms are generally less stable than commercial ones. On the other hand, although their functionalities are less easily customizable, proprietary platforms are normally better options while developing commercial IoT applications which have to fulfill specific requirements, e.g., in terms of robustness and reliability. Conversely, open-source platforms are suitable choices for research and prototypical projects (needing a fast development) and, in general, applications with less restrictive constrains on service quality.

Some examples of commercial IoT platforms are Google Cloud IoT [60], Amazon IoT platform [61], IBM Watson IoT [62], and Microsoft Azure IoT Hub [63]. On the other hand, for what concern open-source platforms, they include OpenIoT [64], Sitewhere [65], OpenRemote [66], and Thinger.io [67].

A relevant open-source framework (and project), to adopt to build an IoT MW, is FIWARE [68]. The FIWARE project aims at simplifying and speeding up the development of interoperable smart solutions, able to retrieve and manage “context data” of an application (namely, data which are relevant for the application) coming from different data sources (ranging from IoT devices to legacy systems) and using standard technologies.

A “Powered by FIWARE” platform is based on a core entity, called FIWARE Context Broker, which is enriched with additional modules (namely, Generic Enablers) to implement the specific requirements of a solution. More precisely, the Context Broker is in charge of making accessible to the application the current status of context information and to update it. On the other side, additional modules are integrated to support, for example, data persistence (using different storage

technologies, e.g., relational or non-relational databases); the gathering of data coming from heterogeneous data source, as IoT devices exchanging information with diverse communication protocols (thanks to IoT Agents) or as social networks (e.g., for Twitter, cygnus-twitter); and the development of highly customizable operational dashboards (i.e., WireCloud Generic Enabler).

As anticipated, together with its main purpose of being a context management framework, FIWARE is a promising technology to build IoT MWs. Indeed, a reduced set of software entities (modules), developed by the FIWARE community and implementing functionalities commonly provided by IoT MWs (as data storage), can be integrated to build a custom IoT platform. Moreover, additional modules (Enablers) can be embedded within the resulting MW, thus, customizing it in a very flexible way. The effort (and time) needed to build a powered by FIWARE IoT MW is probably higher than developing the same MW relying on a ready-to-use IoT platform, but lower than creating a custom MW starting from scratch. Moreover, with respect to IoT platforms, a FIWARE-based application is more flexible towards the introduction of new features; conversely, it can reach higher degrees of openness than custom solutions.

Storage Technologies

Storage technologies (such as databases) are central components of IoT systems, indeed, they allow to store data collected from sensors, creating historical repositories, and to make such data accessible. Moreover, storage technologies can be deployed on edge devices, as local repositories of information, or in the Cloud, where bigger and more complex bulk of data can be generally stored. Different database technologies can be integrated in the same IoT application to properly manage data differing considerably, in terms of, for instance, data source, and to satisfy specific requirements (e.g., data availability and consistence).

Along with well-established relational (or Structured Query Language, SQL) databases, the need to develop storage technologies better satisfying nowadays applications' requirements has led to the birth of the so-called non-relation databases (in other terms, database technologies not storing data in the form of tables, columns,

and rows). Indeed, the volume, the unstructured nature, highly variability and mutability of certain applications' data, together with the need of building high-responsive and distributed databases, lead to the adoption of storage technologies which, with respect to relational databases, are more flexible, dynamic, and efficient to retrieve data. The aforementioned type of applications definitely include IoT solutions. Indeed, IoT applications are, usually, characterized by heterogeneous, mutable, semi or unstructured data, which may vary significantly in terms of typology, size, and source of origin [69, 70].

Considering the type of approach followed to store data within the database (i.e., the storage methodology), the main difference between relational and non-relational technologies is that the firsts store data using tables, columns, and rows, conversely, the seconds exploit alternative approaches. Moreover, non-relational (or noSQL) databases can be classified according to the adopted storage methodology in different categories. Some examples of the most used non-relational technologies, together with the adopted storage methodology and possible application fields are reported in Table 1.7.

Distributed Computing Model

From a computing perspective, an IoT system is a distributed system. Indeed, it is composed by spatially spread entities (e.g., edge devices, gateways, Cloud platforms, web applications) equipped with storage, computing and networking capabilities, which cooperate, communicate among each others, and act as a single system. Moreover, each component of an IoT system, placed in the Cloud or in a real scenario, can produce, elaborate, and consume data, according to the owned capabilities. In the field of IoT, data processing made in the Cloud (or, generally, in a remote location, far from the source which has generated data) is usually denoted as Cloud computing. Alternatively, data processing performed by devices belonging to an IoT edge network (which is, usually, spatially close to the origin of information) is defined as edge computing.

Defining how to distribute computing between Cloud entities and edge networks (in other terms, how to integrate both Cloud and edge computing within the same

Typology	Storage methodology	Possible application fields	Examples
Key-value	Data are stored as key-value pairs; values are retrieved using keys.	In-memory data caching; real-time recommendations and advertising; storage of user session details and preference in web applications.	Redis, MemCached
Document	Information are stored as documents. In documents, which are usually organized as key-pair values, can also be stored metadata.	Storage of big volumes of semi- or un-structured data; collection and aggregation of context information; real-time Big Data storage and retrieval.	MongoDB, CosmosDB, CouchDB
Graph	Data are organized as graphs: nodes represent entities and edges describe relations between two or more entities.	Applications which data are characterized by complex or many relationships, e.g., social networks, fraud detection, network mapping; fast and simple retrieval of correlated data.	Neo4j, Ontotext GraphDB
Time-series	Data are stored with a time-based reference (i.e., as time-series).	Applications which data change over time (with a strong time dependence, namely, time-series), e.g., sensor and machinery data, including predictive maintenance, environmental and physiological parameters monitoring.	InfluxDB, Prometheus, TimeScale

Table 1.7: A few representative examples of non-relational technologies.

application) is a key aspect to consider, since it can enhance the overall performance of an IoT application. Indeed, although resource-intensive operations (such as, the execution of complex AI algorithms) are, nowadays, mainly performed in the Cloud, where extensive storage and computing capabilities are available, the execution of less resource-demanding tasks can be delegated to edge devices, which are, generally, less capable than the Cloud. This means that, for instance, edge computing allows to reduce the volume of data forwarded by edge devices to the Cloud (e.g., data coming from edge devices are fused or aggregated before being forwarded over the Internet). Moreover, this leverages the network load, decreases the response time of the overall system, and prevents private data (e.g., the habits of a smart home householder) from being transmitted over the Internet (i.e., to the Cloud) [71].

1.2.4 Open Challenges

There exist several challenges while deploying IoT systems, some of which are collected in Table 1.4. Challenges related to standardization, interoperability and integration, scalability, power source and consumption, and security and privacy are detailed in the following paragraphs [31].

Standardization. As said, the proper integration of heterogeneous technologies, products, and services (coming from different vendors) inside the same system is one of the main obstacles to overcame while deploying IoT applications. One of the major solution to the problem is the adoption of open standards, which follow publicly available and shared "rules." Moreover, the usage of open standards prevents applications from being bounded to a specific vendor or technology (a key aspect for IoT systems).

Interoperability and Integration. IoT system's heterogeneity poses challenges also in terms of interoperability, indeed, it makes cooperation among different devices more complex. Interoperability can be of different typology, for instance, IoT *technical interoperability* is generally related to the characteristics of the adopted networking infrastructure. Indeed, this is often composed of multiple networks of edge devices, which exchange information using sets of diverse communication protocols. Implementation of APIs, usage of gateways, IoT platforms, and Software-Defined Networking (SDN) can solve issues related to technical interoperability [31].

Scalability. A desirable property of an IoT system is scalability, which enables new components (e.g., devices, services) to be added to the system without degrading the existent performance. When the number of devices, characterized by constrains of memory, computing, and other resource capabilities, increases, scalability becomes more challenging. Layered frameworks and architectures (as the models presented in Sub-section 1.2.2) are two possible solutions to support scalability (and interoperability) within IoT systems.

Power Source and Consumption. To stay active, edge devices has to be powered by a power source such as, a cable connection to a power outlet, a battery, or a power scavenging technique (in other words, the device gathers energy from the surrounding environment, e.g., from solar light). Although designing edge devices to efficiently use the available energy is commonly a good practice, it becomes crucial when devices are powered by constrained power sources, such as batteries. Indeed, since the energy consumed by an IoT device is influenced by several factors, it has to be accurately managed. Some examples of these factors include the energy required by the device to perform computation, send data, and sleep; the protocols adopted to communicate with other edge nodes; and the program ran on the device. Adopting low-power communication protocols, selecting boards having natively provided low-power modes, and reducing the time during which a device is active (and, thus, is not sleeping), can considerably decrease the power daily consumed by a device.

Security and Privacy. The huge heterogeneity and distributed nature characterizing an IoT system, which building components have to hold, get access and exchange (often sensible and proprietary) information in a secure way, leads to remarkable challenges in terms of security and privacy management. Indeed, secure mechanisms (such as authentication and access control) should be implemented at every layer of an IoT architecture. This to avoid threats directed towards, just to name few examples, network communications (e.g., Denial-of-Service, man-in-the-middle attack, communications sniffing), physical devices (e.g., malicious code injection), and software components (e.g., sensible data breach and violation of privacy regulations). The introduction of data encryption algorithms and application layer security protocols, which are lightweight enough to be executed by IoT devices with constrained capabilities; appropriate authorization, authentication, data access and intrusion detection mechanisms; and suitable key distribution policies are some possible solutions to improve security of IoT systems.

1.3 IoT/SA-oriented Architectures

1.3.1 Literature Review

As already said, IoT systems can enhance several agricultural processes related to, for instance, crop, greenhouses, and livestock production [28, 32, 72]. Moreover, the set of technologies to adopt to develop an IoT architecture for SA applications is wide. Due to this, a brief (not exhaustive) literature review of 12 representative papers presenting each an IoT/SA-oriented system, aiming at highlighting the potential of IoT for SA applications and their heterogeneity, is provided in the following paragraphs. More precisely, first, the selected works are briefly introduced, together with their main purposes; secondly, the main technologies adopted to implement them are discussed. To conclude, in Table 1.8, selected works are summarized in terms of target applications and used technologies.

Covered Activities

For what concerns crop cultivation, in [3] an IoT system aiming at collecting environmental data (i.e., air temperature and humidity, wind speed, and rain fall) and predicting their future values to prevent the born and diffusion of fungal diseases on crop fields, is proposed. In [4] an IoT architecture, based on open hardware, intending to prevent mildew disease of vineyard, is presented. In [5], a soil monitoring system, collecting and storing information related to soil temperature, EC, soil moisture, and CO^2 content, is proposed.

In the field of greenhouse production, an IoT system aiming at collecting greenhouses' environmental parameters (namely, air temperature, pressure and humidity, and light) and displaying them using a smartphone or a computer, is presented in [6]. In [7], IoT and computer vision algorithms are integrated to build a system able to monitor the growth of leaves of a *Phalaenopsis* orchid. In [8], an IoT system aiming at supporting the monitoring of relevant parameters for aquaponics

Ref.	Field	Farm device	Wireless technology	Middleware	Data visualization	Advance data processing
[3]	Crop (environmental) monitoring	Custom-made board (pic32mx250f128b)	WiFi & Zigbee	Sparkfun Data (Sparkfun Cloud storage service)	Web-based dashboard	Support Vector Machine (SVM) regression to forecast future environmental variables trend
[4]	Crop (environmental) monitoring	Development board (LinkIt One)	GPRS (SMS)	Ad-hoc	Mobile App, web-based and desktop dashboard	Goidanich model to detect downy mildew disease
[5]	Crop (soil) monitoring	Custom-made board (ATmega2560)	LoRaWAN	ChirpStack Network and Application Server	Web-based dashboard	None
[6]	Greenhouse environmental monitoring	Development board (MicaZ mote)	ZigBee	MIB 250 Service Support Platform	Mobile App, computer application	None
[7]	Greenhouse leaf growth detection and analysis	Development board (Octopus II)	Zigbee	Ad-hoc (Host Analysis Platform)	N/A	Image processing algorithm to estimate Phalaenopsis leaf size and discover correlation between greenhouse environmental variables and leaves' growth
[8]	Aquaponics environmental variables monitoring and control (greenhouse)	Development board (Arduino Mega)	N/A	Ad-hoc	Mobile App	None
[9]	Livestock living environment monitoring	Smart sensor (Texas Instruments Sensor Tag)	BLE	Ad-hoc	Web-based dashboard, mobile App	Decision Tree and Sliding Window Streaming Analytics for diary production optimization
[10]	Beehive (internal environmental condition) monitoring	Custom-made board (ATmega1281)	Zigbee & 3G/GSM	Ad-hoc	None	Decison tree-based classification of beehive environmental conditions and for weather forecasting
[11]	Animal GPS position tracking	Development board (Libelium WASPmotes)	LoRaWAN	Ad-hoc	Web application	None

Table 1.8: Development of IoT systems targeting SA applications: 12 selected representative papers found in literature.

production within a temperature-controlled greenhouse (i.e., light, water pH, and temperature) and to discover the correlation among them is proposed.

In the context of livestock animals production, a possible application for IoT systems is the monitoring of environmental variables characterizing the living space of animals, which include, for example, Heat Stress (HS) for cows [9], or gases, temperature, and relative humidity for bees [10]. Another possible application in this field is the tracking of animal positioning using Global Positioning System (GPS). For instance, in [11] a cattle's monitoring system named CowTrack, displaying animals' positions with a web application, is presented.

Adopted IoT Technologies

The IoT systems presented in the previous sub-section are based on similar building components, according to the architectural model presented in Subsection 1.2.2. More exactly, the components include devices installed in the farm to collect data (or farm devices, e.g., smart collars, development boards equipped with sensors), a network infrastructure to enable communications and to forward data coming from devices to the Cloud, where data are stored, visualized, and, eventually, processed. Furthermore, from the perspective of the used technologies, the discussed works considerably differ in terms of selected farm devices' hardware, wireless communication protocols enabling farm communications (namely, communications among farm devices and the Cloud), middleware platforms, data visualization and data analysis technologies, as can be derived from Table 1.8.

Indeed, considering farm devices, the solutions listed in Table 1.8 rely on the usage of (i) development boards equipped with sensors and/or actuators (namely, LinkIt One [4], MicaZmote [5], Octopus II [6], Arduino [7], and Libelium WASPmotes [11]); (ii) custom-made boards, which are assembled starting from an MCU (i.e., pic32mx250f128b [3], ATmega 2560 [5], and ATmega1281 [10]); or (iii) smart sensors, in other words, small intelligent devices based on one or more sensors integrated with the electrical circuitry needed to read and share sensor data with other devices (namely, Texas Instruments Sensor Tag [9]).

Moreover, wireless communications can lean on technologies having different

transmission range: (i) low-range, as BLE [9]; (ii) medium-range, such as, IEEE 802.11 (Wi-Fi) and IEEE 802.15.4 (Zigbee) [3, 5–7]; and (iii) long-range, i.e., cellular technologies (GPRS and 3G/4G/5G) [4, 10], Sigfox, and LoRaWAN [5, 11]. The great heterogeneity of the presented wireless technologies is justified by the need of selecting the most appropriate type of connectivity to use considering the characteristics of the agricultural scenario to manage, which include, for instance, its dimension and the availability of an Internet AP. As an example, IoT systems targeting the environmental monitoring of open field crops and enabling the localization of livestock animals over a wide area of space usually exploit long-range wireless power-efficient communication protocols (namely, LPWANs), such as LoRaWAN [73], Sigfox (e.g., for cow geo-localization [11]), and Narrowband IoT (NB-IoT) [74].

The MWs of the systems collected in Table 1.8 have been implemented with the usage of an existent IoT platform [3, 5, 6] or developing an ad-hoc solution to store, maintain, and made available data coming from farms [4, 7–11]. Furthermore, the visualization of collected data and the provision of functionalities to end-users are implemented with the usage of web-based dashboards [3–5, 9], web-based applications [11], computer applications [4, 6], and/or mobile Apps [4, 6, 8, 9].

To conclude, gathered data can be processed to build data analysis algorithms (e.g., based on AI) to support crop, soil, disease, weed, and livestock management [75]. Some possible applications in these fields include: forecasting future value of variables relevant for agricultural production [3, 7, 10]; forecasting the occurrence of a disease [4]; and optimizing diary production [4], the growth of plants [7], and the environmental condition of animals' living space [10].

1.3.2 Research Gaps

The IoT systems presented in Table 1.8 aim to enhance the management of a specific SA scenario, or, in other words, a reduced set of agricultural activities, but future farms will be characterized by multiple scenarios and diverse activities to handle (as said in Sub-section 1.1.4). This means that the development of IoT systems able to manage multiple scenarios and activities of farms, in a straightforward and

efficient way, is a key step in the process of agricultural digitization, which, however, has not being already taken.

Furthermore, given that agricultural scenarios differ considerably from each others, different hardware and network technologies have to be selected, considering the specific characteristics of a target farm, and integrated within the same FMS. The above-mentioned notable challenges, namely, heterogeneity integration, abstraction, and development of a modular, scalable, and flexible IoT architecture (together with the additional ones collected in Table 1.4), have not being effectively addressed for SA applications. In other words, to the best of the knowledge of this doctoral thesis's author, an IoT/SA-oriented system satisfying the presented requirements has not been proposed.

The aforementioned research gap is addressed by this doctoral thesis as follows. *First*, an architectural model to rely on to develop general-purpose IoT/SA-oriented systems, based on both a solid theoretical basis (i.e., literature study and analysis) and practical experience (acquired from developing real IoT systems targeting SA scenarios), has not being already proposed in the literature. This research gap is covered in Chapter 2, conceptualizing the above described type of framework. *Second*, the validation of an IoT reference framework is, often, experimentally undertaken developing a single IoT system (based on it), targeting a specific use case (or, sometime, without any use case). However, in order to be as general as possible, a framework needs to be ideally validated on multiple and heterogeneous scenarios. This gap is covered in Chapter 3, validating the proposed framework, adopting it to develop two real IoT/SA-oriented systems (namely, VegIoT Garden [12] and LoRaFarM [13]) targeting two very different SA scenarios (i.e., small-sized crops and wider generic farms). *Finally*, the EdgeAI trend to develop constrained device-friendly AI algorithms has not being investigated in the context of SA: this gap is covered in Chapter 3, presenting two EdgeAI-based approaches aiming at enhancing a greenhouse's management.

1.4 Artificial Intelligence

1.4.1 Overview

As anticipated, IoT systems allow to gather, fuse, process, and store huge amount of data (i.e., Big Data) coming from heterogeneous data sources (including physical devices and external repositories), which are relevant for an application, and to exploit these data to provide some functionalities to the system end-users.

The creation of new value starting from raw information (held by an IoT system), elaborating and extracting knowledge from them, can be successfully achieved using AI-based algorithms [76]. Indeed, an AI-powered (IoT) system is able to understand which data are relevant, can learn from them, acquire new knowledge, and exploit learnings to accomplish specific tasks, following a flexible and adaptable approach [77].

In particular, Machine Learning (ML) algorithms (a sub set of AI algorithms learning from data, identifying patterns, and taking decisions without or with small human involvement) the development of data analysis and forecasting models can be made automatic. In this field, one of the most popular and versatile technologies are NNs, which are briefly introduced in Sub-section 1.4. Due to their interesting features, NNs have been selected as ML tools to develop the forecasting algorithms presented in Chapter 4.

As expected, the joint adoption of AI-based algorithms and IoT technologies provides several advantages; this is equally valid in the context of SA. As an example, greenhouse production is one of the several fields of SA which be improved by these technologies, as introduced in Sub-section 1.4.3 (and further discussed in Sub-section 4.3). Furthermore, despite in the last decades the execution of AI-based algorithms have been principally performed in the Cloud, the need to relocate intelligence and processing tasks (e.g., run an AI algorithm) from the Cloud to the edge (for the reasons explained in Sub-section 1.2.3) has led to the rise of a new promising trend, for both IoT and AI. This trend is usually labelled as *EdgeAI* [78]

and is deeply discussed in Sub-section 1.4.4.

1.4.2 Neural Networks

NNs are computing models aiming at building ML algorithms able to solve several tasks, as prediction and classification [79]. Indeed, starting from a set of data called *data set*, NNs are able to fit the provided data into a parametric model [80–82]. In detail, model's parameters are learnt from a set of samples, labeled as *training set*, in the algorithm training phase, while their validity is investigated on a *test set* of samples in a (successive) test phase, adopting one or more evaluation metrics.

The usage of NN-based models allows to (i) predict the values of output variables approximately in real time; (ii) to reveal hidden relations among data; and (iii) to build a solution which is robust against data uncertainty. For these reasons, NNs are attractive choices for many application fields, including SA. As a drawback, commonly, the number of available data has to be sufficiently large (although gather the required data is sometimes difficult) and data need to be representative to build a precise model.

A simple type of NN-based architecture, among the several ones which can be found in the literature, differing in terms of building elements, interconnections, and learning algorithms, is the Multi-Layer Perceptron (MLP)-based NN [83] (denoted as ANN, for shortness, in the following). In particular, the model is based on a set of many and connected processing units, called *neurons* (or, for simplicity, network nodes), organized in *layers*. Due to the internal arrangement, input data are processed from the first layers of the network to the last ones, enabling information to get through the network before generating output variables.

In *feed-forward* NN models, data are linearly elaborated within the network from the first layer (or input layer) across one or more internal layers (or hidden layers), up to the final layer (or output layer). Moreover, in *fully-connected* NN, each neuron of a layer is linked with every node of the previous layer. Although a comprehensive description concerning the process of training a NN (from a mathematical view point) is out of this doctoral thesis' scope, some high-level considerations, in particular

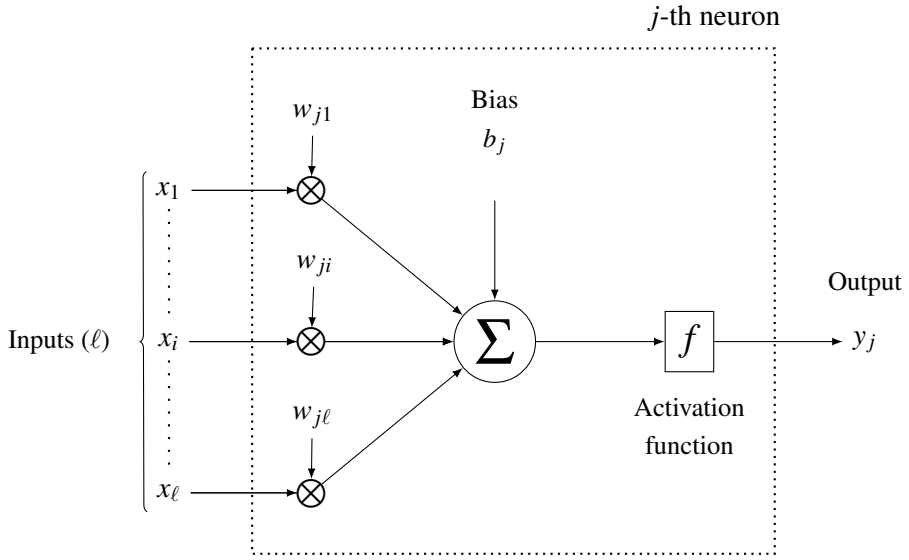


Figure 1.5: A simplified version of the mathematical model which is implemented by the j-th neuron of a MLP ANN.

related to the training process of a fully-connected feed-forward ANN, are presented in the following paragraphs.

Each network's neuron (apart from neurons in the first layer) receives data from all neurons in the previous layer (or, if the ANN is not fully-connected, from a reduced set of neurons). A simplified version of the mathematical model on which is based the j-th neuron of an hidden layer of a MLP ANN is shown in Figure 1.5. Input data coming from the ℓ neurons connected to the j-th neuron and belonging to the previous layer, which are labeled as $\{x_i\}_{i=1}^{\ell}$, are aggregated within the j-th neuron using a suitable parametric function. One of the widest adopted functions for the scope is based on a weighed sum of j-th neuron's input, with weights $\{w_{ji}\}_{i=1}^{\ell}$. More precisely, each input x_i of the j-th neuron is multiplied by a weight w_{ji}, the value of which is learned during the ANN training phase (and, therefore, not a-priori fixed) and, then, summed to the other weighed inputs data. A *bias* term b_j is, then, added to

the sum. To conclude, before being transmitted to the following layer in the network, the output of the j-th neuron is passed to a non-linear *activation function*, denoted as $f(\cdot)$. As an example, a commonly adopted activation function is the Rectified Linear Unit (ReLU) function $f(x) = \max(0,x)$ [81]. The whole output of the j-th neuron, labeled as y_j, can be summarized as follows:

$$y_j = f\left(\sum_{i=1}^{\ell} w_{ji} x_i + b_j\right). \tag{1.1}$$

The training phase of a NN is generally achieved performing multiple stages in which, based on the values of a cost function, aiming at numerically quantifying the similarity between the network's outputs and its expected values, the weights (namely, the model's parameters) are updated to improve the model performance (more exactly, to minimize the selected cost function). The set of algorithms which can be used to perform the learning process is wide: the most known ones among them is Back-Propagation (BP) with gradient descent [79].

1.4.3 Greenhouse Air Temperature Forecasting based on NNs

As more deeply discussed in Chapter 4, the management of several agricultural scenarios (including controlled environments, as greenhouses) can be enhanced by the usage of AI algorithms (as NNs) and data analysis tools. In particular, for what concern greenhouse production, one of the crucial activities which, according to literature, can considerably benefit from AI, is the regulation of a greenhouse internal environmental variables, such as air temperature and humidity (as more exhaustively explained in Sub-section 4.2). As an example, the tight control of a greenhouse internal air temperature has been effectively achieved through the deployment of NN-based algorithms aiming to predict the future trend of the internal air temperatures.

With respect to this topic, relevant literature papers in the field of air temperature forecasting inside greenhouses using NNs are summarized in Table 1.9. Despite models listed in Table 1.9 aim to achieve the same prediction goal using NNs, they differ in terms of adopted (i) input variables; (ii) NN architecture; (iii) design approaches; and (iv) data sources.

Ref.	NN Model			Performances (on test set)			Data set details		
	Input Variables	**Architectural type**	**Training algorithm**	**RMSE** (°C)	**Mean Absolute Percentage Error (MAPE) (%)**	**Coefficient of determination (R^2)**	**Size (samples no)**	**Collection interval**	**Sampling interval**
[84]	External temperature and solar radiation, wind speed, heater temperature, datetime reference	ANN	BP, CGA	2.5–3.0	N/A	N/A	1368	$\approx$ 2 months	1 hour
[85]	Internal solar radiation, air temperature and humidity, and soil moisture, CO_2, atmospheric pressure, datetime reference	ANN	BP	0.839	N/A	0.977	$\approx$ 87408	19 months	10 min
[86]	External solar radiation and temperature, wind speed	ANN, RBF	BP	0.20 ± 0.02, 0.13 ± 0.01	0.93 ± 0.10, 0.59 ± 0.07	0.76 ± 0.05, 0.89 ± 0.03	N/A	N/A	N/A
[87]	External solar radiation, heater temperature, internal air temperature and humidity, wind speed, history of actuators, shadow screen	RBF	BP, LM	0.0019	N/A	N/A	1728	12 days	10 min
[88]	External temperature, solar radiation and humidity, wind speed and direction, history of actuators	ANN, RNN-Long Short-Term Memory (LSTM), NARX	BP	0.89–0.94, 0.45–0.71, 0.52–1.32	N/A	0.94, 0.96–0.97, 0.86–0.96	$\approx$ 470000	1 year	5, 10, 15, 20, 25, 30 min
[89]	Internal air and soil temperature, internal solar radiation, humidity and CO_2	RNN	BP	0.865	1.7	0.925	1152	8 days	10 min

Table 1.9: Representative literature papers in the context of air temperature forecasting inside greenhouses using NNs.

Input Variables. Different type of variables can be selected as input parameters for the forecasting models. They can be related to (i) the surrounding environment external to the greenhouse; (ii) the internal climate of the greenhouse; or (ii) a mix of the firsts two. Although the *first* type of variables cannot be, generally, controlled, they impact on the inner habitat of the greenhouse. Some examples of variables belonging to this group are the external air temperature, humidity, solar radiation, and wind speed [84–86, 89]. Conversely, the *second* group of variables can, instead, usually be controlled by activating (or deactivating) some actuators installed within the greenhouse. The internal air temperature and humidity, soil moisture, and CO_2 are some examples of these variables [85, 87, 88].

NN Architectural Type. According to literature papers in Table 1.9, good prediction performance (in the task of greenhouse air temperature forecasting) are achieved adopting the following NN architectures: Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs), LSTMs and Radial Basis Function (RBF) networks.

Design Approach. Based on their design principles, the proposed models can be conceptually divided in (i) time series-oriented; (ii) "pure" ML; and (iii) hybrid. More precisely, the *first* type of approaches addresses the air temperature forecasting task as a time series forecasting problem. More exactly, the following features characterizing time series data, namely, regularly sampled data which have a time reference (as sensor data), are exploited by this approach: seasonality, trends, and correlation between samples which are near in time. Indeed, since NN models, such as RNNs [89] and LSTM networks [88], can successfully discover relations existing between temporally close data which, usually, characterize time series, they can reach prediction performance which are higher than other type of NN architectures. Besides the above-mentioned technique, another valuable option to forecast air temperature is to exploit data coming from different data sources, without considering the relation existing between temporally close data of the same time series. These sources can include data related to internal and external variables of a greenhouse, which

have some kind of correlation with the air temperature values to be forecast. RBF networks [86, 87] and ANNs [90] have accomplished the tasks of air temperature forecasting following this approach. To conclude, the two approaches can be jointly adopted, leading to an hybrid approach; then, the NN architecture, which best suits the forecasting problem, is accordingly selected [84].

1.4.4 Embedding AI at the Edge

Although in the last decade, AI-based algorithms, executed in the Cloud, have been successfully integrated with IoT technologies, empowering applications in several scenarios (including SA, as said in Sub-section 1.3.1), the trend of EdgeAI is moving the execution of AI algorithms from the Cloud to the edge of a network [78]. Indeed, there exists several advantages in running AI algorithms at the edge, in other words, near the source which generates data. They include, for instance, the reduction of the volume of data to be sent to the Cloud by edge devices, thus lowering the network load and the response latency, and, the support for scalability. Moreover, the execution of algorithms at the edge can solve possible issues related to (i) connectivity, where an existing, robust, and stable connection is not available; and (ii) privacy, e.g., sensitive data, collected and processed locally, are not forwarded to the Cloud as they have been gathered [78]. It is clear that, although network-related issues will probably become less relevant with the rise of 5G [91], more and more IoT applications, including IoT/SA-oriented systems, will benefit by adoption of EdgeAI.

One of the main challenges while developing EdgeAI algorithms is related to the hardware capabilities of constrained edge devices. In fact, these devices have generally a memory footprint, computational, and power resources remarkably lower than those provided by Cloud platforms. This means that EdgeAI algorithms needs to be computationally "lightweight enough" to be run by constrained devices (e.g., the memory required to store and run the model has to be consistent with the one available on the device).

Two main classes of strategies can be adopted to make possible the deployment of AI algorithms on edge devices. The first is to equip constrained devices with hardware especially designed to execute resource-intensive AI algorithms. These techniques

include, for instance, hardware accelerators and the usage of boards having built-in AI dedicated chips (e.g., MediaTek Inc. [78], Google tensor processing unit). The second class intends to decrease AI models complexities, making them "more lightweight" to be run by an edge device with constrained capabilities (e.g., reducing the architectural and computational complexity of the model, namely, lowering the number of its parameters and Multiply–ACcumulate (MAC) operations, respectively) [92]. Reducing model complexity, in other words, implementing a model compression, based on different techniques, such as pruning and weights quantization, usually results in having a less accurate model, in terms of prediction performances [92, 93].

Due to this, building a model which has good prediction performance but is also computationally lightweight enough (e.g., the required memory occupation has to be compatible with the one available on the device running the algorithm), becomes a relevant challenge in developing EdgeAI algorithms. In this field, an interesting framework developed by STMicroelectronics, aiming at simplifying the development of AI models on edge devices, making automatic the process of model compression and deployment on prototypical boards, is the STM32Cube.AI [94].

The design of evaluation metrics aiming at quantifying the achieved trade-off between prediction performances and lightweight features of an EdgeAI algorithm is an interesting research topic, which has not been satisfactory explored in literature yet. An example of a preliminary work in this area, proposing a quantitative relative assessment of the prediction performance with computational and architectural complexities of a Deep NN, is the NetScore metric [95], which is described in Subsection 4.3.3.

Chapter 2

IoT/SA-oriented System Architecture

In this chapter, an architectural model expedient to build general-purpose IoT/SA-oriented architectures, which are modular, scalable, able to easily integrate heterogeneous technologies, and supporting the diffusion of SA, is presented. More precisely, in Section 2.1, the main structure of the model is described and critically discussed. Its building components and key elements, together with promising technologies and approaches to adopt to build them, are then presented in the remaining sections.

2.1 Overview

Since relying on a layered structure provides several advantages, the architectural model proposed in this doctoral thesis is based on a stack of four layers (derived from the IoT reference models presented in Sub-section 1.2.2). Indeed, for instance, since its building components are, generally, quite decoupled, an IoT system built around a layered framework is scalable (as outlined in Sub-section 1.2.4).

The proposed IoT/SA-oriented reference model can be illustrated according to two different perspectives, as displayed in Figure 2.1. From a conceptual point of

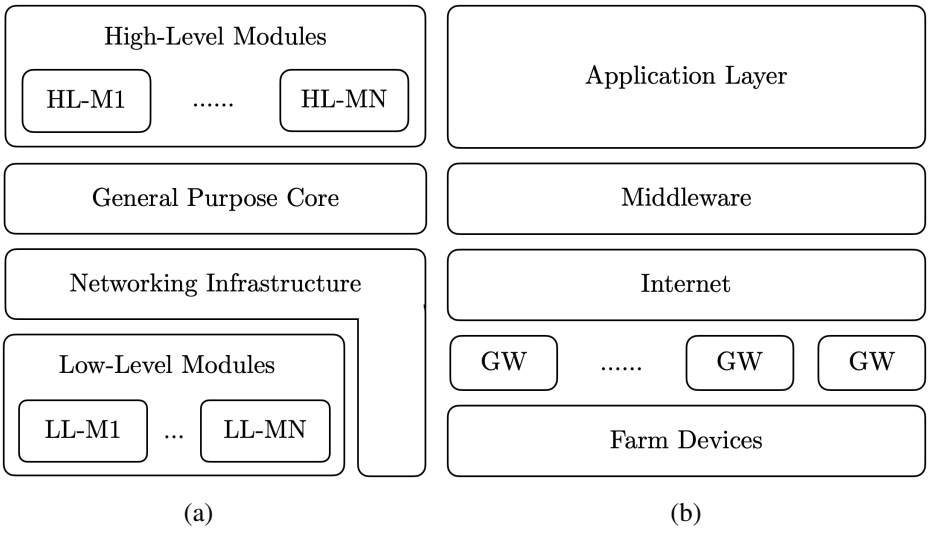


Figure 2.1: Proposed architectural model for IoT/SA-oriented systems from (a) a conceptual point of view and (b) an IoT point of view.

view, shown in Figure 2.1a, the model is built around a core layer providing general-purpose functionalities (as, for instance, devices management, and data storage), usually implemented by a MW layer in an IoT system. Furthermore, the core central layer (or MW) is integrated with modules aiming at customizing the system implementing functionalities needed by a target farm and according to the scenarios to be managed.

More precisely, devices placed in the farm, as smart collars, sensors, tractors, are organized in Low-Level (LL) modules according to their objectives. Moreover, they can exchange information with the core layer thanks to a network infrastructure, including the Internet network and gateways enabling farm devices to access it. As an example, a *greenhouse module* (namely, a LL module), aiming at monitoring air temperature and humidity within a greenhouse, can be built around a set of sensors deployed inside a greenhouse in order to collect information related to its internal climate.

Moreover, the model is completed by the introduction of High-Level (HL)

modules, embedding web, mobile, and desktop applications, data analytical tools, source of historical data, and other software components which provide functionalities to end users (e.g., farmers) and generate new value from data coming from LL modules. For instance, a *weather forecasting module* (namely, a HL module) can be added in order to predict future values of environmental variables, such as air temperature and humidity. In detail, the module can retrieve both data coming from LL modules and historical weather databases, process them, and provide the processing results to a forecasting algorithm (e.g., based on AI), which predicts future weather trends.

2.2 LL Modules

As anticipated, LL modules are composed of the following two classes of devices:

- physical devices deployed in the farm (i.e., farm devices) with a common goal (i.e., monitoring the internal climate of a greenhouse). They can collect different types of information from the farm (e.g., air temperature and humidity) and/or perform some actuation inside its scenarios;
- GWs, enabling farm information to be forwarded to the Cloud (namely, the MW of the architectural model displayed in Figure 2.1a).

A set of best practices and approaches to be followed while developing prototypical devices belonging to LL modules, in details, targeting the collection and monitoring of environmental variables relevant for agricultural activities, are discussed in the following.

2.2.1 Development Boards

Due to the reasons already discussed in Sub-section 1.2.2, building prototypes of farm devices and gateways around development boards is a well-established and convenient practice (Sub-section 1.3.1). The set of development boards to rely on to build custom devices is vast and heterogeneous. Indeed, several low-cost boards,

significantly differing in terms of hardware specifications (e.g., computational power), supported software tools (e.g., programming languages), and built-in networking connectivity, are available on the market.

In general, while selecting a development board (to build a custom SO), multiple, sometimes conflicting, and strictly influenced by the target application requirements have to be concomitantly met. For this reason, two key steps in the development of prototypical devices are, generally, the definition of (i) guidelines for choosing proper boards (in other terms, a set of selection criteria), and, consequently, (ii) the specifications boards should have to meet the selection criteria. To be more clear, selection criteria allow to define which boards, within a large set of valuable choices, are more suitable for a target application.

Board Specifications & Possible Selection Criteria

Criteria driving the selection of development boards can be defined as a set of requirements to be satisfied by the board in order to be selected and used within a target application. These requirements concern the specifications of boards and can be broadly divided in two main classes.

The *first* class is related to boards' hardware requirements and includes: computing capabilities (i.e., MCU architecture); RAM and Flash dimension; built-in wireless transceiver; and power consumed by the board while working. As an example, in order to execute computationally intensive tasks, boards having high computational power and storage space (but, usually, more energy draining than less performing boards) are required. In this case, a possible selection criteria is that the size of the board's RAM and the frequency of its MCU have to be higher than certain thresholds. Moreover, if the board has to join a network built around a specific communication technology (e.g., WiFi) selecting a board with a built-in (WiFi) transceiver can be an additional decision criteria.

The *second* group of boards' requirements is related to firmware and software development, for instance, it includes the programming languages and Operating Systems (OSs) supported by the board. While choosing the programming language to adopt to program a board, some relevant factors, such as the quality of language

documentation for the board and FW/SW debugging support, and the presence of an active and wide developers' community, have to be considered. Indeed, these features directly impact on how easily and rapidly the board can be programmed and, thus, the time required to deploy the firsts versions of code (and the followings) run by it.

2.2.2 Sensor Nodes

SNs, in other terms, IoT nodes gathering relevant information from farms using a set of sensors, can be assembled starting from development boards, sensors, and other components. Indeed, due to the great number of low-cost (waterproof) sensors available on the market, agricultural environmental variables such as, just to mention few parameters, air and soil temperature and humidity, soil pH, Near InfraRed (NIR) and visible light intensity, gas concentration (e.g., CO_2), and air quality, can be efficaciously monitored.

Practically, the information exchange between boards and sensors (and thus sensor data readings) commonly relies on serial communication protocols (such as I^2C). Moreover, at software code level, the above-mentioned communications are implemented using libraries targeting the particular sensor to read, selected board, and used serial protocol. If they already exist (and have not to be developed for a target application), the usage of such already developed sensor libraries simplify and accelerate the integration of sensors and boards: this can be additional criteria to consider in the process of selecting boards and sensors to use in a target application.

Since they usually have to perform lightweight computations and are powered by constrained power sources (e.g., batteries), the development of SNs, targeting SA applications, normally relies on low-power boards with limited computing capabilities. In other words, SNs are generally built around boards which consume few energy, especially while they are not processing data, i.e., are *sleeping* or in *sleep* mode. Indeed, since SNs are deployed in farms, which are characterized by outdoor scenarios where a stable power supply is usually not available, they are normally battery-powered and, thus, have to consume small amount of energy (e.g., in order to not have batteries replaced often). Furthermore, their software programs are usually simple, in terms of number and complexity of code's instructions, small, in terms of

required memory space to be stored, and, thus, not resource demanding. In fact, the program run by a SN generally relies on a periodical cycle: (i) the device wakes up from sleep mode; (ii) reads sensor data; (iii) eventually processes them; (iv) forwards them to the Cloud; and, finally, (v) returns to sleep.

Two popular programming languages for constrained boards are C/C++ and MycroPython. In detail, MycroPython is a framework to develop programs to be run by IoT boards using a lightweight version of Python. Some examples of boards usually programmed in C/C++ include the STM32 Nucleo Boards family by STMicroelectronics [96], the Launch Pad^{TM} evaluation boards by Texas Instruments [97], and Arduino boards [47]. On the other side, MicroPython programmed boards include, for instance, the Pycom boards [49], the Pyboard family [98], and other boards based on ESP32 and ESP8266 System on a Chip (SoC) series (which also support C/C++ language).

A consideration, coming from the practical expertise of many developers, is that developing boards' programs using C/C++ language is usually more challenging than adopting MycroPython (with exception to Arduino-based boards, which, from the presented perspective, are more similar to MycroPython-programmed boards). Indeed, since it had a steeper learning curve, learning C/C++ programming requires both beginners and developers more effort than learning MycroPython (i.e., Python) programming. In fact, the MycroPython framework provides (while developing code) a higher level of abstraction to programmers than C/C++-based frameworks (with respect to the board's hardware characteristics). This, usually, speeds up the software development, furthermore allowing programmers to easily deploy firmware programs (especially those not having an embedded programming background, i.e., beginners). As a drawback, the aforementioned level of abstraction sometimes makes it harder (or even impossible) to access to the board's hardware-related configurable variables (e.g., CPU frequency) and to call low-level primitives provided by the board (which are features enabling to more tightly control a board's operating parameters, as, usually, can be done with C/C++ programmed boards).

For completeness, it is remarked that, although commercial products, providing functionalities similar to those of the above-mentioned devices, are already available

on the market (e.g., Libelium Plug&Sense products family [99]), the realization of custom prototypes of SNs based on development boards is, nevertheless, a valuable option in supporting the creation of IoT/SA-oriented systems. Indeed, commercial products are usually less personalized, less flexible, and more expensive choices with respect to custom-assembled devices. Indeed, costs, together with a board's physical dimensions, are additional key parameters to evaluate while selecting an IoT device for a target application.

2.2.3 Gateways

IoT gateways deployed in the farm can be divided into two groups according to the main implemented functionalities, which can be: (i) enabling Internet access to other farm devices and/or (ii) acting as an intermediary between farm devices and other gateways. Both type of gateways can exchange information with more than one communication interfaces and be implemented buying commercial solutions or development boards. The last option, which allows to more easily customize the gateway's functionalities and is more cost-effective, usually relies on boards which are better-performing, in terms of computing capabilities, than those that SNs are built around. This is justified by the fact that gateways have to, usually, accomplish more complex tasks with respect to SNs. Indeed, GWs' tasks are, normally, real-time and concurrent among each others; moreover, they include, for instance, always listening for incoming packets from different interfaces and processing them; keeping a connection with the Cloud alive; and being remotely controllable. All these activities can be more easily managed introducing an OS of reduced size (e.g., Contiki, TinyOS, and FreeRTOS) on the board. The introduction of an OS, the need of being always active, and the complexity of performed operations usually leads to select boards which consume significant amount of energy. This means that these boards have to be, normally, powered by a stable power source as a power outlet.

An example of a popular family of boards to implement GWs is the RPis boards' family [48]. In detail, RPis are mini-computers supporting several Debian-based OSs and programming languages, with a big supporting community of developers. Moreover, in addition to the built-in WiFi and Bluetooth, there exists several

hardware modules which can be integrated in order to enrich a RPi with extra network interfaces. This aspect is key to develop gateways built around custom devices, due to the reasons already explained (i.e., GWs usually need to support multiple networking communication protocols).

For what concern the internal structure of a LL module, farm devices belonging to the same module usually rely on the same GWs to communicate with the other model's components (and on a shared set of communication protocols). Conversely, devices coming from different LL modules can exploit diverse GWs and networking technologies to reach the Internet.

2.3 Network Infrastructure

Data forwarding between LL modules and the MW is enabled by a network infrastructure, which is composed of local networks of farm devices (i.e., LL modules), gateways, and the Internet. Within local networks, farm devices exchange information with different communication protocols, the most promising of which are presented in Sub-section 2.3.1. Moreover, the main technologies and approaches enabling gateways to access the Internet are discussed in Sub-section 2.3.1.

2.3.1 Networking Technologies

As said, several wireless technologies, differing in terms of power consumption, data rate, and transmission range, can be evaluated to enable communications between farm devices and the Internet, considering the characteristics of the target scenarios. Some of them, grouped in two main categories according to their transmission range, are discussed in the following.

Long-range

The most appealing long-range low-power technologies for SA applications, including crop production, and livestock and tractors outdoor tracking, are LoRaWAN [100], Sigfox [101], and NB-IoT [102]. The three technologies are

critically compared in the following, in terms of key factors to consider in developing real applications, together with some guidelines for selecting one or other for a target implementation [103].

- **QoS.** LoRaWAN and Sigfox communications are asynchronous and rely on the Industrial, Scientific and Medical (ISM) sub-GHz (free-license) bands; the QoS is not guaranteed. On the contrary, NB-IoT adopts a spectrum of licensed bands and the synchronous protocol of LTE; this allows to provide an optimal QoS but at a higher cost. From the perspective of cost and QoS, NB-IoT is more suitable for applications requiring guarantees of QoS, whereas LoRaWAN and Sigfox are better choices for applications which do not have these requirements.

- **End-device battery life.** Although the three protocols are all low-power technologies and, thus, devices consume low-energy while transmitting data using them, NB-IoT devices spend more energy while transreceiving data than LoRaWAN and Sigfox-enable devices, due to synchronous communications and guaranteed QoS.

- **Scalability.** Despite LoRaWAN, Sigfox, and NB-IoT are designed to support thousands of devices within the same network, NB-IoT enables to connect more than the double of devices with respect to Sigfox and LoRaWAN (i.e., $>$100K and 50K devices, respectively).

- **Payload dimension.** Among the three protocols, NB-IoT has the bigger payload size (1600 bytes), followed by LoRaWAN (maximum of 243 bytes) and, finally, Sigfox (12 bytes), for this reason, NB-IoT is more suitable for applications which need to send more data within the same message.

- **Transmission range.** The higher transmissible range is achieved by Sigfox ($>$40 km), which is one of the main advantages of such protocol, followed by LoRaWAN ($<$20 km), and NB-IoT ($<$10 km).

- **Cost.** Considering the cost of devices, deployment, and spectrum (license), LoRaWAN and Sigfox are more convenient than NB-IoT.

As a final comment, since agricultural scenarios are not always properly covered by a stable cellular network, (which NB-IoT relies on), Sigfox and LoRaWAN seem, today, the most valuable long-range low-power technologies for SA applications. Probably, this will change in the future, when LTE network coverage will become even more pervasive than today.

Medium/Short-range

The most common medium/short-range technologies for SA applications, including greenhouse production and livestock indoor monitoring, are WiFi, Zigbee, and BLE. They are compared in the following.

- **Power consumption.** Zigbee and BLE communications are more low-power with respect to WiFi, which is more power consumptive [104].
- **Transmission range.** The WiFi and BLE communication protocols have an higher transmission range (100 m) than ZigBee (10 m), as reported in Table 1.6.
- **Scalability.** The maximum number of devices which can be connected to the same network is 8 for BLE (i.e., 7 devices acting as slaves and 1 as master), 65000 for a Zigbee star network, and 2007 for WiFi. For this perspective, the Zigbee technologies is the more scalable [105].
- **Payload dimension.** Among the three protocols, WiFi is the more capable technology, having a message payload of 2312 bytes, which is followed by BLE (251 bytes) and, finally, Zigbee (102 bytes) [105].
- **Market availability.** Zigbee is less commonly adopted with respect to WiFi and BLE (i.e., WiFi and BLE transceivers can be found in smartphones and other daily life devices), this means that the number of devices available on the market which natively support Zigbee is smaller than the number of WiFi- and BLE-enabled devices.

2.3.2 Internet Access Point

IoT GWs enabling farm devices to exchange information with the Cloud (namely, the MW, according to the proposed IoT reference model displayed in Figure 2.1) need an Internet back-haul in order to forward data to the Cloud. In SA applications, valuable options to connect GWs to the Internet are WiFi hotspots, fiber-cabled connections, and LTE data links. Whenever the first two options are not available (e.g., an Internet connection is not already present in the farm), 3G/4G (or even 5G, in the near future) routers, properly equipped with SIM cards having suitable traffic plans, are convenient solutions in connecting GWs (and, thus, farm devices) with the Cloud.

As an example, two representative devices which can be selected for the purpose are: (i) the HUAWEI 4G Mobile WiFi, and (ii) the GL-X750 Spitz 4G Router, displayed, respectively, in Figure 2.2b and in Figure 2.2a. The first modem is more cost-effective than the second, which, conversely, provides functionalities simplifying its remote control and that of connected devices, which is a desirable feature while deploying smart solutions in remote locations. In detail, these functionalities include, just to name two of them, configurable VPN and open-source OS (i.e., OpenWrt) running on the modem.

2.4 Middleware

As anticipated in Sub-section 1.2.3 and can be concluded by representative literature papers listed in Table 1.8, an IoT MW can be implemented following two main approaches. The first approach is to select an already developed (commercial or open-source) IoT platform, whereas, the second is to design and deploy a custom solution, acting as an intermediary layer between farm devices and high-level applications.

As previously discussed, the first option allows to more easily and fastly develop new applications since it free developers from the management of aspects related to, e.g., devices management, security issues, platform's maintenance and updating. Conversely, it leads to less customizable and flexible (towards the introduction of new

(a) (b)

Figure 2.2: Two representative 3G/4G routers which can provide Internet access to GWs deployed in the farm: (a) GL-X750 Spitz 4G Router and (b) the HUAWEI 4G Mobile WiFi.

functionalities) MWs than the second. Moreover, an additional option is to adopt the FIWARE framework [68, 106, 107], which allows to implement a scalable, modular, and interoperable MW layer. Indeed, modularity, scalability, flexibility, openness, and usage of standards, are key features for MWs able to easily evolve, integrate heterogeneous technologies and components (as the LL and HL modules of the model proposed in Sub-section 2.1).

In general, the design pattern shaping the software architecture of a MW layer, in other terms, the set of principles to be followed to develop it, directly impacts on its modularity, scalability, and on how ease it can evolve. Moreover, the aforementioned desirable characteristics can be further achieved, for instance, adopting open standards, suitable application protocols and communication patterns to design communications existing between internal MW's components and external modules. In detail, these communications are usually based on a share set of Application Programming Interfaces (APIs), in other terms, commonly agreed interfaces, developed according to some rules, which simplify information exchange between different entities. Some examples of tools, more precisely, design patterns for developing APIs and architectures, application protocols, and communication patterns, which can be adopted to build modular and scalable MWs, are detailed

in the following sub-sections.

Microservices Architecture

A worthwhile architectural style, which allows to build highly maintainable, testable, loosely coupled (and, thus, modular and scalable) applications, is the microservices-based architecture. In detail, software applications based on microservices are built around collections of decoupled entities (microservices), which can be deployed and can evolve independently one from the others, but cooperate within the same system. Moreover, in the same application, services, which can be implemented with different technologies, are in charge of executing a reduced set of tasks and can exchange information with other services using a wide set of communication patterns and protocols.

According to the application's specifications, interactions between services (i.e., software's components) can be implemented using request/response-based communications, i.e., an entity makes a request to another and waits for the response, or with an event-driven approach, namely, entities are triggered to perform tasks when an event occurs (e.g., new information are available). Developing a microservices-oriented application starting from scratch could be a very challenging (but feasible) task, for this reason, this architectural pattern is more commonly used to evolve large monolithic applications towards more flexible and decoupled microservices-based applications [108].

APIs

MW functionalities, such as farm devices' data storage and retrieval, can be easily accessed by LL and HL modules (of the IoT reference model in Figure 2.1) thanks to a set of deployed APIs. More precisely, APIs enable data flow between LL and HL modules, defining the set of application protocols, communication styles (e.g., request/response, publisher/subscriber), and syntaxes LL and HL modules have to adhere to in order to access to MW's services. Some examples of design styles and application protocols commonly adopted to develop APIs targeting IoT MWs are

presented in the following.

REpresentational State Transfer (REST). REST is an architectural style allowing to build flexible, scalable, client/server decoupled web applications. A RESTful application is based on the concept of resources and is developed according to six principles: client/server architecture; stateless interactions; cache; uniform interface; layered system; and code-on-demand [109]. In RESTful applications, resources are data units holding information (related to, e.g., a concept, an object, a person) which can change over time. In other terms, resources have a temporary state, which can evolve during their life cycle. Moreover, resources (or rather, their states) are exchanged between the components of an application in the form of representations (i.e., using a target data format, as JSON).

With respect to Service Oriented Architecture (SOA), which is based on the server-side execution of remote calls (e.g., RPCs) according to predefined messages, a RESTful application is based on the exchange of resources (more precisely, a representation of their state in a given instant of time).

REST principles, such as the concepts of resources, their states, representations, and how to exchange them, can be applied to develop *REST APIs* (namely, APIs which adhere to a set of REST principles). REST APIs can exploit different (client/server-based) application protocols in order to exchange resources: as an example, HTTP and CoAP are two popular options in implementing REST APIs targeting IoT applications.

Application Protocols. Two widely adopted application protocols in the context of IoT to develop APIs are MQTT, CoAP, and HTTP, which are compared in the following. It is remarked that, although they aim at supporting the selection of proper application protocols targeting the implementation of MW APIs, the following considerations are generally valid [110].

- **Communication abstraction.** MQTT communications are based on a publisher/subscriber pattern; HTTP is based on a request/response (client/server) abstraction; and CoAP supports both. Due to their

communication abstractions, only CoAP and HTTP support the development of REST APIs; in contrast, being based on a publisher/subscriber pattern, MQTT (and CoAP) simplifies the development of event-driven applications.

- **Message size and overhead.** HTTP has the highest message size and overhead, followed by MQTT and, then, CoAP. This is justified by the fact that HTTP and MQTT run over TCP, which introduces overheads in communications due to connection establishment and closing; otherwise, CoAP runs on UDP, which does not introduce the same connection overhead of TCP, since UDP works in fire and forget basis. Moreover, MQTT and CoAP are binary protocols with respect to HTTP which is text-based: this makes HTTP more verbose and heavyweight. For these reasons MQTT and CoAP are more suitable than HTTP in implementing communications between constrained devices (e.g., farm devices).
- **M2M/IoT.** MQTT is the most adopted communication protocols for IoT/M2M applications, followed by CoAP and, then, HTTP, which usage is limited due to its slow performance and heavyweight size.

2.5 HL Modules

HL modules are sets of software tools and applications exchanging information with the MW (which come from farm devices or other data sources) through a set of APIs exposed by the MW to provide some high-level functionalities, as shown in Figure 2.3. According to the implemented functionalities and objectives, HL modules can be broadly divided in three categories: (i) end-user interactions and services delivery; (ii) information gathering from external data sources; and (iii) advanced analytical and forecasting tools.

End-user Interactions and Services Delivery

The first type of HL enable end-users to easily interact with the system and to access to its functionalities. Some examples of software entities belonging to this

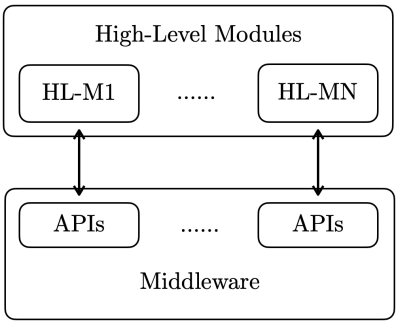


Figure 2.3: HL modules access to MW's stored data and functionalities thanks to a set of APIs.

group are dashboards, which displays real-time sensor data collected within a farm, and systems able to notify farmers of dangerous situations within the farm, moreover, suggesting actions to manage them. This means that, these software programs can be moreover exploited to present results coming from (other) HL modules (e.g., output values of a forecasting algorithm deployed as a HL module and, thus, belonging to the (iii) type of HL modules).

Moreover, final users services delivery and interactions can be achieved developing desktop, mobile, and web applications. In particular, desktop and mobile applications are particularly suitable when hardware ad-hoc solutions have to be deployed for, respectively, PCs and mobile devices, such as smartphones and tables, eventually integrated within agricultural scenarios (e.g., a control panel for smart tractors or greenhouses). Indeed, they usually allow to achieve better run-time performance and prevent sensitive data to be forwarded to the Cloud. Conversely, web applications are cross-platforms and more accessible, since they can be consulted using any browser-equipped device, connected to the Internet.

Information Gathering from External Data Sources

The second category of software tools aims at gathering information from data sources which are external to the IoT system (e.g., historical databases). This in order to enrich the system with additional data, which are relevant for the completion of

target tasks. Possible examples of external sources of relevant data are: governmental reports and statistics; satellite images; online repositories; web services; and social media feeds [111]. For instance, historical weather information can be gathered from a web repository and merged with collected sensor data in order to forecast future trends of weather-related variables (e.g., air temperature).

Advanced Analytical and Forecasting Tools

The last category of software targets the processing of collected data, using advanced techniques, to enable data forecasting, knowledge discovery, and to support decision-making tasks starting from raw data. The main analytical tools and techniques belonging to this group are AI and ML algorithms, image processing, modeling and simulation, statistical analysis, NDVI vegetation indices, and Geographical Information Systems (GIS). As an example, AI and ML algorithms can perform prediction, classification, and clustering tasks; meanwhile, image processing techniques can be used to merge information coming from sensors and images to produce new information [111].

Chapter 3

IoT/SA-oriented System Implementation

In order to validate, from a practical perspective, the architectural framework proposed in Chapter 2 and to present how the design of an IoT/SA-oriented architecture can rely on such reference framework, two IoT systems, aiming at supporting agricultural activities and based on the proposed model, are presented in this chapter, together with correlated experimental results. More precisely, the *VegIoT Garden* architecture [12], an IoT system aiming at enhancing the management of small-size crops, and the *LoRaFarM* architecture [13], intended to improve the management of generic future smart farms in a highly personalizable way, are presented, respectively, in Section 3.1 and in Section 3.2.

3.1 VegIoT Garden

The main goal of the VegIoT Garden architecture [12] (in short, *VegIoT*) is to enhance the management of small-size crops collecting, monitoring, and analyzing data coming from sensors and related to parameters which are relevant for the proper growth of crops' products. These parameters can include, for example, air and soil temperature and humidity. The VegIoT Garden architecture has been mainly designed

to help amateur farmers, e.g., persons which cultivate vegetables in their town house's garden, in growing plants over a reduced area of space (such as a small urban vegetable garden).

More precisely, the application scenarios of the VegIoT Garden architecture are limited-size crops (having an area under 100 m^2), such as urban vegetable gardens and even greenhouses, placed in locations where an Internet AP might not be available in a range lower than 100 m. Furthermore, the system mainly targets users having some knowledge on how growing agricultural products (which are produced for personal consumption) which may benefit from a low-cost system simplifying some crops-related management tasks, e.g., supporting decision-making process, such as deciding when to irrigate a crop or another.

3.1.1 System Architecture

A valuable option to implement the above introduced system is to cover target crops with devices (e.g., microcontrollers) equipped with sensors (in other terms, SNs) measuring a set of selected environmental variables which are relevant for the proper growth of plants. These variables can include solar radiation, weather conditions, temperature, carbon dioxide (CO_2), water, and soil nutrients [112].

The deployed devices (detailed in Sub-section 3.1.3), are organized in a WSN (called Garden WSN or, for short, GaWSN, described in Sub-section 3.1.4), in which information exchange is enabled with the usage of a short/medium communication protocol (namely, IEEE 802.11 or equivalently WiFi), due to the limited size of the area to cover with the selected connectivity (which is under 100 m^2).

Sensor data coming from SNs are collected by a border router, named Gateway Node (GN), which, in turn, forwards the gathered information to another device, labeled as Home Node (HN) and placed far from the GN, in a location where an Internet AP is available, using a long-range communication protocol (namely, LoRa). This last device, i.e., the HN, stores the collected information in a non-relational database and exposed them through REST APIs. The system is completed with an iOS mobile App which allows the end-user to visualize the collected data and to set alarms by which can be notified through emails (e.g., if a crop has to be irrigated).

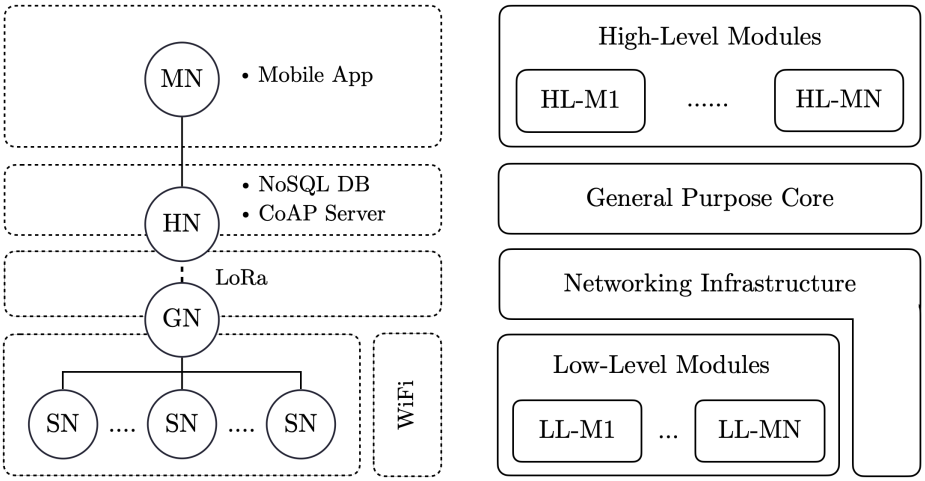


Figure 3.1: Architectural main components of (a) the VegIoT Garden architecture compared with (b) the proposed IoT reference model for IoT/SA-oriented systems.

To summarize, the VegIoT Garden architecture is composed of (i) an IEEE 802.11-based WSN, denoted as GaWSN, deployed in the crop to gather sensor data relevant for the crop's production (i.e., soil/air temperature and humidity) and including a GN enabled with both IEEE 802.11 (towards the WSN) and LoRa connectivity; (ii) an HN, located into an indoor location, connected to an available Internet AP, which receives and stores sensor data coming from the GaWSN; and (iii) an iOS mobile App (i.e., MN), to visualize data collected from sensors deployed in the crops and to set notification alarms for the farmer.

The presented architectural structure is detailed in the following sub-sections with respect to the architectural framework proposed in Chapter 2, from which derives the infrastructure of VegIoT. Moreover, in Figure 3.1, key elements composing the VegIoT Garden architecture (at the left) are compared with the proposed architectural model (at the right).

3.1.2 Evaluation Scenario

The general architecture of the VegIoT Garden has been ad-hocly implemented in order to improve the management of a real scenario, on which the system's functionalities have been evaluated. The evaluation scenario is based on two main locations. The *first* is an urban vegetable garden of approximately 20 m^2, divided into plant beds, where multiple kinds of vegetables are grown (namely zucchini, tomatoes, cabbages, savoy cabbages, and beets). The *second* is the farmer's house, at approximately 200 m from the garden, where is available an IEEE 802.11-based Internet connectivity.

The production period of the garden starts in March and ends in November; furthermore, crops watering is manually performed by the garden's owner, usually in the late afternoon (about 6.00 PM). Crops are qualitatively irrigated almost every day in summer by the farmer, considering the moisture level of the soil. One of the main questions, that the farmer wants to solve, concerns the optimization of watering. More precisely, he cannot understand how much water he has to give to tomatoes plants in order to make them properly grow. Moreover, besides solving irrigation-related problems, another issue (he hopes the system will address) concerns the definition of possible correlations between weather conditions, crops productivity, and plants' health. Due to the above-mentioned goals and issues to solve, air and soil temperature and humidity have been selected as variables to be monitored. Indeed, they can help to optimize irrigation and to forecast potential plants' disease.

3.1.3 LL Modules

As anticipated, an IEEE 802.11-based WSN, based on three SNs, collecting information related to soil and air temperature and humidity, and a GN, acting a sink node, has been deployed in an evaluation scenario, which is a small-size urban vegetable garden. The deployed WSN is a particular implementation of a LL module of the framework proposed in Chapter 2. Indeed, the WSN is based on physical objects (i.e., three SNs and a GN) collecting relevant information from an agricultural

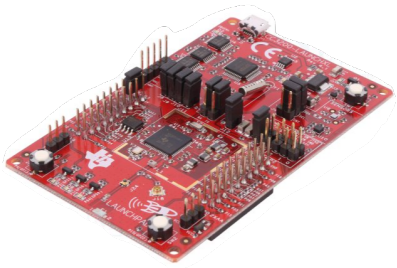


Figure 3.2: CC3200-LAUNCHXL development board.

scenario (namely, the vegetable garden) and forwarding them to the Cloud (which corresponds to the HN).

Sensor Nodes (SNs)

The deployed SNs, which are composed of a development board and a couple of sensors each, are based on low-cost hardware. More precisely, the CC3200-LAUNCHXL (CC) [113], shown in Figure 3.2, has been selected, among other boards available on the market, because of its useful built-in WiFi connectivity, price (27 €), and reduced dimension. Moreover, it can be easily programmed with some of the most popular IDE intended for C/C++-based boards, such as Code Composer Studio, IAR Embedded Workbench, and Energia. In detail, Energia is an Arduino-like IDE which allows to simplify the development of code providing an additional abstraction layer to developers (corresponding to the Arduino platform).

Furthermore, the selected sensors are detailed in Table 3.1, together with their prices and a datasheet reference. As anticipated, three SNs have been deployed in the vegetable garden in order to monitor tomatoes (TOMA), zucchini (ZUCC), and cabbages (CABB) plants grown in the garden. Moreover, they integrate a couple of humidity and temperature sensors, as summarized in Table 3.1. To conclude, each SNs is powered by two type-AA batteries and its hardware is placed inside an IP66 enclosure, which protects it from possible environmental damaging factors

Sensor Name	Common price (€)	Typology	Measured parameter(s)	TOMA	ZUCC	CABB	Datasheet ref.
AOSONG AM2302	1.48	Digital	Air temperature and humidity	x		x	[114]
Soil Moisture Sensor Module	1.80	Analogical	Soil moisture	x	x		[115]
DS18B20	5.48	Digital	Soil temperature		x	x	[116]

Table 3.1: Selected sensors, their typology, related measured parameters, and a datasheet reference with respect to the deployed three SNs (TOMA, ZUCC, CABB).

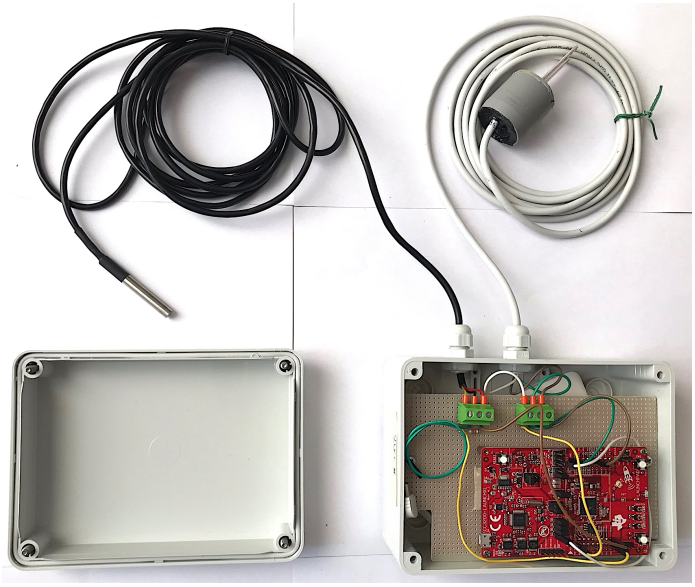
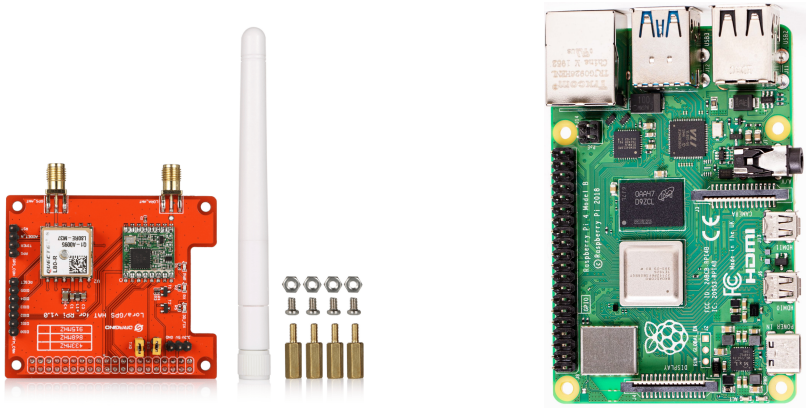


Figure 3.3: An example of implemented SNs of the VegIoT Garden architecture (ZUCC).

characterizing gardens (e.g., bad weather conditions, rain, animals, and insects). An example of a deployed SN (ZUCC) is shown in Figure 3.3.



(a) (b)

Figure 3.4: The GN and the HN of the VegIoT Garden are both based on (b) a RPi equipped with (a) a LoRa GPS HAT.

Gateway Node (GN)

The GN of the designed WSN (GaWSN) implements the following functionalities. *First*, sensor data are gathered from the (three) SNs with a sampling interval of 10 minutes. *Second*, gathered data are locally stored by the GN into a MongoDB database. *Third*, collected data are periodically forwarded to the HN. *Finally*, an IEEE 802.11-based AP is implemented by the GN in order to provide network connectivity inside the GaWSN.

From an hardware perspective, the GN is based on a RPi equipped with a LoRa GPS HAT (both shown in Figure 3.4), which allows the GN to exchange information with the HN using a bidirectional point-to-point LoRa communication. Furthermore, at system start-up, a date time reference (in other words, a timestamp) is sent by the HN, which is connected to the Internet, to the GN to update the GN's internal clock with a correct date/time reference. This is necessary to assign proper date/time references to data coming from SNs.

The main benefits on selecting a RPi to implement a gateway device (as the GN) has been already discussed in Sub-section 2.2.3 and include, just to name few

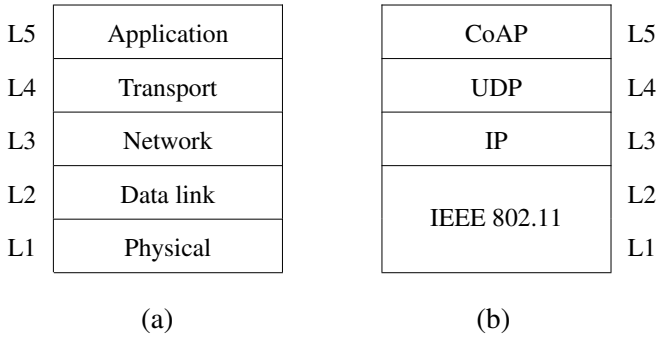


Figure 3.5: Protocol stack of (b) the designed GaWSN compared with (a) the 5-levels IoT protocol stack.

examples, reduced dimensions and other useful functionalities typical of PCs (e.g., embedding in the device a database to store data of interest). As said, the high-power consumption characterizing a RPi is a well-known disadvantage which leads to a careful power management, especially in battery-powered applications. Due to this, the GN cannot be battery-powered, but has to be connected to an available power outlet.

3.1.4 Networking Infrastructure

Communications inside the GaWSN rely on the protocol suite shown in Figure 3.5. The communication protocols composing the designed suite have been selected trying to reach a fair trade-off between heterogeneous connectivity requirements, which include choosing technologies having a good standardization level but also implementing lightweight communications, due to the constrained nature of devices composing the GaWSN.

In detail, since they are widely adopted and compatible with several IoT devices, including microcontrollers, development boards, and smartphones, IEEE 802.11 and Internet Protocol (IP) have been selected as networking technologies. Moreover, due to its medium/short transmissible range, IEEE 802.11 is a valuable solution to enable

communications among devices spread over crops of small size, which the VegIoT Garden architecture targets. Conversely, due to their support for asynchronous and lightweight information exchange, sender and receiver decoupling, and other features which enable an efficient traffic flow among GaWSN's nodes (as the *observing mode* of CoAP), CoAP and UDP have been included in the protocols stack shown in Figure 3.5. As already said, since the garden to monitor misses a proper IEEE 802.11 coverage, an IEEE 802.11-based dedicated AP is provided by the GN.

As already discussed, data coming from the GaWSN, are forwarded by the GN to the HN thanks to a LoRa bidirectional point-to-point link. The LoRa modulation has been selected in order to implement the just mentioned link, since it support long-range and low-power communications. Indeed, as said, in the evaluation scenario, the HN and the GN are placed more than 200 m far from each others and can benefit by the usage of a communication technology which does not require devices much energy while they are exchanging information among each others or are waiting to receive new messages (in other words, when they are listening to the transmissive medium, which is generally a power-consumptive task).

3.1.5 Middleware

According to the structure of the architectural model proposed in Chapter 2, the middleware of VegIoT is built around a layer ad-hocly implemented (and not based on an already existent IoT platform), which is the HN. The main reasons behind this choice are related to the reduced dimension of the application, in terms of volume of collected data (which is not huge), agricultural area to cover (which is reduced in dimension), and number of deployed SNs (i.e., three for the evaluation scenario) composing the GaWSN. These characteristics simplify the deployment of an ad-hoc solution, with respect to that of more complex applications. Moreover, as discussed in Sub-section 2.4, relying on a custom MW allows to build more flexible and easily extensible (with new functionalities) systems, which is one of the VegIoT Garden's requirements.

The HN is built around a RPi board equipped with a LoRa GPS HAT (which corresponds to the same hardware selected to build the GN and shown in Figure 3.4)

and is located inside the house of the farmer. Sensors data coming from the GN are received and stored by the HN inside a MongoDB database. MongoDB is a non-relational, open-source, and document-oriented database. Since MongoDB databases are simple, lightweight (namely, requires small amount of memory to store data), and particularly suitable for semi/unstructured data, having a structure which may change over time (as sensors data collected within the GaWSN), it has been selected as storage technology.

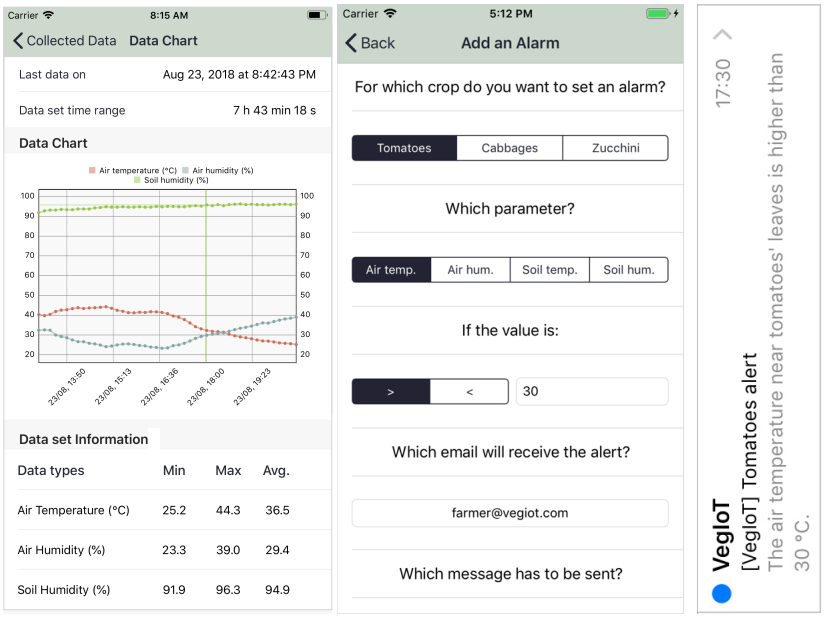
The HN is connected to Internet thanks to an available IEEE 802.11 AP in the farmer's house. Thanks to the existing Internet access, the HN can:

- allow the proper date/time synchronization of the internal clock of the GN at system start-up, forwarding to the GN a UNIX-like timestamp;
- gather relevant data from the Internet, as weather information related to pollutant concentrations and climatic data, in order to enrich the collected sensor data with additional information;
- make accessible the stored data to outside entities through REST APIs and the CoAP protocol (namely, entities composing the HL modules of the proposed model, as the MN of the VegIoT Garden system), performing different types of processing on them.

To conclude, data stored in the HN can be visualized by the end-user with the usage of a MN based on an iOS mobile App (i.e., the VegIoT App). In detail, thanks to the VegIoT App the farmer can monitor the status of its garden's plants and to be alerted if conditions potentially dangerous for plants' health are met, in a simple way.

3.1.6 HL Modules

A developed HL module, using data collected from sensors by the GaWSN of VegIoT, is the VegIoT App (or MN) which, as said in the previous sub-section, allows to visualize collected sensor data and to monitor the status of garden's vegetables. This module belongs to the first class of HL presented in Sub-section 2.5, namely, *end-user interactions and services delivery*.



(a) (b) (c)

Figure 3.6: Two snapshots of the VegIoT App (i.e., MN) displaying (a) a set of collected sensor data compared with a line chart and (b) the form that the farmer can use in order to set email alerts; and (c) an example of an email alert.

An example of service delivered to the farmer by the MN is related to the evaluation of the soil moisture of a target plants bed, which is measured by the SNs deployed in the garden. Indeed, the soil water content value can be visualized with the VegIoT App by the farmer, which can then decide if some plants have to be watered. Moreover, he/she can set custom alerts, as displayed in Figure 3.6b, which notifies the farmer, e.g., via an e-mail message, as shown in Figure 3.6c, if the values of certain environmental variables exceed set thresholds.

The MN has been developed using Swift 3 as programming language. Moreover, it has been deployed, for testing purpose, on an iPhone 7, with iOS 12 running as OS. The MN retrieves data collected from SNs and made available by the HN using the CoAP protocols. In Figure 3.6a is shown a snapshot of the VegIoT App in which

the trend of some sensor data collected by the SN denoted as TOMA are compared among each other with the usage of a line chart.

An example of an additional element which can be integrated in the VegIoT Garden system belongs to the *information gathering from external data sources* category of HL modules. More precisely, it is related to the acquisition of interesting data coming from data sources external to the VegIoT Garden system, such as, weather forecasting data coming from a web repository. These additional information can be stored in the MongoDB database hosted by the HN (together with collected sensor data) and being exploited in order to enable advance data analysis on gathered data (i.e., forecasting a plant's disease given the forecast future values of some environmental variables based on data retrieved by the selected online repository).

3.1.7 Experimental Results

The main functionalities of the VegIoT Garden system have been validated taking the following steps: (i) sensor data have been collected over a fixed period of time of 5 days (as detailed in the first of the following sub-sections); and (ii) the energetic autonomy of the system's battery-powered devices (namely, SNs) have been evaluated (as presented in the second of the following sub-sections).

Collected Sensor Data

As anticipated, the VegIoT Garden system has been deployed in a real scenario and validated collecting sensor data every 10 min during a time period longer than a week. Data collected during five days of test and related to soil moisture, air humidity, and (both) air and soil temperature are, respectively, shown in Figure 3.7, Figure 3.8, and Figure 3.9.

Except for soil moisture, air and soil temperature, and air humidity seem to adhere to some cyclical trends and to have some kind of relations among each others. As an example, the value of air temperature rises during daylight hours and lowers at night; otherwise, the value of air humidity follows the opposite trend, as expected. Moreover, during the considered five days of test, soil moisture and temperature has

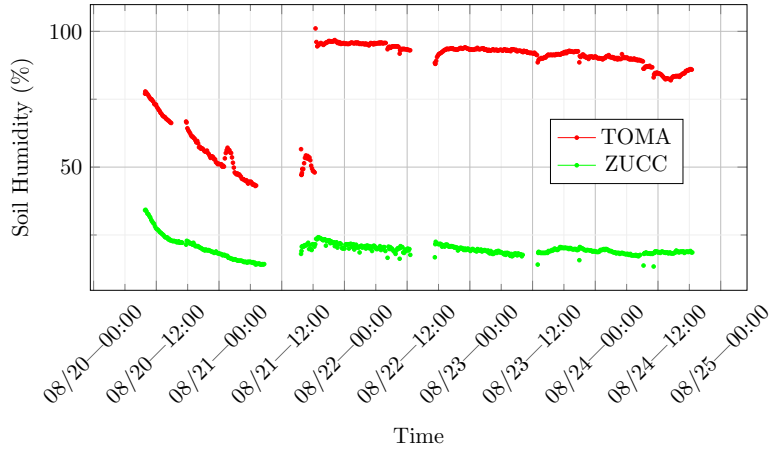


Figure 3.7: Soil moisture values collected during five days of test with the VegIoT Garden system.

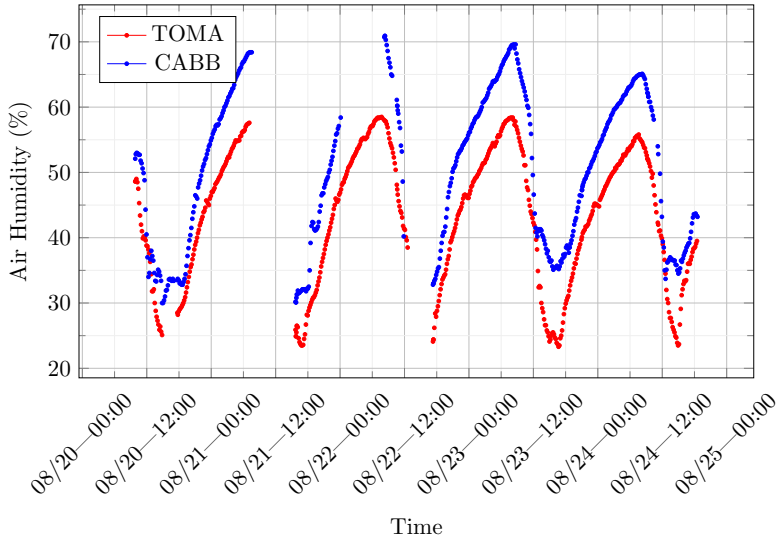


Figure 3.8: Air humidity values collected during five days of test with the VegIoT Garden system.

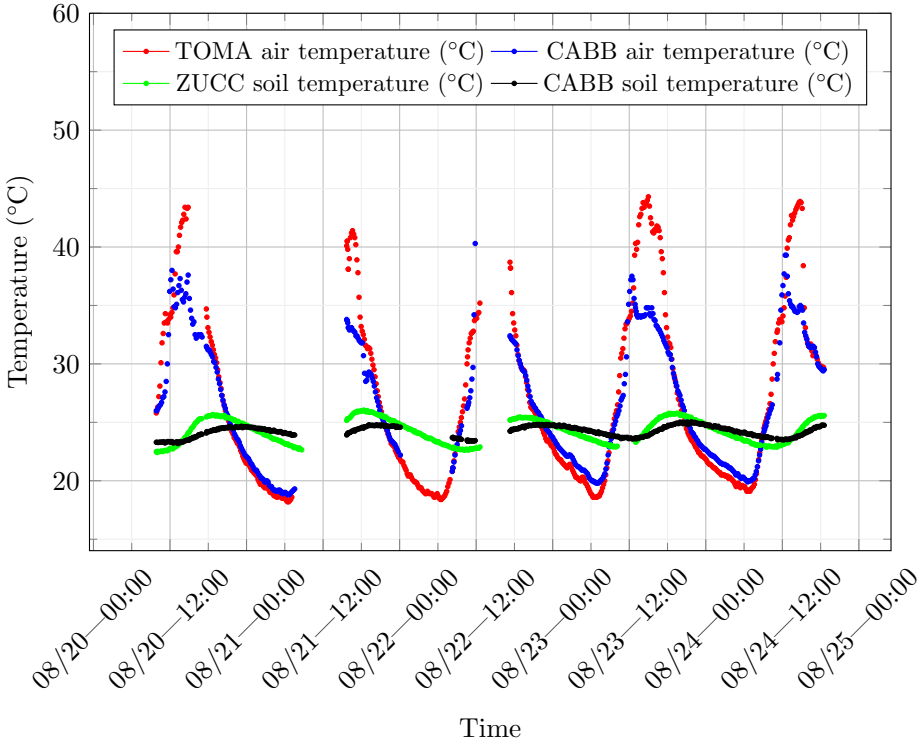


Figure 3.9: Soil and air temperature values collected during five days of test with the VegIoT Garden system.

less significant variation with respect to air parameters.

As can be observed by Figure 3.9, in the first hours of the afternoon, the air temperature which is measured near the leaves of the tomato plants touches very high values, which are grater than 30 °C. Furthermore, high level of soil moisture of tomato plants have been collected during the last three days of test displayed in Figure 3.7. The above-mentioned environmental conditions (namely, high value of air temperature and too high—or too low—soil water content), combined with others (e.g., calcium deficit), can lead to the development of a disease called “blossom-end root” [117]. Some tomatoes grown in the selected garden got ill from the blossom-end root disease, as displayed in Figure 3.10, due to the discussed environmental

Figure 3.10: Tomatoes grown in the garden afflicted by the blossom-end root disease.

conditions. The usage of the VegIoT Garden system could have inhibited the expression of this disease.

Energetic Autonomy Analysis

As already said, IoT devices may be powered by constrained power sources (e.g., batteries, as the SNs of VegIoT are), which makes the proper usage of the available power extremely relevant. This in order to increase the energetic autonomy of power-constrained devices as much as possible.

Given these reasons, the energetic autonomy and current consumption of SNs of the VegIoT Garden system (namely, the battery-powered nodes) are investigated in the following. As a remark, the term "energetic autonomy" refers to the working autonomy of a SN, based on its operative patterns and battery energy. Furthermore, from a temporal perspective, the energetic autonomy of a SN coincides with its lifetime, namely, the period of time by which a SN can work without requiring its batteries being replaced.

Mean Current Consumed and Time Duration

As a first step, the mean current consumed by SNs while working in both of their operational modes, which are active mode and inactive/sleeping mode, have been measured together with their time duration. The obtained experimental results are

State	**Mean current** [mA]	**Duration** [s]
Activity	$I_{\text{active}} = 47.2$	$t_{\text{active}} = 9$
Inactivity/sleep	$I_{\text{sleep}} = 9$	$t_{\text{sleep}} = 600$

Table 3.2: Mean current consumption and time duration of working modes of a SN (active and inactive/sleeping).

collected in Table 3.2. The mean current I_{mean} consumed by a SN can be computed with the following approximate expression:

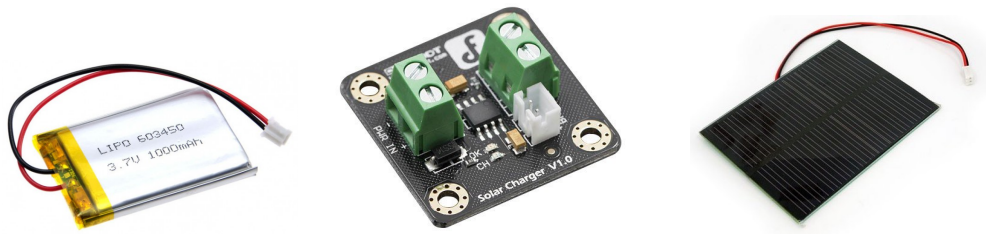
$$I_{\text{mean}} = \frac{t_{\text{active}} \cdot I_{\text{active}} + t_{\text{sleep}} \cdot I_{\text{sleep}}}{t_{\text{active}} + t_{\text{sleep}}} \approx 9.57\,\text{mA}. \tag{3.1}$$

Selecting two type-AA batteries, having both a nominal voltage of 1.5 V (LR6 model) and a capacity C_{AA}, expressed in [mAh], as power source for a SN, the expected lifetime t_{SN} of a SN can be computed as:

$$t_{\text{SN}} = \frac{2 \cdot C_{\text{AA}}}{I_{\text{mean}}}. \tag{3.2}$$

For instance, selecting two type-AA batteries, having a per-battery capacity $C_{\text{AA}} = 2300$ mAh, the theoretical expected lifetime, which is obtained using Eq. (3.2), is approximately of 20 days. From a practical perspective, since the voltage provided by the two type-AA batteries decreases over time while powering a SN, the nominal capacity of batteries cannot be completely leveraged. This because, for every board, there exists a voltage value, which corresponds to 2.5 V for the CC, under which the board stops to work properly: this is the reason behind the reduction of the real lifetime of the SN.

With the goal of defining a more realistic value concerning a SN's lifetime, the following batteries discharge experiment has been carried out. Two type-AA alkaline batteries (i.e., Panasonic Pro Power) have been selected as power supply for a SN until the batteries' potential gains a value under which the CC board can no more forward sensor data properly. The node lifetime obtained with the performed experimental evaluation is around 11 days, as detailed in the following sub-section.



(a) (b) (c)

Figure 3.11: An example of low-cost components which can be adopted in order to power SNs of VegIoT for a system long-term deployment: (a) a LiPo battery; (b) a solar charger module for LiPo batteries; and (c) a solar panel of 1W.

Energetic Autonomy (Lifetime)

As said in the previous sub-section, a SN can operate for approximately 11 days without requiring its batteries being replaced when powered by two type-AA batteries. The obtained value is 9 days lower than the theoretical lifetime computed with Eq. (3.2), which is about 20 days. The undertaken experimental result provides a broader and more realistic perspective related to the requirements of the power source of SNs. In detail, it is clear that another type of power source has to be designed while dealing with a long-term deployment of the VegIoT Garden system, since no farmer would adopt a system requiring him/her to have batteries changed every 11 days).

An alternative, valuable option to two type-AA batteries as power source for SNs, is a powering system built around a lithium battery recharged by a solar panel. This kind of energetic system can be easily implemented selecting inexpensive hardware components, as the ones displayed by Figure 3.11. In detail, the “LiPo Charger Module” displayed in Figure 3.11b can be selected in order to integrate a Lithium Polymer (LiPo) battery with a solar panel. This allows to design a more long-lasting and sustainable power source for SNs (than alkaline batteries), which reduces the need of human interventions, namely, to frequently replace discharged batteries.

Indeed, with optimal environmental conditions (namely, sunny days, having an average air temperature around 25 °C), a small-sized solar panel of 1 W, if properly

directed towards the sun, can completely recharge a 1800 mAh LiPo battery, having a working tension of 3.7 V, in approximately 13-14 hours. This means that a sunshine summer day is usually enough to completely recharge the selected LiPo battery, since the estimated number of hours generally is equal to the number of daylight hours in summer.

Despite the presented considerations rely on optimistic suppositions, even with suboptimal working conditions, such as, bad weather days, and/or non-optimal solar panel orientation, the selected LiPo battery is recharged within two or three days. Due to this last consideration and considering the energy which is daily consumed by a working SN (which is about 10 mAh), it can accomplish its tasks for multiple months without any issue. In fact, although, as an example, several successive days of rain (which lead to a SN's battery being fully discharged) occur, without requiring any sort of human mediation, a SN can return to properly work and forward data, when a sufficient charge level is reached by its battery.

Another possible enhancement related to the management of energetic autonomy of SNs is related to the introduction of techniques aiming at monitoring the charge level of SNs' batteries. As an example, messages, including battery status updates, can be periodically forwarded by SNs to the GN, which, in turn, analyses them and acts accordingly. In detail, whenever the depletion level of a SN's battery becomes lower than a set threshold, a notification can be sent by the GN to the HN, which can notify the farmer through an e-mail or the VegIoT App.

3.1.8 Conclusions

Thanks to the VegIoT Garden system, sensor data coming from an urban vegetable garden have been gathered, stored, and analyzed. This allows the farmer to improve the management of its garden, solving different problems and suggesting actions to undertake in order to enhance the production of its garden's vegetable. As an example, the monitoring of soil moisture enables the farmer to water plants only when needed. Moreover, the analysis of soil temperature allows defining when to seed the garden, considering in which week the germination of seeds would be the fastest.

Moreover, possible crops' diseases can be forecast and prevented visualizing the collected environmental variables and discovering relevant trends and relationships among them. Thanks to the selected technologies, a balanced trade-off among hardware costs, satisfactory standardization level of the adopted networking protocols, modularity, and scalability has been reached. Furthermore, these features simplifies the integration of additional SNs, eventually deployed with different hardware (in terms of selected development board or sensors), and new functionalities inside the system.

Moreover, the components of the GaWSN (i.e., a variable number of SNs and one GN), enabling the collection and monitoring of relevant environmental variables for agricultural production, can be reused and integrated within more complex systems aiming at managing bigger farms (as presented in Section 3.2).

To conclude, the VegIoT Garden can be seen as a powerful system able to collect data of interest for small-size crops production. These data can be furthermore analyzed and processed in order to predict crops productivity and to prevent plant diseases, for example, using AI and ML-based algorithms. For instance, the joint analysis of collected sensor data and information coming from external data sources (e.g., weather forecasting repository) may allow to automatically forecast a plant disease which is caused by some specific environmental conditions and take preventive actions in order to avoid it.

3.2 LoRaFarM

As highlighted in Sub-section 1.3.2, most of the IoT/SA-oriented systems found in literature aim to enhance the management of a limited set of agricultural activities or scenarios a future smart farm may have. Furthermore, an IoT architecture which is modular, scalable, flexible, and can efficaciously handle challenges such as heterogeneity integration has not being already developed for SA applications.

The LoRaFarM architecture (namely, LoRaWAN-based Smart Farming Modular IoT Architecture) [13], presented in this section, aims to address the above-mentioned research gaps following state-of-the-art approaches (including multi-protocols GWs and MWs) and implementing the architectural model presented in Chapter 2 (as more clearly discussed in Sub-section 3.2.1). More precisely, LoRaFarM allows to gather, handle, and process data which are interesting for the production processes of a generic farm, such as the environmental growing condition of crops (as the VegIoT Garden system presented in Sub-section 3.1) and greenhouses' products, and livestock animals. This to improve the management of the overall activities of a farm.

To conclude, it is remarked that, due to the above-mentioned objectives, farms of heterogeneous sizes (from small to big companies), characterized by several activities, and which may benefit by the introduction of a single integrated system to optimize their production processes are target scenarios of LoRaFarM.

3.2.1 System Architecture

The architectural structure of LoRaFarM, shown in Figure 3.12a and briefly summarized in the following, is directly derived from the model proposed in Chapter 2. With respect to Figure 3.12a, it is remarked that the LoRaFarM architecture is built around both networking and software components, although these last once are not explicitly represented in the figure. In particular, the middleware and the HL modules layers include a set of software entities, modules, and services which, for the sake of clarity, are not displayed by Figure 3.12a.s More precisely, the LoRaFarM architecture is built around a core central layer (namely, the middleware), making the system applicable to any farm, enriched with a set of modules, aiming

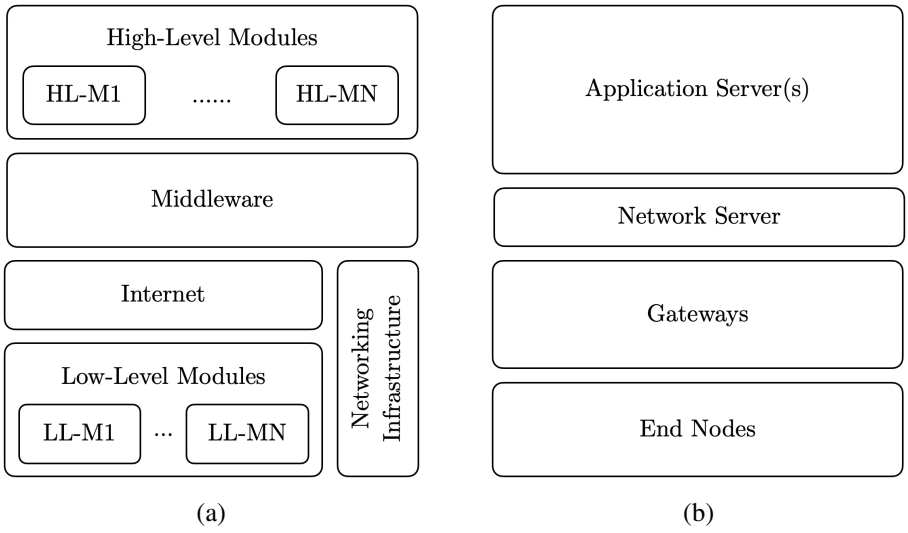


Figure 3.12: Architectural main components of the LoRaFarM architecture (a) compared with the LoRaWAN architecture (b).

at managing different scenarios and activities a target farm may have. For instance, if cows' GPS position has to be monitored, a *cows' position tracking* module can be integrated within the system without redefining its core structure; moreover, this is even true for *vineyard* and *greenhouse management* modules.

In detail, there exists two types of modules which can be integrated inside LoRaFarM, in compliance with the architectural model proposed in Chapter 2. The first class corresponds to LL modules, which are introduced at *farm-* (or *low-*) *level* and includes physical devices to be deployed in the farm; the second, namely, HL modules, is added at *high-level*, in order to enable advanced processing and analytics on data of interest.

Moreover, the network infrastructure layer of LoRaFarM, which, as said, allows LL modules to exchange information with the MW, is built around the LoRaWAN architecture. According to the LoRaWAN architecture, data coming from farms (namely, from LL modules) are collected by LoRaWAN-enabled devices called ENs and forwarded to a Network Server (NS) by a LoRaWAN Gateway (GW). As a

remark, the just mentioned ENs can play two different roles within LoRaFarM: they can act as (i) actuator nodes/SNs or as (ii) border routers, if they allow not-LoRaWAN-enabled devices to join the LoRaWAN network. In detail, the purpose of devices belonging to the first category can be:

- to directly collect data which are interesting for agricultural activities (e.g., water content of a crop, air temperature values of a greenhouse), coming from connected sensors, and send them to the MW (namely, NS and Application Server (AS)) over LoRaWAN (leading to sensor nodes);
- to make automatic some farm's operations, as field irrigation and greenhouse's sides opening (leading to actuator nodes).

Conversely, the aim of an EN belonging to the second class, denoted as multi-protocol Gateway (mp-GW), is to gather data from not-LoRaWAN-enabled devices (e.g., WiFi-based SNs) and forward them to the MW using a LoRaWAN communication (and vice versa), as more clearly explained in Sub-section 3.2.3. To be more clear, mp-GWs, namely, devices equipped with more than one network interfaces, are introduced as an expedient to simplify the integration of IoT devices which exchange information with different communication protocols inside LoRaFarM [31, 118]. (As an example, a similar approach has been followed in [119], in which a multi-protocols gateway aiming at connecting IoT devices with heterogeneous communication interfaces to the Cloud, has been built around a smartphone.)

To conclude, an AS retrieves data coming from ENs from the NS and stores them inside the LoRaFarM's middleware, which, in turn, makes them available to HL modules. In Figure 3.12, the LoRaFarM layered structure (at the left) is compared with the architectural components of LoRaWAN (at the right).

3.2.2 Evaluation Scenario

The evaluation scenario in which LoRaFarM has been deployed and validated is an existing Italian farm (namely, the "Podere Campáz" [120], which is one of

demonstrators of the H2020 AFarCloud project). The set of agricultural activities covered by the selected farmhouse is large and heterogeneous. In fact, it includes both open field and greenhouse cultivation, together with the production of several edible products, such as vegetables, fruits, edible flowers, wine grapes, herbs, and honey.

The farm is based on the following parts: (i) a small greenhouse with a dimension of 20x9x5 m^3 (length, width, maximum height) where horticultural products are grown; (ii) a wide vineyard of about 3 ha; (iii) an area equipped with beehives; and (iv) a mixed fruits' orchard. The farmer manually manages the farm's greenhouse, without being supported by any technology. For instance, no system supporting the remote monitoring of the environmental variables characterizing the internal climate of the greenhouse is present. In detail, the greenhouse's left and right sides are manually opened and closed by the farmer in order to control the internal air humidity using a natural ventilation; moreover, the activation of irrigation sprinklers and other tasks are performed by the farmer according to his experience.

3.2.3 LL Modules & Networking Infrastructure

As already explained, the network infrastructure of LoRaFarM is based on the LoRaWAN technology; moreover, LL modules embed networks of ENs, which can act as sensor/actuator nodes or mp-GWs. Due to these reasons and that introducing additional ENs in a LoRaWAN network, generally, requires few actions (as the registration and activation of devices to the NS), one can conclude that LL modules can be added to LoRaFarM with small effort. In other terms, the adoption of the LoRaWAN paradigm (as networking technology) makes LoRaFarM "inherently" modular and scalable. Moreover, additional advantages provided by LoRaWAN are that it is inherently simple and that a LoRaWAN coverage can be, generally, deployed almost everywhere, without the need of having an already existent networking infrastructure covering the area of interest (in contrast with other long-range technologies, as NB-IoT and Sigfox), as further discuss later in this section.

Physical devices belonging to a LL module are arranged according to one of the two topologies shown in Figure 3.13. More exactly, a LL module can be classified as:

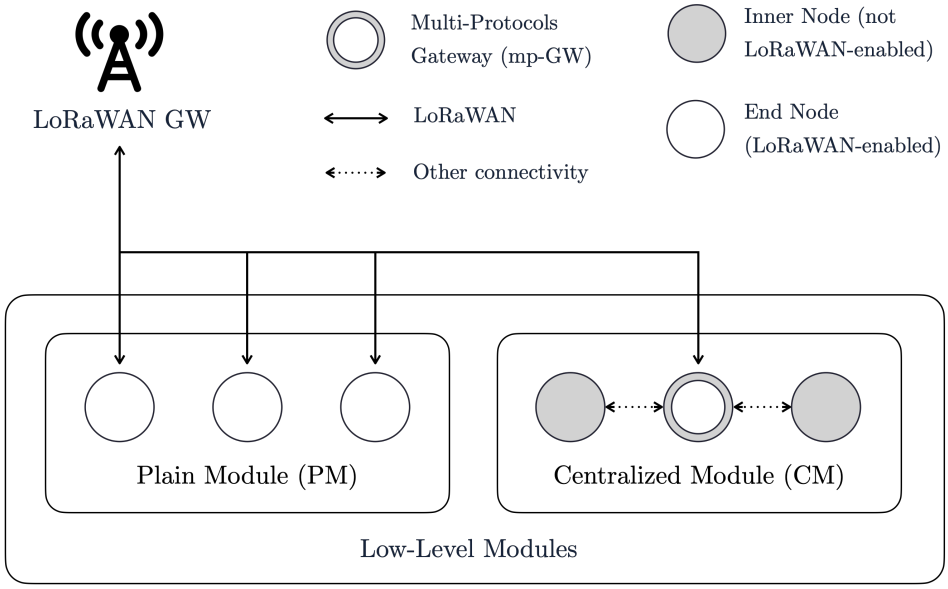


Figure 3.13: The two typologies of LoRaFarM's LL modules.

- *Plain Module* (PM): if it consists of nodes which are LoRaWAN-enabled (i.e., ENs) and exchange information with the MW directly, without the need of an intermediate entity (namely, a mp-GW);
- *Centralized Module* (CM): if it embeds a set of no-LoRaWAN-enabled nodes, labeled as Inner Nodes (INs), and a GW which is LoRaWAN-enabled (i.e., a mp-GW).

In detail, a mp-GW is an IoT node which can dialogue with (at least) two different networking protocols: the first is LoRaWAN, whereas the latter(s), adopted to gather information from INs, is not fixed. Furthermore, one of the main tasks of mp-GWs is the translation of messages coming from and directed to INs between one (or more) protocol and another. This in order to allow non-LoRaWAN-enabled nodes to exchange information with the LoRaFarM middleware, in a straightforward way. As a clarification, a mp-GW is linked to the MW of LoRaFarM thought the LoRaWAN GW, which receives and dispatch packets coming from the mp-GW (and, thus, INs)

to the MW, and vice versa.

The following advantages are offered by the outlined integration methodology of different devices to LoRaFarM. *First*, there is no need to alter the system's infrastructure in order to incorporate heterogeneous, in terms of, e.g., transmission range, energy consumption, and data rate, subnetworks of devices. This makes LoRaFarM flexible, highly scalable, and suitable to handle different agricultural activities, since the most convenient communication technology and traffic management pattern can be selected considering the farm's scenarios to monitor and to control (e.g., fields, greenhouses, and stables). Due to this and based on their dimension, topologies, and data flow's specifications, subnetworks made of INs can more efficaciously operate. Moreover, edge computing functionalities (in addition to protocol translation) can be embedded in the mp-GW, thus, enabling different kind of processing on gathered data, as data aggregation and data fusion. As an example, with such expedient, the uplink traffic of a CM, directed to the LoRaWAN GW, can be optimized.

Another possible advantage concerns the internal structure of a farm and the spacial distribution of its agricultural scenarios. In detail, a farm can be considered as a set of "Productive Units"(PU), where animals and agricultural products are grown (in other words, the farm's scenarios), and a "Central Management Unit" (CMU), where a farmer can remotely monitor and control his/her farm. Moreover, in the CMU, which can, for instance, be the farmer house, is usually available an Internet AP. On the contrary, a reliable Internet connectivity may not be present in the farm's PUs; in other cases, an Internet AP is present, but a data plan has to be signed to access it (e.g., this is true for Sigfox, NB-IoT, and cellular network coverage). Due to the above considerations, devices composing the first layer of LoRaFarM can be installed in the farm as follows. The LoRaWAN GW can be placed within the CMU and connected to the available AP; conversely, one or more LL modules, exchanging information over LoRaWAN, can be deployed in different PUs to cover them. The presented approach provides connectivity to several devices, placed in the farm scenarios (i.e., PUs), using a unique Internet AP. Together with the other discussed reasons, this last feature makes LoRaFarM ideally applicable to any farm,

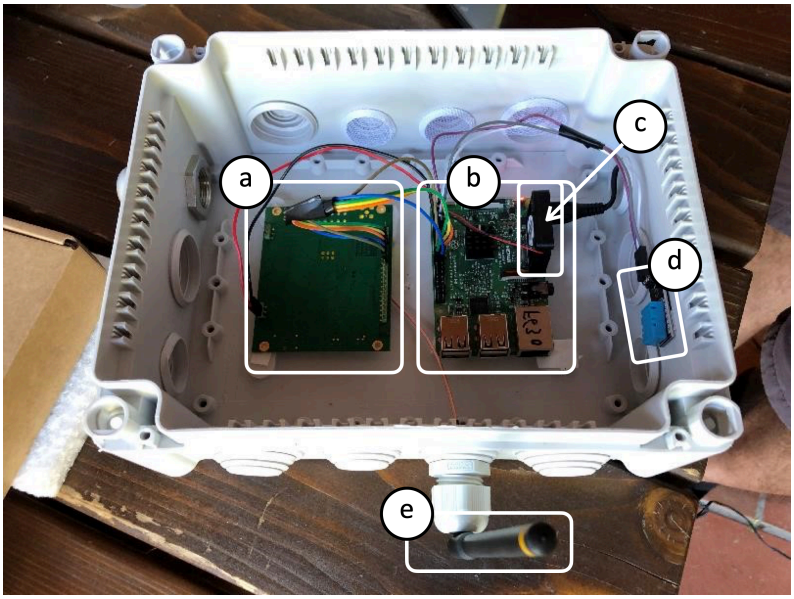


Figure 3.14: The developed LoRaWAN GW: (a) iC880A-SPI concentrator board; (b) RPi; (c) cooling fan; (d) DHT11 air sensor; and (e) a 868 MHz antenna.

despite its specific characteristics.

To conclude, in order to validate LoRaFarM in the selected evaluation farmhouse (described in Sub-section 3.2.2), two LL modules have been designed and deployed in the farm: a (i) *vineyard module*, aiming at monitoring soil moisture and temperature of the farm vineyards; and a (ii) *greenhouse module*, gathering sensor data related to internal environmental conditions of the farm's greenhouse.

LoRaWAN Gateway

The considered evaluation scenario of LoRaFarM (i.e., the Podere Campáz), as many other farms, lacks of an already present LoRaWAN coverage. For this reason, in order to enable LoRaWAN communications, a LoRaWAN GW has been deployed in the CMU of the considered farm, namely, the farmer's house, and connected to an available WiFi AP, following the approach described in the previous paragraphs.

From an hardware perspective, the GW, which is shown in Figure 3.14, is built around low-cost hardware. In detail, a RPi has been equipped with (i) networking components making the RPi compliant with the LoRaWAN technology (namely, a iC880A-SPI concentrator board and a 868 MHz antenna); and (ii) a cooling system, to avoid the inner temperature of the GW reaching too high values. (More information concerning the GW hardware components and their objectives can be found in Table 2 of [13].)

Plain Module

As said, PMs are based on LoRaWAN-enabled devices (called ENs) which directly join the LoRaWAN network, since an intermediary entity (namely, a mp-GW) is not needed to process or forward their data. Due to this, PMs are particularly suitable to monitor wide areas of space, where devices which gathers data of small dimensions and at a low rate are spread. Moreover, a PM can be designed to control a set of actuators based on decision processes usually performed at high-level (in other words, by HL modules). For instance, a set of ENs equipped with sensors and collecting information related to soil moisture values can be deployed as a PM in a large vineyard. The measured sensor data are forwarded by the installed ENs to an HL module, which processes them, decides if the vineyard has to be watered or not, and, then, notifies the farmer accordingly.

Vineyard Module A vineyard module, collecting sensor data related to soil humidity and temperature, has been deployed in the evaluation farm to help the farmer planning crops irrigation and other tasks. The developed module is based on two ENs, each built with the following inexpensive components: (i) a processing unit, based on a LoPy4 board [121], equipped with a 868 MHz antenna, which reads a set of connected sensors and forwards read data; (ii) a couple of water-resistant soil sensors (DFRobot Soil Moisture Sensor [122] and DS18B20 [116]); (iii) a power source system, composed of a rechargeable LiPo battery charged by a solar panel; and (iv) an IP66 box and other connectors for the node's hardware (as detailed in Table 3.3). The hardware components of one of the deployed ENs is shown in

Block Name	Components	Details
Processing Unit	LoPy4 board	Functionalities: duty-cycling (time period of 10 min) between short active stage (sensor are red and data forwarded) and deep sleep stage. [121]
	868 MHz antenna	Connected to the LoPy4 board; allow sensor data forwarding over LoRaWAN in the European ISM bandwidth.
Two soil sensor	DFRobot Soil Moisture Sensor	Analogical sensor; water-proof; voltage signal; calibration required; measures soil moisture. [122]
	DS18B20	Digital sensor; water-proof; 1-Wire serial protocol; measures soil temperature. [116]
Power source system	LiPo battery	Voltage: 3.7 V; capacity: 1800 mA.
	Solar LiPo charger	Enables LiPo battery to be charged by a solar panel. [123]
	Solar Panel	Power: 1 W; size: 80x100 mm.
Other components	Protective enclosure	Three cable claps and one IP66 box.
	Veroboards; cables; female pin headers; jumpers; mammut terminals; terminal blocks.	

Table 3.3: Details concerning ENs of the vineyard module's building blocks and selected hardware.

Figure 3.15.

Among other boards available on the market, the LoPy4 has been chosen because of its reduced size, low price, support for four communication technologies (i.e., WiFi, BLE, LoRaWAN, and Sigfox), and easily programmable. Furthermore, it has natively a deep sleep mode, during which the LoPy4 drains a current of only 25 µA. In LoRaFarM, where ENs sleep for most of their life-cycle and wake up only for few seconds before coming back to (deep) sleep, the native deep sleep mode of LoPy4 is an interesting feature since it allows to maximize the usage of batteries powering ENs.

The software program running on the LoPy4 board is developed using the MicroPython framework presented in Sub-section 2.2.2 (namely, a reduced version of Python particularly designed for microcontrollers). In detail, the software is based on a main cycle, periodically repeating the following stages: (i) the node awakes from deep sleep; (ii) reads values coming from sensors; (iii) forwards sensor data over LoRaWAN; and, finally, (iv) comes back to deep sleep. The cycle is repeated every

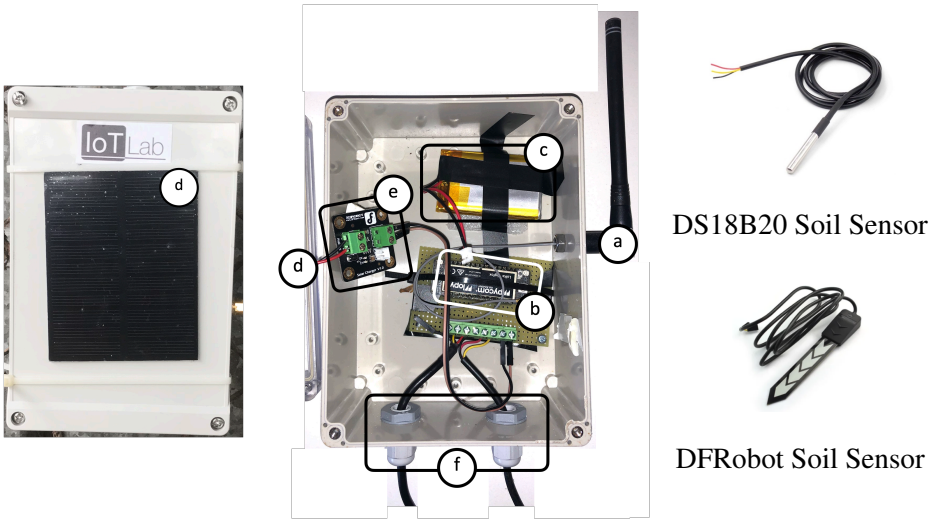


Figure 3.15: The hardware selected to implement an EN of the vineyard module: (a) a 868 MHz Antenna; (b) a LoPy4 board; (c) a 3.7 V, 1800 mA LiPo Battery; (d) a 1W Solar Panel; (e) a Solar LiPo Charger; (f) two soil sensors (i.e., DS18B20 and DFRobot soil sensors).

10 min.

The analogical DFRobot Soil Moisture Sensor [122] together with the digital 1-Wire DS18B20 [116] sensor have been selected to, respectively, measure soil humidity and soil temperature. These sensors are waterproof, accurate enough, and have an operational range of temperatures which is appropriate for the target application (indeed, they can correctly work, at least, from $-40\,°C$ to $80\,°C$). The power source system of an EN is based on a 3.7 V LiPo battery, with a nominal capacity of 1800 mA, connected to and recharged by a solar panel of 1 W thanks to a solar LiPo charger, as displayed in Figure 3.15.

Two ENs, denoted as SN-V1 and SN-V2 and sharing common hardware, have been deployed in the vineyard of the Podere Campáz. The limited number of deployed nodes is justified by the main goals of the undertaken validation campaign: test the robustness of LoRaFarM, in terms of developed devices stability and

Figure 3.16: An EN (SN-V1) belonging to the vineyard module deployed in the Podere Campáz.

appropriateness of adopted sensors, during a fixed period of time (i.e., 6 months) before deploying the system for a longer period, during which more nodes will be included (at least six). An EN (namely, SN-V1) deployed in the farm's vineyard is shown in Figure 3.16.

Centralized Module

As outlined in Sub-section 3.2.3, a CM is composed of a subnetwork of INs, which are devices transmitting and receiving messages with a communication protocol differing from LoRaWAN and connected to system's MW thanks to a mp-GW. The main benefits provided by this approach are discussed in the following.

First, CMs can cover scenarios which may benefit by some sort of local

data processing (in other terms, edge computing) because they generate, or, more generally, have to manage, a considerable volume of data. In fact, knowing that (i) the payload of a LoRaWAN message is limited in size (i.e., the maximum payload dimension is around 243 bytes) and that (ii) the period of time between the transmission of two successive packets is of several minutes, optimizing the number of forwarded packets and the usage of bytes of messages' payload (in other terms, the uplink traffic) is highly recommended. As an example, two valuable options the mp-GW can implement to achieve the above-mentioned goals are data aggregation and data fusion. As a remark, the mp-GW is usually more capable than INs, in terms, for instance, of power supply, available storage memory, and computing capabilities. Indeed, with respect to INs, which are more often powered by batteries, a mp-GW is usually connected to a power outlet and may have a larger storage memory and RAM. This allows to delegate the execution of some tasks normally performed in the Cloud, as the remote control of INs equipped with actuators (e.g., to open a greenhouse's sides, to manage an irrigation system), to the mp-GW. Besides the reduction of the MW's workload and computation latency, with this approach, "smart environments", namely, environments which are self-managed regardless the Internet connection provided by the LoRaWAN GW is available or not, can be implemented.

Greenhouse Module and VegIoT Garden Integration An enhanced version of the GaWSN of the VegIoT Garden system presented in Section 3.1 have been deployed as a CM in the greenhouse of the evaluation farm. This in order to monitor relevant information related to internal climate of the selected greenhouse (i.e., air temperature and humidity). Some improvements aiming at enhancing and effectively integrate the GaWSN in LoRaFarM have been conducted; they are outlined in the following, with respect the ones described in Section 3.1.

As a brief recap, the connectivity within the GaWSN is enabled by the protocol stack reported in Figure 3.5. In accordance with the WSN nomenclature, the GaWSN is built around three SNs and a GN; conversely, according to the notation of LoRaFarM, the GaWSN, still formally a WSN, can be seen as a CM, composed of, respectively, two INs (instead of three SNs) and one mp-GW (namely, the GN). (This

last notation is adopted, in the following, to refer to the components of the GaWSN which have been integrated within LoRaFarM.)

Multi-protocol Gateway A RPi has been equipped with a LoRa GPS HAT (both displayed in Figure 3.4b) in order to build a mp-GW which can join a LoRaWAN network. In detail, the point-to-point LoRa communication between the GN and the HN of the VegIoT Garden has been changed to a LoRaWAN link between the mp-GW and the LoRaWAN Gateway. Moreover, an IEEE 802.11-based AP is implemented by the mp-GW, which is powered by an available power outlet near the greenhouse's entrance, in order to provide WiFi connectivity to INs. Before being sent to the system's MW (thanks to the deployed LoRaWAN GW), sensor data coming from INs are aggregated by the mp-GW which, from this perspective, acts as a concentrator of data.

The reasons behind the selection of a such "powerful" board, which is the RPi, in order to implement the above-mentioned reduced set of tasks (i.e., data collection, aggregation, and forwarding), considering also the limited number of INs deployed in the greenhouse (namely, two), are discussed in the following. *First*, during a long-term deployment of the system, the importance of data aggregation will grow significantly since more INs will be installed in the greenhouse. *Second*, the introduction of new functionalities, locally executed in the greenhouse and related to the management of its actuators, is simplified by the usage of a RPi. In fact, when the actuator allowing to open and close the sides of the greenhouse is mounted, the mp-GW will be used to automatically pilot them, according to the values of sensor data it collects from INs (e.g., if unwanted values are touched by air temperature, the sides are opened).

Inner Nodes The greenhouse module is composed of two INs monitoring air temperature and humidity inside the evaluation greenhouse. From an hardware perspective, INs share some HW components with ENs, e.g., in terms of power source system, and design principles, but differ in terms of selected board and adopted sensors, as summarized in Table 3.4. Indeed, since they are derived from SNs of the

Block Name	Component Name	Component Details
Processing Unit	CC3200 Board	Functionalities: duty-cycling (period of 10 min) between short active stage (sensor data reading and forwarding) and inactive stage (hibernate mode). [113]
One air sensor	AM2032	Digital sensor; 1-Wire serial protocol; measures air temperature and humidity. [114]
Power source system	LiPo battery	Voltage: 3.7 V; capacity: 1800 mA
	Solar LiPo charger	Enables LiPo battery to be recharged by a solar panel. [123]
	Solar Panel	Power: 1 W; size: 80x100 mm.
Other components	Protective enclosure	One IP66 box.
	Veroboards; cables; female pin headers; jumpers; mammut terminals; terminal blocks.	

Table 3.4: Building blocks and selected hardware in order to implement INs of the greenhouse module's.

GaWSN, greenhouse INs are built around a CC board [113], whereas ENs are based on a LoPy4 board [121].

Due to the reasons already discussed in Sub-section 3.1.3, the CC board is an appealing choice in order to implement INs. In fact, together with its limited size and built-in IEEE 802.11 connectivity, it natively provides three low-power sleep modes in which the board consumes small amount of energy: low-power deep sleep, deep sleep, and hibernation.

In order to evaluate the power consumed per hour by a CC during the hibernation mode, which is the low-power sleep mode of the CC having the lowest energy consumption among the overall (and, thus, considered in this work), an experimental analysis has been undertaken using a multimeter. Results show that the CC board has an hourly energy consumption of approximately 14 mAh in hibernation mode. A possible solution aiming at further reducing the energy consumed per hour by the board from 14 mAh to 9 mAh is to remove one of the two on-board LEDs which are always on when the CC is powered. To conclude, the two INs (SN-G1 and SN-G2) are equipped with an AM2032 air humidity and temperature sensor [114], as summarized in Table 3.3. For a software perspective, the program running the CC board and presented in 3.1 has been unaltered. One of the two INs deployed inside

Figure 3.17: An IN of the greenhouse module deployed inside the greenhouse.

the greenhouse is displayed in Figure 3.17.

3.2.4 Middleware

In the context of LoRaFarM, the MW is a set of entities and technologies which allows data coming from LL modules to be gathered, stored, and made available to HL modules. Due to this, the MW can be defined as a sort of "linker" between farm's devices and software components. From the perspective of the LoRaWAN architecture, the middleware embed a NS and an AS, as shown in Figure 3.18.

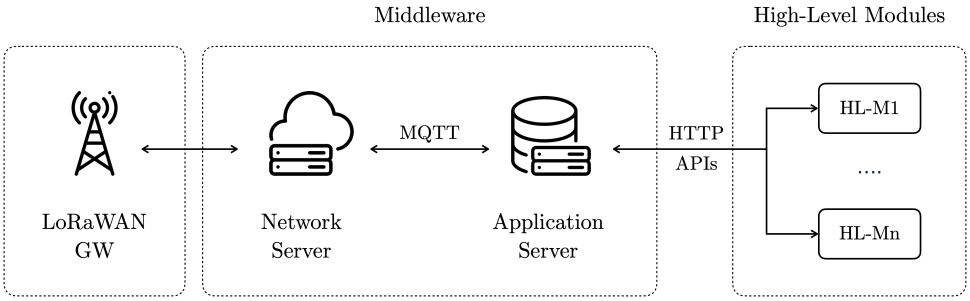


Figure 3.18: The middleware of LoRaFarM enables information exchange between LL and HL modules.

Network Server (NS)

Along with the main functionalities a NS is generally in charge of implementing, which include, just to name few examples, authentication and authorization of ENs devices, data routing, network encryption and decryption, it has, normally, also to provide access to data coming from ENs (and forwarded by LoRaWAN GWs) to other digital entities, i.e., ASs. In LoRaFarM, data coming from LoRaWAN-enabled devices (namely, ENs and mp-GWs) are received by an MQTT broker, which is part of the NS. The NS is implemented thanks to a public network named "The Things Network" [124].

Application Server (AS)

The main functionalities of the AS embed inside the LoRaFarM's MW are related to the management of farm data, which is implemented thanks to the following set of operations and entities: (i) a group of MQTT clients, one per each LoRaWAN-enabled device belonging to a LL module, collect farm data from the NS; (ii) farm data are stored into a relational database; and (iii) stored data are made available to HL modules through HTTP APIs.

A relational database has been adopted as storage technology since the system requires a robust and general-purpose repository at first-level, which can stock

LoRaWAN packets coming from farm devices in a well-structured way. In fact, since LoRaWAN packets are structured data, in other terms, they encode a set of information using a group of fields, which are well-defined and do not frequently change over time, the selected SQL-based storage technology properly fits the storing requirements of these data. (Indeed, LoRaWAN packet's format may, usually, only vary according to the release of new specifications, which have been in the order of around one per year in the last 4 years, namely, from year 2017 to year 2021.)

It is remarked that, along with LoRaWAN packets, data coming from farm devices (e.g., soil humidity values) are stored within the deployed database, indeed, farm data are included in the application-layer payloads of the stored LoRaWAN packets. As a comment, the storage of LoRaWAN packets enables, as an example, to analyze the performance of the deployed LoRaWAN network, for instance, in terms of lost packets, Received Signal Strength Indicator (RSSI), and network latency.

Even though the current data storage requirements of LoRaFarM are satisfied by the adopted solution (namely, a relational database holding both LoRaWAN packets and farm data), in terms of amount and type of data, for a long-term deployed of the system, the data storage strategy can be enhanced.

Indeed, due to the nature, typology, and potential volume of farm data to be handled by the system (in other terms, semi/unstructured data coming from sensors and IoT devices), non-relational storage paradigms (as time series-based databases) aiming at storing farm data can be included in LoRaFarM. More precisely, the storage of LoRaWAN packets and farm data can be distributed among multiple and more specialized entities. As an example, a possible storage strategy is to use both a relational technology, to store structured data (e.g., LoRaWAN packets), and a time series-oriented database, to store big volume of small-sized data, which are periodically collected (i.e., sensor data). Moreover, since time series-oriented databases allows to store sensor data in a lightweight and simple way, data can be more fastly and easily retrieved from them by HL modules, thus improving the overall system performance.

3.2.5 HL Modules

The main functionalities of HL modules include, just to new few of them, enabling the farmer to interact with the system; processing collected data with advanced processing techniques (e.g., ML algorithms); and retrieving data from external data sources (Section 2.5). From a practical perspective, HL modules gather data from the MW using a set of public HTTP APIs. Thanks to these interfaces, HL modules can be independently developed and integrated on the top of the MW in order to fulfill the specific needs of a target farm, without altering the core functionalities of LoRaFarM. Indeed, since the middleware is designed to decouple physical devices from back-end functionalities (and vice versa), the addition of HL modules is simple and requires a limited number of operations (as discussed for the VegIoT App in the following).

Two representative HL modules, related to a web dashboard and the VegIoT Garden mobile App, which have been developed and integrated in LoRaFarM, are presented in the following sub-sections. Furthermore, an additional HL module, aiming at forecasting future air temperature values within the greenhouse, based on a NN algorithm and air temperatures gathered with LoRaFarM inside the greenhouse, is presented in Section 4.3.

Web Dashboard

As anticipated, a web-based dashboard has been deployed as an HL module to enable the farmer (as well as other end-users of the system, e.g., developers) to monitor and visualize farm data. Furthermore, the dashboard shows devices installed in the farm together with their locations using a map, as displayed in Figure 3.19. Moreover, collected data are presented using plot charts, as shown in Figure 3.20, which displays soil temperature values collected by an EN of the vineyard module (SN-V1) during one day of September 2021.

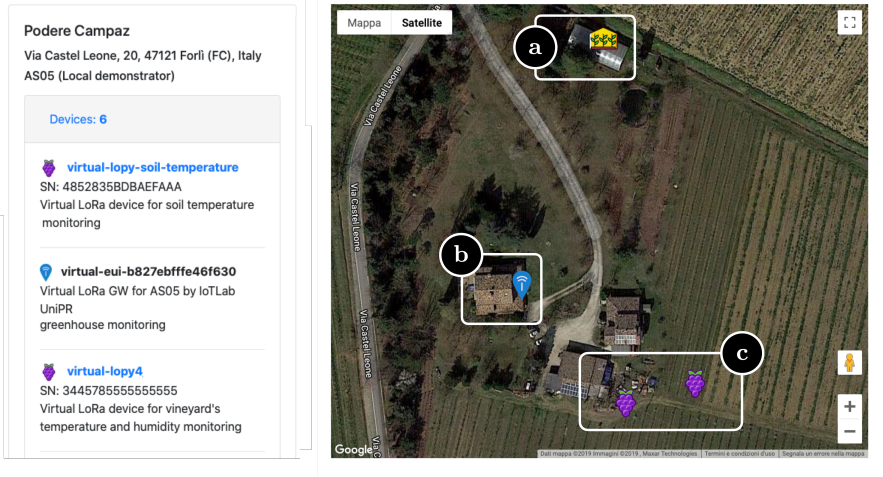


Figure 3.19: The developed web-based dashboard of LoRaFarM displaying the nodes installed in the farm together with their location: (a) two INs and a mp-GW in the greenhouse; (b) the LoRaWAN GW; and (c) two ENs in the vineyard.

VegIoT Mobile App Integration

An additional tool which can be integrated within LoRaFarM (along with the web-based dashboard) is the VegIoT Garden mobile App presented in Sub-section 3.1.6. As a brief recapitulation, the App allows the farmer (i) to visualize farm data with a smartphone; and (ii) to define alarms by which he/she can be notified, if sensor data reach values beyond determined thresholds. The VegIoT App can be easily integrated within LoRaFarM developing a CoAP Server module, which gathers farm data from the middleware using the provided HTTP APIs and made them available in CoAP to the mobile App.

3.2.6 Experimental Results

As for the VegIoT Garden system (Sub-section 3.1.7), LoRaFarM has been experimentally evaluated in terms of collected data and energy consumed by the system's battery-powered nodes during a validation campaign of 6 months. As

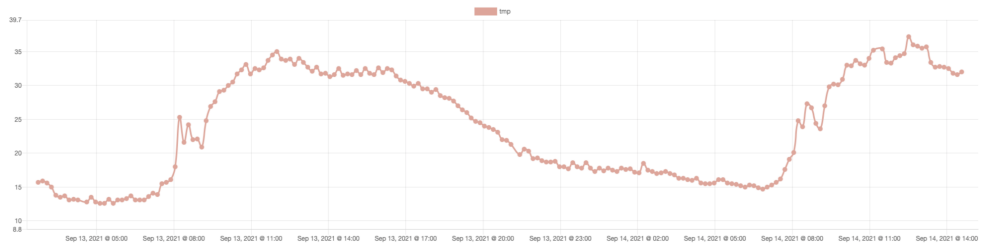


Figure 3.20: Soil temperature values collected during 1 day by SN-V1 (i.e., an EN of the vineyard module).

outlined in Sub-section 3.2.3, two LL modules, each based on two devices collecting sensor data related to environmental variables to be monitored and coming from agricultural scenarios (i.e., two INs and two ENs, for greenhouse and vineyard modules, respectively), have been deployed in the evaluation farm (Podere Campáz). Sensor data were periodically sampled every 10 min, stored into a MySQL database (at MW level) to be further analyzed, and made accessible to HL modules via HTTP APIs. Experimental results related to (i) the collected data and (ii) the lifetime of battery-powered sensor nodes of LoRaFarM (evaluated both theoretically and experimentally) are presented in the following sub-sections.

Collected Data

As said, two INs (namely, SN-G1 and SN-G2) have been deployed in the greenhouse to measure air humidity and temperature, and two ENs have been installed in the vineyard to monitor soil temperature and humidity (namely, SN-V1 and SN-V2). Data, which are presented and discussed in the following sub-sections, are gathered by the deployed nodes with a sampling interval of 10 min. The numbers of packets which have been successfully transmitted by the deployed nodes, lost packets, the corresponding success rate, the distance from the GW, and the employed spreading factor (fixed to SF7) are summarized in Table 3.5.

As reported in Table 3.5, the success rate of transmitted packets during the 183 days of test is approximately equal to or higher than 80% for almost every node,

Node	Location	Received	Lost	Total	Success Rate	Distance from GW [m]	Spreading Factor
SN-G1	Greenhouse (air)	18429	7923	26352	69.93 %	93	SF7
SN-G2	Greenhouse (air)	22325	4027	26352	84.72 %	90	SF7
SN-V1	Vineyard (soil)	20955	5397	26352	79.52 %	120	SF7
SN-V2	Vineyard (soil)	21525	4827	26352	81.68 %	85	SF7

Table 3.5: The performance of the deployed nodes in terms of received and lost observations during the 183 days of test.

except for SN-G1, which reaches a success rate around 70%. This can be justified considering the working conditions and the location where the SN-G1 has been deployed, namely, in the greenhouse, close to plants' leaves. Indeed, due to the position of SN-G1, the sunlight is received by the node's solar panel with suboptimal conditions: this influences the recharge of the node's battery. Moreover, the insufficient energy negatively impacts on data transmission. Furthermore, although SN-G2 and SN-G1 share the same scenario, the position of SN-G2 (namely, far from the surrounding vegetation) allows to better recharge the node's battery, thus, leading to an higher success rate, which is approximately equal to 85%.

Greenhouse Collected Data

Air humidity values collected by SN-G1 during 10 days of test are shown in Figure 3.21. Predictably, air humidity is higher than 70% during evening and night hours (from 18 pm to 6 am): in this time period the greenhouse is closed, so no external air can enter inside it. Conversely, air humidity falls from around 6 am to 12 am and then increases again, resulting in a periodic cycle on a daily basis. As a last comment, the farmer manually opens the greenhouse every day at about 6 am: this motivates the abrupt reduction of air humidity around 6 am.

Vineyard Collected Data

Figure 3.22 displays soil temperatures collected by nodes deployed in the vineyard (namely, SN-V1 and SN-V2) during approximately 3 months. As shown in

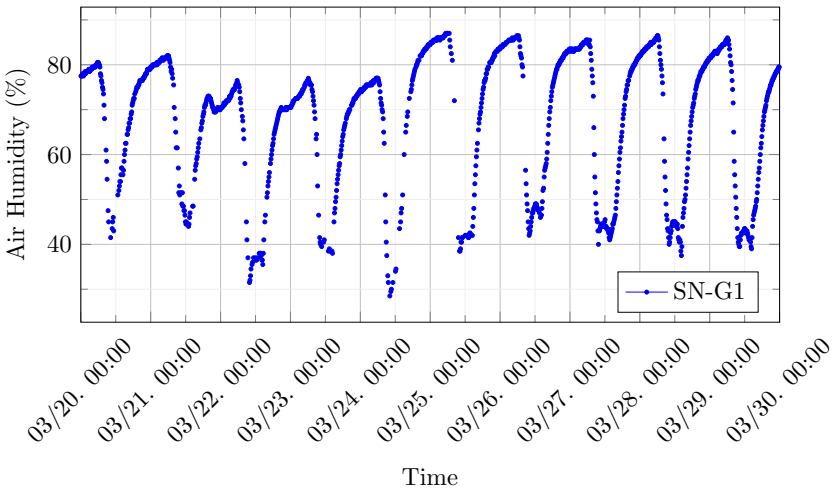


Figure 3.21: Air humidity values collected within the greenhouse during ten days of test (by SN-G1).

Figure 3.22, a similar periodic trend on a daily basis is followed by data coming from SN-V1 and SN-V2. More precisely, higher values of soil temperature are reached during night hours, otherwise, lower values during daylight hours. This consideration is valid during all three months of testing (i.e., July, August, and September 2019). Moreover, data gathered by SN-V1 and SN-V2 seem to be vertically shifted by approximately 1 °C in the first part of the plot chart. Conversely, in the other half of the plot, the divergence between node's soil temperature values decrease.

As an additional analysis, the daily minimum, maximum, and average soil temperature for vineyard's nodes have been computed starting from collected data. More precisely, since the same trend is followed by soil temperatures collected by both SN-V1 and SN-V2, one node has been randomly selected, i.e., SN-V1, and its collected data analyzed. The obtained results are shown in Figure 3.23.

Furthermore, the same approach has been applied to the daily temperature variation, which has been calculated, for each day, as the difference between maximum and minimum soil temperature collected by SN-V1. As a result, the daily

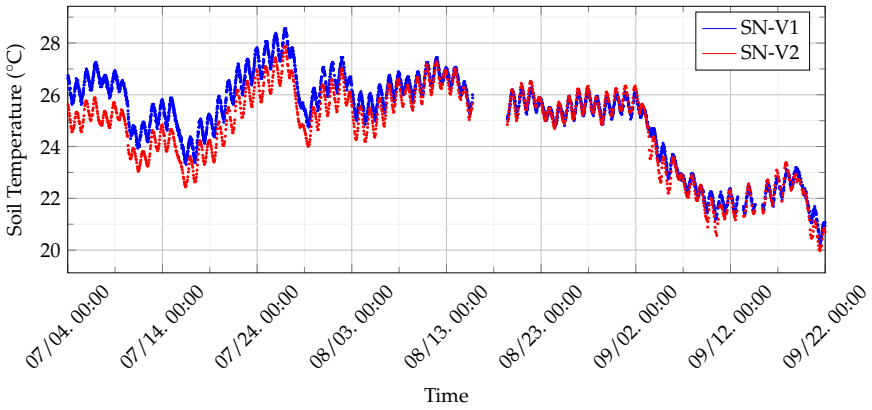


Figure 3.22: Soil temperatures coming from vineyard's nodes (i.e., SN-V1 and SN-V2), which have been collected during three months of test in 2019.

soil temperature variation tends to decrease from July to September, as displayed by the plot chart in Figure 3.24.

Energy and Power Consumption

The deployment of IoT devices in scenarios lacking of stable power sources, as agricultural environments, has to address relevant challenges in terms of power consumption. Indeed, selecting a suitable power source to feed devices and optimizing the usage of the energy provided by the selected source, during the lifetime of the device, are two key aspects to deal with. Moreover, for what concern IoT outdoor applications, such as SA, since power outlets are generally not available in these scenarios, powering devices using them is, thus, not a viable solution. In these contexts, more "constrained" power sources, including batteries and/or power scavenging techniques, such as, electromagnetic waves and sunlight, are (usually) the unique valuable options. Moreover, the energy supplied by the above-mentioned limited power sources, in terms of output power, has to be meticulously managed to prevent, for instance, the need of replacing batteries too often. The above presented considerations are applicable to LoRaFarM and, in particular, to the power source

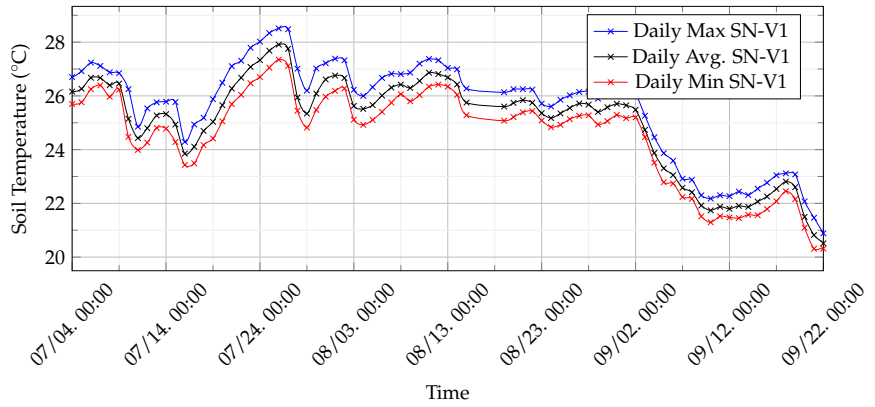


Figure 3.23: Minimum, maximum, and average daily soil temperatures collected by SN-V1 during three months of 2019.

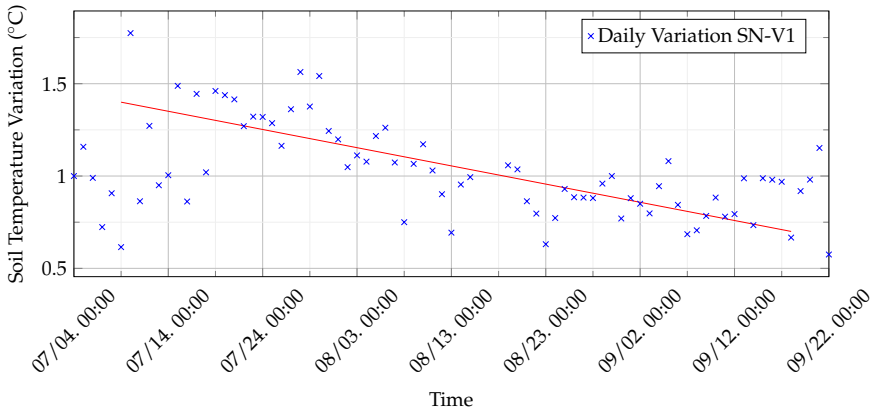


Figure 3.24: Daily soil temperature variation (i.e., differences between daily maximum and minimum temperature) of SN-V1's data during three months of 2019.

management of its sensor nodes. Indeed, they are installed outdoor, are powered by LiPo batteries connected to a solar panel (as outlined in Sub-section 3.1.3), and need to operate for several months without human interventions.

Moreover, in the following sub-sections are investigated: (i) the energy autonomy

State	Mean current [mA]		Duration [s]	
	End Nodes	Inner Nodes	End Nodes	Inner Nodes
Activity	$I_{EN_{\text{active}}} = 73$	$I_{IN_{\text{active}}} = 47.2$	$t_{EN_{\text{active}}} = 7$	$t_{IN_{\text{active}}} = 9$
Inactivity/sleep	$I_{EN_{\text{sleep}}} = 0.05$	$I_{IN_{\text{sleep}}} = 9$	$t_{EN_{\text{sleep}}} = 600$	$t_{IN_{\text{sleep}}} = 600$

Table 3.6: Experimental results on mean current consumption and time duration of LoRaFarM's sensor nodes during their operational phases (activity and inactivity/sleep).

and (ii) the average current consumption of the nodes of LoRaFarM which are battery-powered (namely, INs and ENs). In particular, first, the theoretical estimation of a node's lifetime while powered only by a LiPo battery is evaluated; then, experimental results concerning the joint adoption of the LiPo battery and the solar panel are presented. To be more clear, is remarked that (as stated in Sub-section 3.1.7) with energy autonomy is intended the working autonomy of a sensor node, considering its LiPo battery capacity and operational pattern. Otherwise, from a time point of view, the energy autonomy corresponds to the node's lifetime.

Theoretical Energy Autonomy Analysis

The lifetime of LoRaFarM's sensor nodes, in other terms, the number of working days until their batteries are totally discharged, when powered only by a LiPo battery, has been computed for both greenhouse's INs and vineyard's ENs, according to the methodology followed in Sub-section 3.1.7. This in order to acquire more knowledge on the energy requirements of LoRaFarM's battery-powered devices.

First, the INs and ENs' average current consumed during their operational phases, namely, activity and inactivity/sleep, together with their time duration, have been experimentally measured and are summarized in Table 3.6.

Second, the average consumed currents by, respectively, INs and ENs, can be computed with following expressions:

$$I_{IN_{\text{mean}}} = \frac{t_{IN_{\text{active}}} \cdot I_{IN_{\text{active}}} + t_{IN_{\text{sleep}}} \cdot I_{IN_{\text{sleep}}}}{t_{IN_{\text{active}}} + t_{IN_{\text{sleep}}}} \approx 9.57\,\text{mA}. \tag{3.3}$$

$$I_{EN_{\text{mean}}} = \frac{t_{EN_{\text{active}}} \cdot I_{EN_{\text{active}}} + t_{EN_{\text{sleep}}} \cdot I_{EN_{\text{sleep}}}}{t_{EN_{\text{active}}} + t_{EN_{\text{sleep}}}} \approx 0.891\,\text{mA}. \tag{3.4}$$

Assuming that devices are powered by a 3.7 V LiPo battery with a nominal capacity C_{LiPo}, their expected lifetime (namely, their energy autonomy) can be calculated as follows:

$$t_{EN/IN} = \frac{C_{\text{LiPo}}}{I_{EN/IN_{\text{mean}}}}. \tag{3.5}$$

For instance, with a battery having a capacity C_{LiPo} = 1800 mAh, the theoretical expected lifetimes, computed with Eq. (3.5), are around 84 days and 8 days for, respectively, an EN and an IN.

A significant difference between the estimated lifetimes of INs and ENs is outlined by the obtained results. Indeed, the lifetime of an EN is approximately an order of magnitude higher than the lifetime of an IN. This difference is justified by the characteristics of the boards around which sensor nodes are built, namely, CC for INs and LoPy4 for ENs. More precisely, the difference is related to the current consumed by the boards, mainly during their inactivity stage (see Table 3.6), which is 9 mA and 0.05 mA, for CC and LoPy4, respectively. In fact, since nodes are inactive for at least 98.5% of the time, the power consumed during their inactivity state is the principal factor which influences a node's lifetime.

To conclude, as already said in Sub-section 3.1.7, from an operational perspective, considering that the potential of a battery decreases with usage, its nominal capacity cannot be entirely employed by a connected node. Indeed, for all boards and devices, there exists a potential value, i.e., 3.5 V for the LoPy4 [121] and 2.5 V for the CC [125], under which they stop working properly. Due to this, the actual lifetime of a LoRaFarM's sensor node is lower than of that theoretical, as discussed in Sub-section 3.1.7.

Practical Energy Autonomy Considerations

The results presented in the previous sub-section and related to a node' lifetime motivate the usage of a solar panel to recharge a node's battery in order to prolong its lifetime in the order of months (and years). As discussed in Sub-section 3.1.7, one bright summer day is usually sufficient to fully recharge a 3.7 V LiPo battery of 1800 mA using a solar panel of 1 W. Moreover, even if nodes have to work in suboptimal environmental conditions, such as cloudy days and installation in a controlled environment as a greenhouse, the same recharging performance is generally accomplished in two or three days.

As a comment, considering the amount of energy which is hourly consumed by a LoRaFarM's device, namely, around 9/0.9 mA h, for, respectively, INs/ENs, it can properly work for several months without any problem. Indeed, although its battery completely runs out or power, for instance due to several successive days of bad weather, after the battery has regained a sufficient level of charge (due to one or more sunny days), a LoRaFarM node can restart forwarding sensor data without additional human intervention. These considerations have been actually validated, indeed, LoRaFarM's nodes have correctly forwarded data for 183 days, with no additional action, even in the presence of suboptimal weather conditions for the battery recharging.

3.2.7 Conclusions

The architectural framework proposed in Chapter 2 and the design principles on its basis have been successfully applied to build the LoRaFarM architecture, a general-purpose IoT/SA-oriented system aiming at supporting the management of arbitrary future smart farms. This is achieved with LoRaFarM effectively integrating heterogeneous IoT technologies, such as communication protocols and physical devices, and collecting, exchanging, processing, and visualizing data which are relevant for a target farm. Moreover, since LoRaFarM is scalable and modular, it can be easily customized according to the specific needs of a farm, thus, finding a wide applicability in the SA domain.

The above-mentioned interesting features of LoRaFarM have been achieved: (i) adopting the LoRaWAN architecture, as communication technology to enable connectivity in farms; (ii) building LoRaFarM around a core middleware layer; and (iii) enriching the core layer with *ad-hoc* modules in order to manage specific agricultural scenarios. Moreover, new modules can be independently developed, customized and integrated into the system, at high or low level, without altering its core structure, as done for the greenhouse module integrated in LoRaFarM. Indeed, this module has been derived from an already existent collection system, which is the GaWSN of VegIoT presented in Section 3.1, and seamless integrated in LoRaFarM.

The LoRaFarM architecture has been deployed in an Italian farm (Podere Campáz) and experimentally evaluated with a validation campaign of 6 months, in terms of collected data and nodes' energy efficiency. In detail, four sensor nodes (ENs and INs) have been installed in the greenhouse and in the farm vineyard in order to collect relevant environmental variables for vegetables and grape plants growth (namely, air and soil humidity and temperature). Furthermore, a web-based dashboard has been included with the goal of making collected data visualizable by the farmer.

To conclude, the deployed nodes, which are powered by a LiPo battery recharged by a solar panel, have correctly gathered data for 6 months, both in indoor (namely, greenhouse) and outdoor (namely, vineyard field) scenarios and regardless of bad climatic conditions. Indeed, despite the loss of some packets due to some consecutive rainy days, nodes restarted to transmit properly once their batteries have been recharged by solar light during the following bright days.

As a final remark, the system has properly worked for more than 183 days, which corresponds to the time period selected to perform the validation campaign outlined in Sub-section 3.2.6. Actually, the period of time during which LoRaFarM has been operational is around 2 years (namely, from June 2019 to September 2021). During this period of time the system has required approximately 3/4 on-field interventions per year to, for instance, install new devices, updated, or removing the already installed nodes, and check the GW's status. Due to this, it can be stated that the robustness of LoRaFarM is furthermore justified by the above-mentioned additional

evidences.

Future Research Directions

There exists several interesting future research direction for the LoRaFarM system, some of which are presented in the following. A *first* direction is related to the analysis of the collected data with advanced processing techniques based on, e.g., AI and ML. As an example, forecasting algorithms aiming at predicting the evolution of environmental variables can be developed to prevent plant diseases or to support the automation of agricultural scenarios (e.g., the Podere Campáz's greenhouse's sides opening). This last option is investigated in Chapter 4, in which two possible approaches to use to develop an air temperature forecasting algorithm aiming at supporting the management of greenhouses are presented.

A *second* direction is related to the software architecture of the middleware layer, which can be made even more modular by the introduction of a microservices-oriented architecture. Moreover, the MW layer can be re-built using the software components provided by the FIWARE framework, due to the reasons and provided advantages already discussed in Sub-section 1.2.2.

Finally, additional LL modules, eventually based on communication protocols not already integrated in LoRaFarM (e.g., BLE) and/or able to monitor other environmental variables, such as air quality, can be developed to expand LoRaFarM with new functionalities.

Chapter 4

IoT Data Analysis

This chapter covers the following three main topics. *First*, a brief overview concerning the integration of AI techniques and IoT is provided, with a focus on SA applications (Section 4.1). *Second*, an outline related to the key aspects of greenhouse production, including technologies to adopt to enhance it, is presented (Section 4.2). *Finally*, two possible approaches to be followed to build AI-based algorithms aiming at forecasting future trends of variables related to the internal climate of a greenhouse are proposed (Section 4.3).

4.1 Brief Overview

A big volume of heterogeneous data, differing in terms of typology, source of origin, and size, is usually generated and has to be managed by IoT applications. The processing and analysis of these highly valuable information, using, for instance, AI-based techniques, allows to create new value from raw data and, thus, to provide smart functionalities to the end-users of an IoT system (as discussed in Section 1.4). These considerations are even more true in the context of SA, where the introduction of IoT systems allows to collect but also process complex sets of heterogeneous data, which are relevant for a farm's production. With this prospective, the management of several agricultural activities and scenarios can be enhanced by the adoption of data mining

techniques and the development of data analysis algorithms (Sub-section 1.3.1).

As an example, greenhouses and, thus, their production processes, including, for instance, the scheduling of irrigation tasks and the control of the internal climate (e.g., inner air temperature and humidity), can benefit by the adoption of such advanced processing techniques. For this reason, a relevant management activity for greenhouse production, which is the regulation of the variables characterizing a greenhouse's internal climatic (with a focus on air temperature), is more deeply discussed in Section 4.2.

4.2 Greenhouse Production

Greenhouse production is one of the most important production fields of worldwide agriculture. In fact, since greenhouses allow to establish optimal growing conditions for indoor cultivation [126], several kinds of edible products can be grown anytime and anywhere inside greenhouses, without being influenced by their seasonality or the (possibly hostile) climatic conditions of their production area. Consequently, greenhouses cultivation provides several advantages, as the production of non-endemic food, the lengthening of cultivation periods of seasonal crops, and the reduction of needed resources (e.g., land, water, and pesticides) [84].

The creation and preservation of suitable growing conditions for the inside products is a crucial and challenging aspects of a greenhouse management. Indeed, the internal climatic conditions of a greenhouse (i.e., habitat, microclimate, or inner climate) is a complex set of environmental variables internal to the greenhouse (e.g., soil moisture, air humidity and temperature, and solar radiation) which influences the proper growth of inner products and depends on multiple interrelated factors [127]. In detail, these include (i) *internal* factors, such as the greenhouse' dimension and the type of cooling/warming systems internally installed (i.e., the greenhouse's actuators); and (ii) *external* factors (or "variables"), as weather meteorological conditions (e.g., solar radiation, wind speed, air temperature and humidity) [128].

4.2.1 Tech-based Management

Monitoring and regulating the inner climate of a greenhouse has been, nowadays, simplified and made automatic thanks to the joint adoption of a set of heterogeneous technologies, which can be classified according to the following three categories.

IoT-based Data Collection Systems. Relevant environmental variables internal to the greenhouse, which have to be maintained within certain intervals (e.g., air temperature and humidity), are measured by devices equipped with sensors (namely, sensor nodes). These devices are usually organized as WSN. Sensor data are forwarded to the Cloud (usually thanks an intermediary GW connected to the Internet) [129], where they are stored, can be retrieved and visualized, as well as used as input data for further processing. The monitoring of relevant variables inside greenhouses, which is important for both end-users (farmers) and researchers [13, 130–133], can be achieved deploying IoT systems.

Actuators and Control Systems. The above-mentioned IoT collection systems can be enhanced introducing additional control devices, namely, actuator nodes, deployed inside the greenhouse to control its inner climate [134, 135]. For instance, whenever too high air humidity values are measured by SNs, a ventilation system would be automatically activated to decrease air humidity.

AI-based Algorithms. Since the forecast of future values of monitored environmental variables of a greenhouse can allow, for example, to preemptively schedule some operations (as the activation of a heater system, in order to avoid internal variables reaching unwanted values, i.e., too low temperatures), complex models or forecasting algorithms having this purpose can be developed. In this context, the development of ML algorithms, for instance, built around NNs, allows to forecast the greenhouse's internal variables, selecting as inputs data collected from different sources [85–87]. Some examples of valuable data sources, which can be selected to design the above-mentioned models, may be internal or external variables of a greenhouse (eventually measured by SNs).

To conclude, nowadays, the execution of AI algorithms is moving from the Cloud to the edge due to the several advantages presented in Sub-section 1.4.4. This trend, namely, EdgeAI, can be applied also to SA applications and, in particular, to greenhouses management, as explained in the following section (Section 4.2.2).

4.2.2 IoT/EdgeAI-based Management

In the field of greenhouses production, EdgeAI algorithms (namely, AI algorithms run by edge devices) based on ML techniques can be used (i) to estimate the current values of a greenhouse's internal variables; and (ii) to forecast their future values. For instance, in the first case, the values of some internal variables (e.g., air temperature), which are usually measured by SNs deployed inside the greenhouse, may sometimes be not available and, thus, have to be estimated. As an example, if their batteries are completely drained, SNs cannot properly measure values of inner air temperature and, thus, these values are unknown. Moreover, the values of missing sensor data have to be estimated whenever knowing these values enable to control the greenhouse's inner climate (i.e., activating or deactivating cooling/warming systems). With respect to the second application, forecast future values of internal variables allows to, e.g., wisely schedule some operations, as activating a cooling system, whenever greenhouse internal future air temperatures are predicted to be too high.

A possible EdgeAI-based approach aiming at enhancing the management of greenhouses and the regulation of their inner variables is presented in [14, 15]. More precisely, the two papers propose two techniques, based on the integration of ML, edge computing, and IoT, aiming at developing a constrained device-friendly air temperature forecasting model and deploying it on an edge device, label as "Smart Gateway" (Smart GW). In detail, the Smart GW, which is enriched with "edge intelligence", acts as a GW (i.e., sink node and border router) for a WSN installed in a greenhouse and is intended to collect inner air temperature values. This configuration allows to (i) monitor the inner air temperature of a greenhouse through an "edge intelligent" approach (and a deployed WSN); (ii) locally forecast future values of inner air temperature; and (iii) pilot the internal actuators in order to regulate the greenhouse air temperature, according to the obtained results. The just

described approach is deeply discussed in Section 4.3, together with two possible methodologies to be followed to develop the above-mentioned forecasting models and to deploy them on a Smart GW.

4.3 Greenhouses Variables Forecasting Algorithms

As discussed in Sub-section 4.2.2, the joint adoption of IoT and ML algorithms (in the field of greenhouse's management) simplifies the control of a greenhouse's internal variables, moreover, making it automatic.

According to literature papers, the following three main stages can be undertaken to enhance the management of a greenhouse. *First*, a WSN aiming at collecting and monitoring a subset of relevant environmental variables (i.e., the greenhouse inner variables) is installed inside the greenhouse [130, 131, 133]. *Second*, one or more control systems, built around devices able to pilot the greenhouse's actuators, are introduced to automatically maintain suitable internal conditions [134, 135]. *Finally*, future trends of the internal variables are forecast with ML algorithms, as NNs [84–89].

Along with more consolidated techniques—e.g., physical methods based on mathematical theory and black box approaches based on modern computational technology, as, Particle Swarm Optimization (PSO)—the task of greenhouse air temperature forecasting has been successfully accomplished with NNs. Indeed, since they can learn patterns from data related to non-linear systems without having an a-priori knowledge of the system's model, NNs have become very popular [127].

4.3.1 Research Gaps

The algorithms proposed by the literature works discussed in Sub-section 1.4.3 (and summarized in Table 1.9) are intended to be executed in the Cloud, where computing and memory capabilities sufficiently high to run the model and store its parameters are available. Furthermore, although literature papers accurately forecast air temperatures (e.g., with a RMSE lower than 1 °C), while developing ML algorithms to be deployed on edge devices, performance optimization has

to consider the computational resources required by the algorithm to be run by the device. In fact, while deploying ML algorithms on devices having limited capabilities (with respect to the Cloud), normally, a fair trade-off between model's performance and its computational requirements has to be found. For this reason, although their prediction performance is normally lower, lightweight forecasting models are usually more valuable options than high-performing but more complex ML models (e.g., a small ANNs is preferred to a model having many hidden layers and parameters). It is remarked that, works collected in Table 1.9 do not provide any information concerning the computational and architectural complexities of the proposed algorithms, which, conversely, are key elements in designing an EdgeAI-oriented algorithm. Indeed, since they are intended to be deployed in the Cloud (and not on edge devices), no discussion targeting the complexity of the proposed models are provided, since this is not a constraining aspect.

4.3.2 Proposed Approach

As outlined in Sub-section 4.2.2, the management of a greenhouse can be enhanced by the deployment of an air temperature forecasting algorithm at the edge. For this reason and since this task has not been already satisfactory explored in literature, two different approaches to be followed to develop an edge device-friendly air temperature forecasting model (based on NNs) are investigated later in this chapter. More precisely, both a pure ML approach and time series-oriented approach are evaluated to develop the algorithm and are, respectively, presented in Sub-section 4.3.4 and Sub-section 4.3.5.

In detail, in Sub-section 4.3.4, methodological steps followed to build a simple ANN-based air temperature forecasting model, which takes as input variables a set of parameters related to the external environmental conditions of the greenhouse, are presented. Moreover, in Sub-section 4.3.5, stages undertaken to build a NN-based air temperature forecasting algorithm, taking as input variables the past and present values of the air temperature to forecast the future ones, are outlined. More exactly, for this last approach, three different NN architecture types—i.e., RNNs, LSTM Networks and ANNs—with various values of the sliding window associated

with input data, are considered. Furthermore, in Sub-section 4.3.5, models developed with the two methodologies are compared in terms of prediction performance and complexities, according to the evaluation metrics presented in Sub-section 4.3.3, and their deployment on a Smart GW is also discussed.

To conclude, the prediction performance of the developed models are compared with those of algorithms proposed in the literature. It will be shown that, even with a limited complexity, the performance degradation of the designed algorithms is limited with respect to Cloud-oriented approaches.

4.3.3 Evaluation Metrics

The prediction performance of a forecasting algorithm, compared to multiple models targeting the same prediction task, are generally quantified according to one or more evaluation metrics. In regression problems, as the forecasting of future air temperature values within a greenhouse, three broadly adopted evaluation metrics are the RMSE, the MAPE, and the R^2 [81], defined as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{j=1}^{n}(d_j - p_j)^2}{n}} \tag{4.1}$$

$$\text{MAPE} = \frac{1}{n}\sum_{j=1}^{n}\left|\frac{d_j - p_j}{d_j}\right| \cdot 100 \tag{4.2}$$

$$R^2 = \left[\frac{\sum_{j=1}^{n}(d_j - \overline{d})(p_j - \overline{p})}{\sqrt{\sum_{j=1}^{n}(d_j - \overline{d})\sum_{j=1}^{n}(p_j - \overline{p})}}\right]^2 \tag{4.3}$$

where: n is the number of predicted (and corresponding real) samples; d_j and p_j are the real and forecast values of the j-th sample of the phenomenon under observation (e.g., the air temperature), respectively; and $\overline{d} \triangleq (\sum_{j=1}^{n} d_j)/n$ and $\overline{p} \triangleq (\sum_{j=1}^{n} p_j)/n$ are the average values of the real and forecast samples, calculated as arithmetic average over all the n real and forecast samples, respectively.

The RMSE, defined as in Eq. (4.1), is the standard deviation of the difference between the real values (to be forecast) and the values predicted by the algorithm (or residuals). The closer to 0 the RMSE is, the better the performance.

The MAPE, defined according to Eq. (4.2), is a percentage value quantifying a relative between predicted and real values. As for the RMSE, the closer to 0 the MAPE is, the more accurate the model is.

Finally, the R^2, defined as in Eq. (4.3), is a normalized (between 0 and 1) statistical metric quantifying the adaptability of a regression model. The closer to 1 the R^2 is, the better the prediction model performs.

As anticipated in Sub-section 1.4.4, an additional metric to consider while developing NN models to be deployed on edge devices is the NetScore metric [95]. In detail, given a NN-based model $\mathcal{N}$, the NetScore metric, denoted as Ω is expressed as follows:

$$\Omega(\mathcal{N}) = 20\log\left(\frac{a(\mathcal{N})^{\alpha}}{p(\mathcal{N})^{\beta}\cdot m(\mathcal{N})^{\gamma}}\right) \tag{4.4}$$

where: $a(\mathcal{N})$ is the accuracy of the model (i.e., the correct prediction rate in a classification task); $p(\mathcal{N})$ is the number of parameters of a NN, also denoted as architectural complexity; $m(\mathcal{N})$ is the number of MAC operations executed by the NN during inference, also denoted as computational complexity; and α,β,γ are coefficients allowing to regulate the relative weights of accuracy, architectural complexity, and computational complexity of the model $\mathcal{N}$, respectively.

According to Eq. (4.4), the NetScore metric attributes to a NN-based model a score which is logarithmically propositional to the rate between prediction accuracy and model's complexity. This means that, for instance, high-accuracy models having a reduced number of parameters have an high NetScore value; conversely, models with moderate accuracy, but high complexity have a small NetScore value. Moreover, models with diverse accuracy, number of parameters, and number of MAC operations may have similar NetScore values.

Furthermore, the exponential coefficients α, β, and γ shapes the relative impact of the components of the logarithmic ratio of the NetScore (namely, accuracy, architectural and computational complexities). Indeed, according to their practical relevance for a target application, the weight of each factor can be adjusted modifying the values of α, β, and γ. For instance, if the NN model has to be extremely simply, in terms of computational and architectural complexities, larger values (than the default)

can be attributed to β and γ. In contrast, a high value of α can be fixed whenever the model's needs to be highly accurate. As a general rule, for most applications, α, β, and γ can be set to, respectively, 2, 0.5 and 0.5, in other terms, to their default values [95]. For this reason, the relevance of a model's accuracy is, normally, higher than the (computational and architectural) complexity of the model itself.

4.3.4 "Pure" ML Approach

The aim of this sub-section is to present a set of possible methodological steps to be followed to build a constrained device-friendly ANN-based greenhouse air temperature forecasting model adopting a pure ML approach. More precisely, the designed algorithm can forecast future values of air temperature internal to a greenhouse, taking as inputs a set of environmental variables related to the outside weather conditions. The algorithm has been developed according to the methodology shown in Figure 4.1 and more deeply discussed in the following.

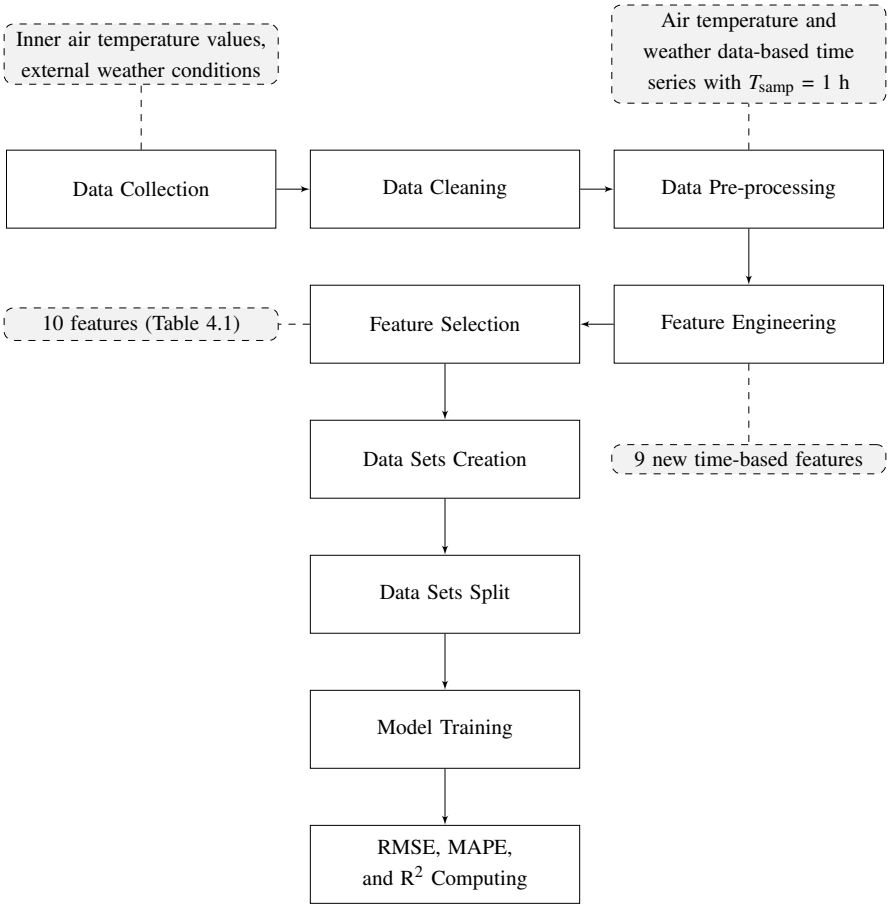


Figure 4.1: Methodological steps (white rectangles) and corresponding outcomes (gray rectangles with rounded corners) followed to build the greenhouse air temperature forecasting algorithm adopting a pure ML approach.

Data Gathering, Cleaning, & Pre-processing

In the data collection stage, the following two types of agricultural data have been gathered: (i) weather data, coming from the DarkSky weather repository [136]; and (ii) air temperature values collected with the LoRaFarM system (presented in Section 3.2), during a period of 10 months (from August 2019 to June 2020), in an

Italian greenhouse (namely, Podere Campáz [120]). The collected data (i.e., weather data and sensor readings) have been, then, timely aligned, thus, resulting in a time series with a frequency of one sample per hour.

Feature Engineering

According to a preliminary analysis based on the graphical visualization of air temperatures values measured inside the greenhouse, the time series associated to the internal air temperature have a daily seasonality. To be more clear, similar trends are shown by air temperatures of different days. Furthermore, hourly values of air temperatures depend on the month of the year and the season in which they are measured. A possible justification for this last trend is related to the fact that during months of the same season, a similar greenhouse sides' opening/closure pattern is adopted by the farmer to avoid air temperature and humidity reaching high values within the greenhouse.

The above-mentioned correlation among the trend of air temperature and the *hour*, *month*, and *season* in which data samples are measured can be leveraged to engineer 9 new features. Since these additional features can be derived from a time reference, they are denoted as *time-based* in the following. Moreover, along with features obtained from the gathered meteorological data, time-based features are considered as possible inputs for the forecasting model, in the stage of feature selection.

In detail, the additional 9 novel time-based features are engineered as follows. *First*, a time-based categorical variable (in other words, a variable which values belong to a finite—usually fixed—set), i.e., *hour*, *month*, and *season*, is labeled as v. The variable v can take an integer value $x \in [0, \dots, T-1]$, where T is the periodicity of v and corresponds to 24, 12, and 4, for hour, month, and season, respectively. As an example, for $v = season$, $x \in [0,1,2,3]$ refers to a season in $[winter, spring, summer, autumn]$.

Second, based on the sine-, cosine-, and 2-argument arctangent transformations, the following three trigonometric functions of the time-based variables presented

above are defined:

$$v_{\sin} = \sin\left(2\pi\frac{x}{T}\right) \quad (4.5)$$

$$v_{\cos} = \cos\left(2\pi\frac{x}{T}\right) \quad (4.6)$$

$$v_{\text{atan2}} = \text{atan2}\left(v_{\sin}, v_{\cos}\right). \quad (4.7)$$

The reason behind the introduction of the trigonometric functions of v expressed by Eqs. (4.5)-(4.7) is that they allow to encode the above-mentioned categorical variables v, namely, *hour*, *month*, and *season*, in numerical variables without altering their periodicity. Furthermore, with the presented trigonometric-based encoding, the transformed values of v numerically near to each other when associated with hours, months, or seasons remain close to each other in time. As an example, when $v = season$, $x = 0$ corresponds to winter and $x = 3$ corresponds to autumn: the two seasons are close to each other but the numerical values identifying them, i.e., 0 and 3, are not. Conversely, when considering the trigonometric transformations expressed by Eqs. (4.5)-(4.7), transformed values of 0 and 3, corresponding to 0 and -1, are closed to each other.

Finally, 9 new (time-based) features related to hour (label as $hour_{\sin}$, $hour_{\cos}$, and $hour_{\text{atan2}}$), month (denoted as $month_{\sin}$, $month_{\cos}$, and $month_{\text{atan2}}$), and season (label as $season_{\sin}$, $season_{\cos}$, and $season_{\text{atan2}}$), are obtained with the trigonometric transformations in Eqs. (4.5)-(4.7).

Feature Selection

A correlation analysis [137], aiming at identifying which features are most likely correlated with the greenhouse's air temperature, has been performed among (i) air temperature samples, measured within the greenhouse; (ii) weather data; and (iii) time-based features, added in the phase of feature engineering. Moreover, the resulting features have been selected as input variables for the ANN-based forecasting model. In detail, features having a value of absolute correlation (denoted as r) with the air temperature lower than 0.25 have been excluded; this threshold value has been selected based on the following experimental assessment.

Meteorological Feature	Correlation r	Time-based Feature	Correlation r
Apparent temperature	0.896	$hour_{\cos}$	-0.470
Dew point	0.690	$month_{\sin}$	-0.625
Air humidity	-0.424	$month_{\cos}$	-0.266
Air temperature	0.900	$season_{\cos}$	-0.652
UV index	0.659	$season_{\text{atan2}}$	0.263

Table 4.1: Selected input variables together with their correlation with greenhouse inner air temperature.

The prediction performance of the obtained forecasting model have been evaluated, in terms of RMSE, with different numbers of features, namely, from 26 to 1, discarded in descending order for r. Experimental results show that larger is the number of discarded variables with low correlation with the air temperature internal to the greenhouse (in this case of study), poorer are the prediction performance. For this reason, the threshold value of 0.25 is selected as a fair trade-off between the number of features used as inputs for the model (≤ 10) and prediction performance (RMSE $\leq 1.50\,^{\circ}$C). The 10 selected features, together with the relative values of the correlation r with greenhouse air temperature, are summarized in Table 4.1.

As shown in Table 4.1 and can be expected, the air temperature inside and outside the greenhouse are strongly and positively correlated. This is mainly due to the fact that the internal microclimate of the greenhouse is only partially controlled by the actuators placed inside it. In fact, air temperatures (and humidity) within the greenhouse are solely regulated by opening or closing the greenhouse's sides. Furthermore, during hot months (from May to August), the greenhouse's sides are open during all daylight hours. For these reasons, the external weather conditions strongly influence the inner climate of the greenhouse, which is, due to this, highly correlated with the air temperature outside the greenhouse.

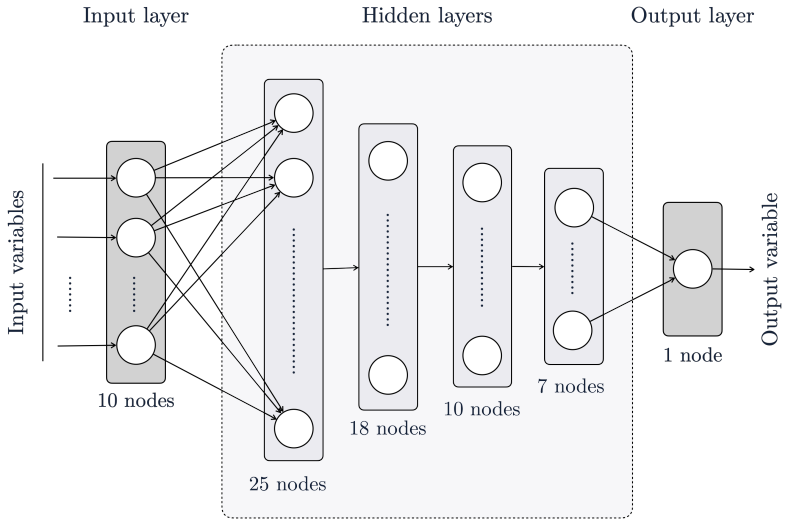


Figure 4.2: The architecture of the proposed fully-connected ANN.

Deployed NN-based Model

After the stage of feature selection, a data set including 5346 sample is obtained. Each sample in the data set is composed of (i) a 10-dimensional vector of input variables, namely, the features shown in Table 4.1; and (ii) an output variable, which is an air temperature value measured by a sensor inside the greenhouse. The data set is, then, randomly divided, with a proportion of 3:1, into a training set and a test set, respectively. Moreover, input variables are standardized, in other words, they are re-scaled so that their values have a mean equal to 0 and a standard deviation equal to 1.

The forecasting problem is solved adopting a pure ML (and not a time series-oriented) approach. This methodology allows to better deal with the forecasting of future values of air temperature while multiple successive sensor data are missed. The developed algorithm is based on a lightweight, in terms of computing capabilities and storage space requirements, a fully-connected ANN with 4 hidden layers and 1018 parameters. As discussed in Sub-section 1.4.3, along with ANN, RBF networks well-performs in the task of air temperature forecasting. Since RBF

networks adopt a weighted sum of non-linear functions (e.g., Gaussian functions) as neurons' activation functions, their execution requires, generally, more computational resources than the weighted sum performed by a canonical ANN node. Considering this last comment and that the algorithm in intended to be deployed on a constrained IoT device, the approach based on RBF networks has not been evaluated.

The architecture of the developed ANN, which has been trained with the BP algorithm and RMSE as loss function, is shown in Figure 4.2. To conclude, the k-fold cross-validation technique (with $k = 4$) has been adopted with the goal of finding the data set split (in other words, the subdivision of samples between training set and test set) which allows the model to reach the better prediction performance (in terms of RMSE).

Experimental Results

The proposed ANN-based model has been evaluated, in terms of prediction performance, over a test set of 1336 samples (gathered from August 2019 to June 2020) using the following three of the evaluation metrics introduced in Subsection 4.3.3: RMSE, MAPE, and R^2. According to experimental results, the developed model can forecast air temperature values with a RMSE, MAPE, and R^2 equal to 1.50 °C, 4.91%, and 0.965, respectively. This means that the air temperature can be forecast with a standard deviation of 1.50 °C with respect to the real value. If compared with the literature papers summarized in Table 1.8, which forecast air temperatures with a RMSE < 1 °C, then the RMSE of the developed algorithm is lightly higher (in the range of 0.5-1.0 °C) [87, 88] or comparable [89]. With respect to the R^2 metric, the obtained value of 0.965 is comparable with the results achieved in [88], but higher (and, thus, better) than the score in [89]. Moreover, considering the accuracy of the environmental sensor used to monitor air temperature, which is of ± 1 °C, and the typology of task, namely, monitoring a greenhouse's inner air temperature, the achieved prediction performance is satisfactory. Furthermore, the features of being lightweight and less computational-intensive then models proposed in the literature justify the slight performance degradation of the obtained algorithm.

Data collected from sensors during 5-days of August 2019, together with the air

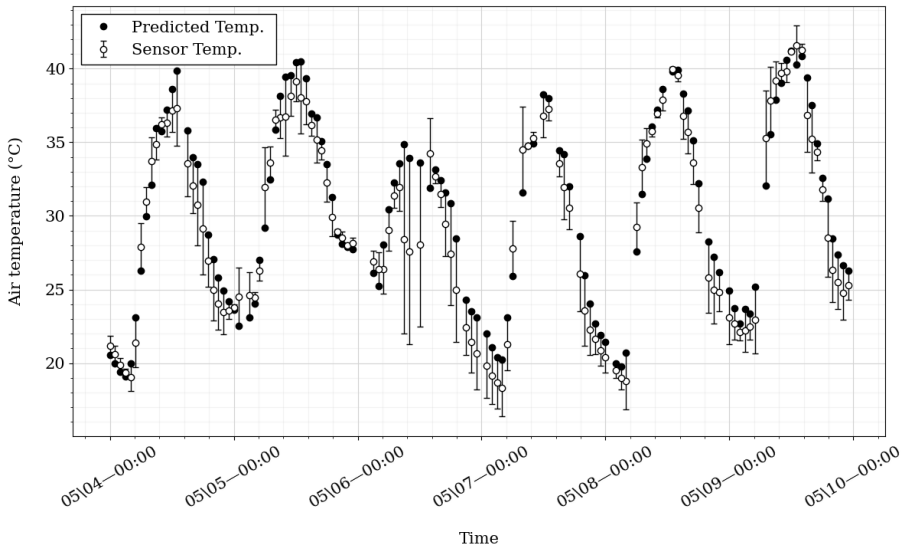


Figure 4.3: Forecast air temperatures compared with data gathered from sensors inside the greenhouse during 5-days.

temperature values forecast by the developed algorithm, are shown in Figure 4.3. Furthermore, the prediction performance of the same model, evaluated on the test set in terms of difference between the gathered sensor air temperature values and the forecast ones, are displayed with a plot chart in Figure 4.4. As can be seen in Figure 4.3 and Figure 4.4, an acceptable agreement is reached between collected (actual) sensor data and the relative forecast values.

Possible Application Scenario and Reference Architecture

For validation scope, the forecasting model has been successfully deployed on a real Smart GW, built around a RPi, which is a very popular board for EdgeAI applications [138, 139]. The Smart GW collects air temperature data from a SN based on a LoPy4 board [121] connected with a AM2302 temperature sensor [114] and placed in a greenhouse. The Smart GW collects sensor data with a sampling interval of 10 min and forwards these data to the Cloud. Moreover, if one or more

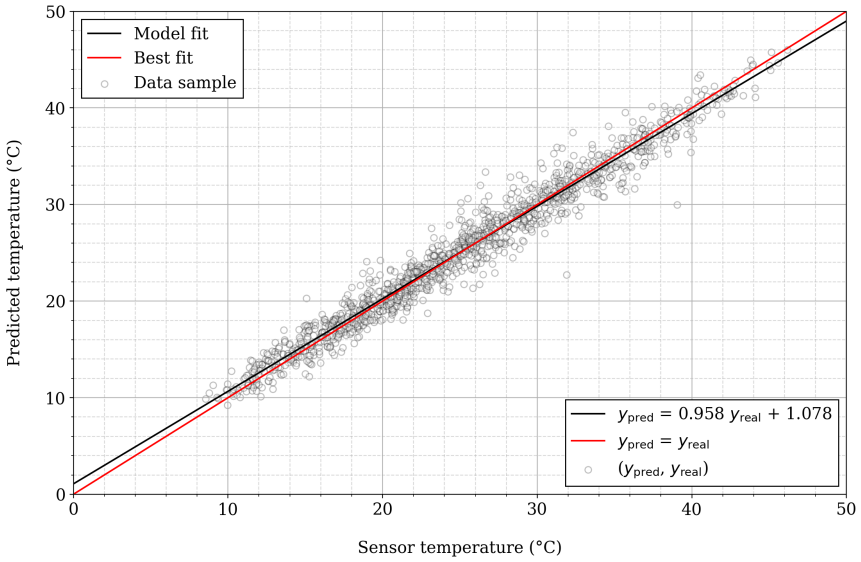


Figure 4.4: Air temperature data, measured by sensors, are compared with forecast values (respectively, y_{real} and y_{pred}) on the test set.

sensor data are lost, the Smart GW retrieves meteorological data from the DarkSky repository and process them in order to provide the proper inputs to the embedded ANN-based forecasting algorithm which, in turn, predicts the values of the missing data and forwards them to the Cloud. As a comment, along with the DarkSky repository, an alternative source of meteorological data, from which collecting the needed information, can be weather stations placed near the greenhouse (if available), as shown in Figure 4.5. Furthermore, the estimated value can be locally used to schedule the opening or closing of the greenhouse's sides, avoiding the need for another component, deployed at Cloud level, taking the decision and reporting it to the GW. The above-mentioned overall procedure is graphically summarized in Figure 4.5. With this approach, a greenhouse could be totally locally managed by a Smart GW, thus reducing network load and latency, since decisions are not taken in the Cloud.

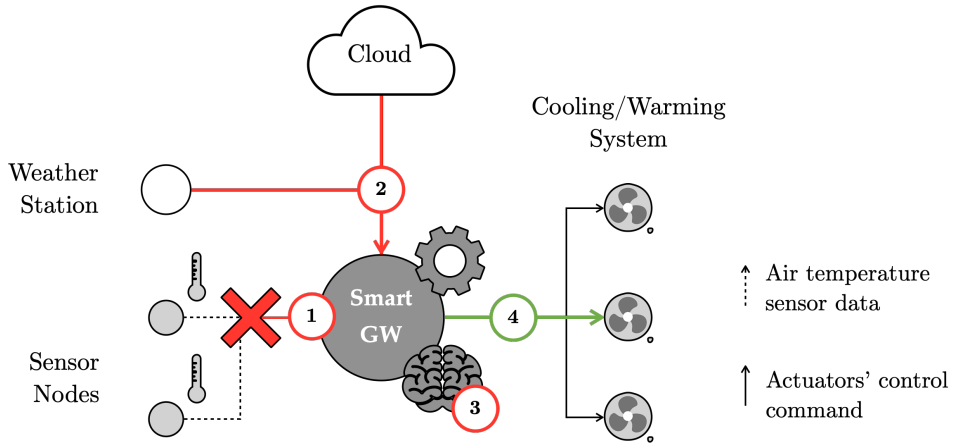


Figure 4.5: The Smart GW deployed inside the greenhouse together with the steps needed in order to automatically managed the inside actuators in case of missing sensor data.

4.3.5 Time Series-oriented Approach

An alternative approach to build the air temperature forecasting model (intended to be executed on the Smart GW) presented in Sub-section 4.3.4 is discussed in this sub-section. In detail, instead of selecting the external weather conditions of the greenhouse as input variables for the model to forecast the air temperature evolution, air temperature values gathered inside the greenhouse are evaluated as input variables for the model. Moreover, the performance of the model developed following this approach, label as time series-oriented, are evaluated in terms of prediction precision and model's lightweightness with respect to the model presented in Sub-section 4.3.4. Furthermore, how the model's parameters, i.e., the number of input variables (air temperature values), their sampling interval, and the type of NN architecture, effect the model's prediction performance is evaluated.

More precisely, three types of NN architectures, namely, ANNs, RNNs, and LSTM networks, which, according to the literature, outperform all other types of NNs for time series-based forecasting, are evaluated. Moreover, the impact on the prediction performance of the selected three architectures of the following two design

parameters is considered: (i) the number of model's input variables, in other words, the dimension of the data sliding window used by the model, denoted as SW; and (ii) their sampling period, denoted as T_{samp} (dimension: [min]).

To conclude, the developed models are compared with literature papers solving the same task (namely, air temperature forecasting inside greenhouses) and collected in Table 1.9, in terms of prediction performance, based on the RMSE, MAPE, and R^2. In contrast with the conventional approach followed by literature papers, a comparative performance analysis is performed in terms of computational and architectural complexities of the proposed model. In detail, the aforementioned goal is accomplished through the use of the NetScore metric, which allows to evaluate the complexity of the NN-based models to be run on constrained devices and, to the best of the author knowledge, has never been used before. Indeed, although it is rarely considered in the literature (especially) in the field of greenhouses' inner variables forecasting, comparing models in terms of NetScore metric is a key step to identify the more suitable models to be deployed on edge devices.

The methodological steps followed to build the algorithm, perform the above-mentioned analysis, and select the best EdgeAI algorithm to be deployed on the Smart GW are summarized in Figure 4.6 and further described in the following.

Overview of Followed Methodology

1. Relevant air temperature values collected with the LoRaFarM architecture (Section 3.2) in an Italian greenhouse (namely, the Podere Campáz [120]) are gathered.
2. Air temperature sensor data measured inside the greenhouse with a sampling period $T_{\text{samp}} = 10$ min are arranged in a time series. Moreover, from this starting time series, six novel time series are obtained downsampling the first time series with longer sampling periods.
3. The sampling period T_{samp} and the number of input variables of the model SW are defined as the two design parameters. Furthermore, T_{samp} is reintegrated as the prediction time horizon; indeed, the forecast temperature value is the one

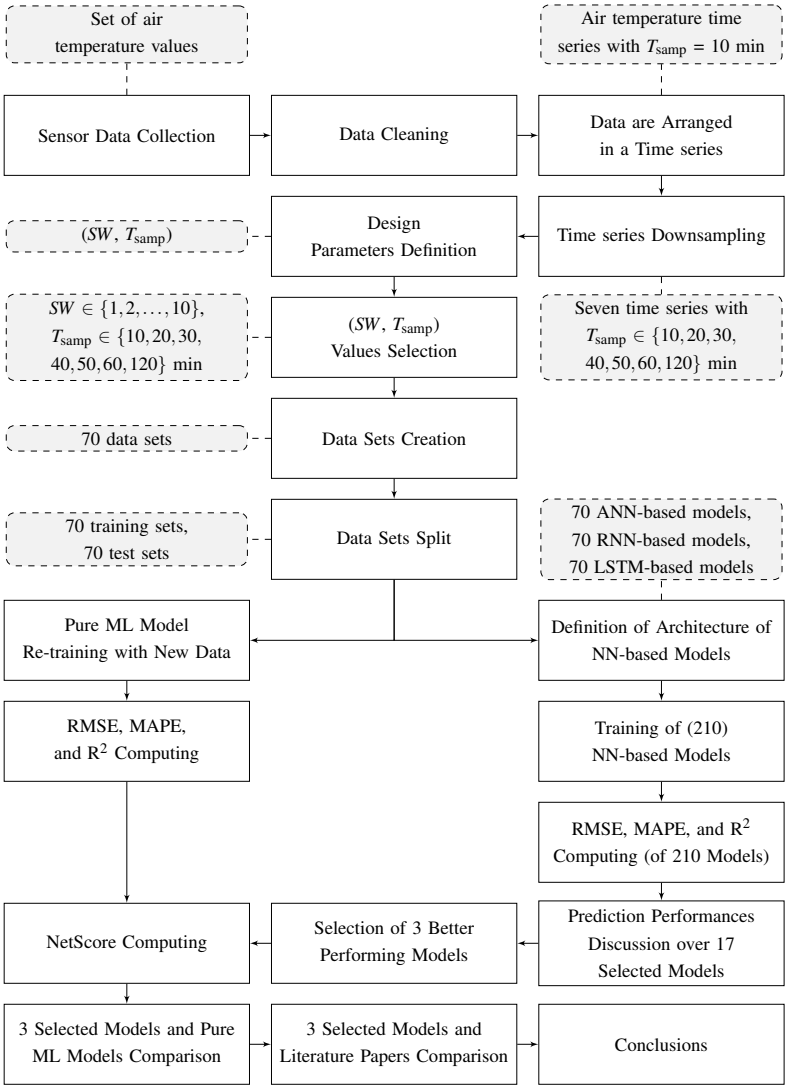


Figure 4.6: Methodological steps (white rectangles) followed to build a greenhouse air temperature forecasting algorithm based on a time series-oriented approach and corresponding outcomes (gray rectangles with rounded corners).

corresponding to the next temperature value after the most recent one of the sliding window: this samples is, by construction, T_{samp} ahead. Moreover, for testing purposes, a limited set of values related to these parameters is chosen.

4. Based on the gathered sensor data and in line with the number of parameters' values to be evaluated, a fixed number of data sets are built. Moreover, each data set is divided into training and test subsets.
5. Three selected NN architectures, built around an ANN, a RNN, and a LSTM, are design and trained with the data sets obtained in stages before.
6. The NN model presented in Sub-section 4.3.4 (for shortness, denoted as "pure ML model") is re-trained with a remarkably larger data set, which includes data from 6 additional months.
7. All models are evaluated on the test subsets in terms of RMSE, MAPE, R^2, and NetScore, and their performances in comparative terms.
8. To conclude, the best three models (over a total of 210) on the synthesized data sets (stage 4) are compared in terms of performance with literature papers (listed in Table 1.9).

Data Collection and Cleaning

As anticipated, sensor data related to air temperatures collected inside the greenhouse have been gathered with the LoRaFarM architecture (presented in Section 3.2) inside the greenhouse of the Podere Campáz [120]. More precisely, air temperatures have been measured with a sampling interval of 10 min, during a period of time of 16 months (from the beginning of August 2019 to the end of November 2020). The distribution of the data gathered during the considered period of 16 months is shown in Figure 4.7: for each month, the average number of daily collected data (over the month) is indicated.

As can be seen analyzing the data distribution displayed in Figure 4.7, data have been irregularly gathered through the months of the collection period. Such unbalanced distribution can be justified by several reasons. As an example, a

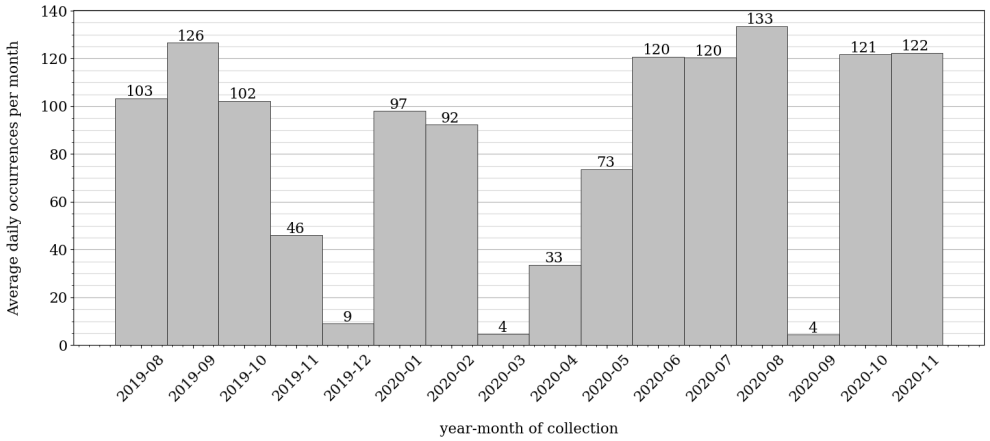


Figure 4.7: Sensor data gathered during a time period of 16-months; in each month, the average daily number of gathered samples, computed as the ratio between the number of collected samples per month normalized and the number of days of the month, is shown.

temporary lack of Internet connectivity, preventing the proper forwarding of data collected thought LoRaFarM to a storage repository placed in the Cloud caused the irregularity occurred during December 2019, March 2020, and September 2020.

Conversely, the distribution of data samples over actual collection daily hours has a nearly perfect uniform trend. This means that the number of data gathered during diverse hours of the day in the time period of 16-months is approximately the same at each hour. Indeed, considering that the total amount of "raw" data collected during this stage is $N_{\text{samp}} = 40033$ samples, the number of hourly gathered samples changes from a minimum of 1649 samples to a maximum of 1694 samples, with an average value of 1668 samples and a standard deviation of approximately 11 samples.

Engineering Time Series from Sensor Data

As mentioned in previous sub-section, raw (sensor) data were gathered with an actual sampling period $T_{\text{samp}} = T_0 = 10$ min. The derived time series, which

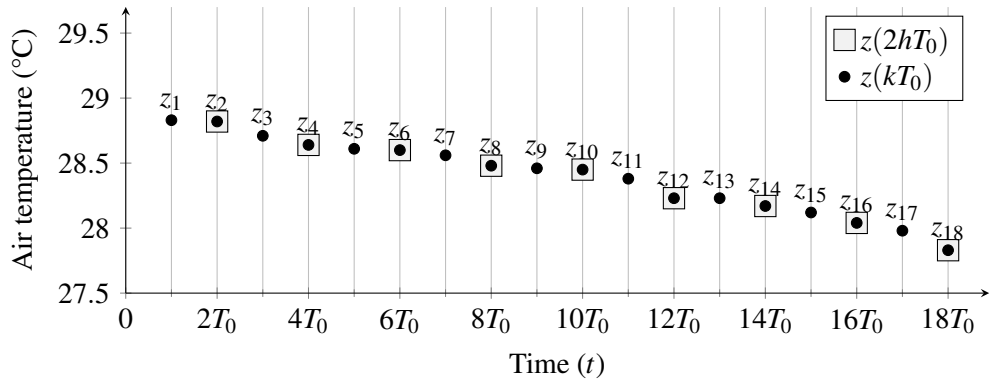


Figure 4.8: Explanatory representation of the first 18 samples $\{z_k^{(T_0)} = z(kT_0)\}_{k=1}^{18}$ (black dots) and of the first 9 samples $\{z_h^{(2T_0)} = z(2hT_0)\}_{h=1}^{9}$ (black dots over gray squares), obtained downsampling $\{z_k^{(T_0)}\}$ with a factor equal to 2.

represents the air temperature trend measured inside the greenhouse during the considered 16 month time period, is termed as $\{z_k^{(T_0)}\}_{k=1}^{N_{\text{tot}}}$, where k corresponds to the instant of time kT_0—denoting T_0 (with $k = 1$) as the instant of time in which the first sample was measured and $N_{\text{tot}}T_0$ (with $k = N_{\text{tot}}$) as the instant of collection of the last sample—or, with a concise notation, merely $\{z_k^{(T_0)}\}$. More specifically, denoting as $z(t)$ the air temperature inside the greenhouse, the sample $z_k^{(T_0)} = z(kT_0)$ of time series corresponds to the air temperature measured at the instant of time $t = kT_0$. For clarity, since it refers to the particular actual data gathered from sensors, a deterministic temperature signal $z(t)$ has been considered in this study. (To make this approach more general, the air temperature should be shaped as a stochastic process $Z(t)$. Nevertheless, this is out of the scope of this doctoral thesis.) An explanatory representation of the first 18 samples of the time series $\{z_k^{(T_0)}\}$ (black dots), labeled as $\{z_k\}$ for compactness, is shown in Figure 4.8.

It is remarked that, due to the approach adopted to collect sensor data within the greenhouse, some samples of the time series $\{z_k^{(T_0)}\}$ may not be accessible. Indeed, since LoRaFarM may experience temporal Internet connectivity loss, some sensor data can be eventually missed. Due to this, there exist N_{lost} instants of

time $(\hat{k}_1 T_0, \ldots, \hat{k}_{N_{\text{lost}}} T_0)$ at which the values of air temperature $\{z_{\hat{k}_i}^{(T_0)}\}_{i=1}^{N_{\text{lost}}}$ are not accessible—for instance, $z_{\hat{k}_i}^{(T_0)} = \texttt{NaN}$ can be set, instead. consequently, the total number of samples in the time series, labeled as N_{samp}, is smaller than the overall number of samples N_{tot} in theory measured in the considered period of 16 months with a sampling period equal to T_0 = 10 min and without data loss. In detail, $N_{\text{tot}} = N_{\text{samp}} + N_{\text{lost}}$.

In the field of time series forecasting and analysis, downsampling is a valuable technique aiming at reducing the number of samples in a time series; moreover, indicating with λ the downsampling factor, the downsampled time series can be expressed as:

$$\{z_h^{(\lambda T_0)} = z(\lambda h T_0)\}_h = \{z_{\lambda h}^{(T_0)}\}_{\lambda h} \qquad h = 1, 2, \ldots$$

For that purpose, the downsampled version, with $\lambda = 2$, of the primary time series, together with the related samples identified through circles with squares, is shown in Figure 4.8.

With the goal of analyzing the way the sampling period T_{samp} impacts on the prediction performance of the model, 6 additional time series are engineered downsampling the primary time series $\{z_k^{(T_0)}\}$ with T_{samp} equal to $2T_0 = 20$ min, $3T_0 = 30$ min, $4T_0 = 40$ min, $5T_0 = 50$ min, $6T_0 = 60$ min, and $12T_0 = 120$ min, in other terms, by selecting as downsampling factor λ equal to 2, 3, 4, 5, 6, and 12, respectively. The obtained time series are labeled as $\{z_k^{(2T_0)}\}$, $\{z_k^{(3T_0)}\}$, $\{z_k^{(4T_0)}\}$, $\{z_k^{(5T_0)}\}$, $\{z_k^{(6T_0)}\}$, $\{z_k^{(12T_0)}\}$, respectively.

A total of 70 different data sets are created starting from the engineered time series. In detail, for each time series, 10 data sets are created, having a different number of input variables each, as more deeply discussed in the following subsections. The construction of these time series allows to discover which is the best sampling period for the phenomenon under investigation, i.e., greenhouse inner air temperature.

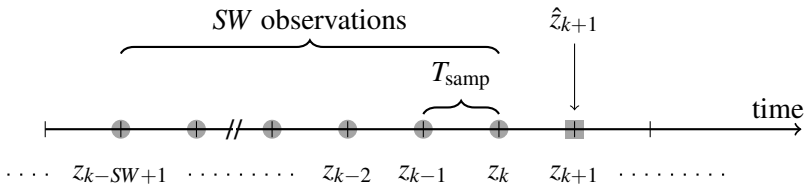


Figure 4.9: The forecasting at epoch k is based on a sliding window: the SW observations $\{z_{k-i+1}\}_{i=1}^{SW}$ are employed to forecast the value z_{k+1}, labeled as $\hat{z}_{k+1}$ (for simplicity, the superscript $^{(T_{\text{samp}})}$ is omitted).

Sliding Window-based Prediction

As already discussed, the adopted methodology is based on the usage of a sliding window. More precisely, the values of SW successive samples of the target time series, with sampling interval T_{samp}, are considered in order to forecast the value of the next sample. With respect to Figure 4.9, at epoch k:

$$\hat{z}_{k+1}^{(T_{\text{samp}})} = \hat{z}_{k+1}^{(T_{\text{samp}})}\left(z_{k-SW+1}^{(T_{\text{samp}})}, \ldots, z_{k}^{(T_{\text{samp}})}\right)$$

where $\hat{z}_{k+1}^{(T_{\text{samp}})}$ represents the predicted value of the (actual) sample $z_{k+1}^{(T_{\text{samp}})}$ and, on the right side, the dependency from the previous samples is underlined.

The prediction performance of the forecasting model depends on the design parameters SW and T_{samp}, for this reason, their optimal value have to be discovered. This means that multiple combinations of SW and T_{samp} are considered, and the corresponding prediction performance compared among each others. More precisely, the following values have been considered: (i) 10 values of SW, i.e., from 1 to 10 samples; and (ii) 7 values for T_{samp}, namely, 10, 20, 30, 40, 50, 60 and 120 min.

It is remark that, since SW shapes the number of the model's inputs (corresponding to the parameter ℓ in Figure 1.5 for a neuron belonging to the input layer), a SW-dimensional vector of SW features is provided at the input of the model. Moreover, the features of the aforementioned SW-dimensional vector are air temperature values sampled with a set period T_{samp} which, thus, come from the time series $\{z_k^{(T_{\text{samp}})}\}$ engineered in the earlier methodological step. For instance, a model

developed choosing $SW = 5$ and $T_{\text{samp}} = 10$ min is fed with input vectors having 5 features corresponding to 5 successive samples of air temperature gathered every 10 minutes (T_{samp}).

Data Pre-Processing and Data Sets Creation

With the goal of evaluating in comparative terms the model's prediction performance selecting different combinations of $\{SW, T_{\text{samp}}\}$ pairs, more than one data sets have to be created (in order to train and test the algorithm). In detail, an overall number of 70 data sets have been engineered. Each data set is linked with a target $\{SW, T_{\text{samp}}\}$ couple, where $SW \in \{1, 2, \ldots, 10\}$ and $T_{\text{samp}} \in \{10, 20, 30, 40, 50, 60, 120\}$ min. More precisely, adopting a compact notation, the data set engineered using the corresponding values of T_{samp} and SW is denoted as $\mathscr{D}_{SW}^{(T_{\text{samp}})}$. Each element of the data set is a vector $\underline{d}(k)$ of real air temperature values, which is defined as follows:

$$\underline{d}(k) \triangleq \left(z_{k-SW+1}^{(T_{\text{samp}})}, \ldots, z_{k}^{(T_{\text{samp}})}, z_{k+1}^{(T_{\text{samp}})}\right).$$

With a shorter notation, the vector $\underline{d}(k)$ can be express as:

$$\underline{d}(k) = (\underline{x}(k), y(k))$$

where

$$\underline{x}(k) \triangleq (x_1(k), \ldots, x_{SW}(k))$$

with

$$x_i(k) = z_{k-i+1}^{(T_{\text{samp}})} \qquad i = 1, \ldots, SW$$

is the vector of the input variables of the model and $y(k) = z_{k+1}^{(T_{\text{samp}})}$ is the true temperature value which has to be estimated by the output of the NN based on $\underline{x}(k)$. In Figure 4.10, the generic entry $\underline{d}(k)$ of $\mathscr{D}_{SW}^{(T_{\text{samp}})}$, for $k > SW$, is shown.

In addition, it is remarked that entries with at least one `NaN` value—related to air temperature values which, for specific instants of time, are not available in the original time series, as previously explained—have been dropped and, thus, not

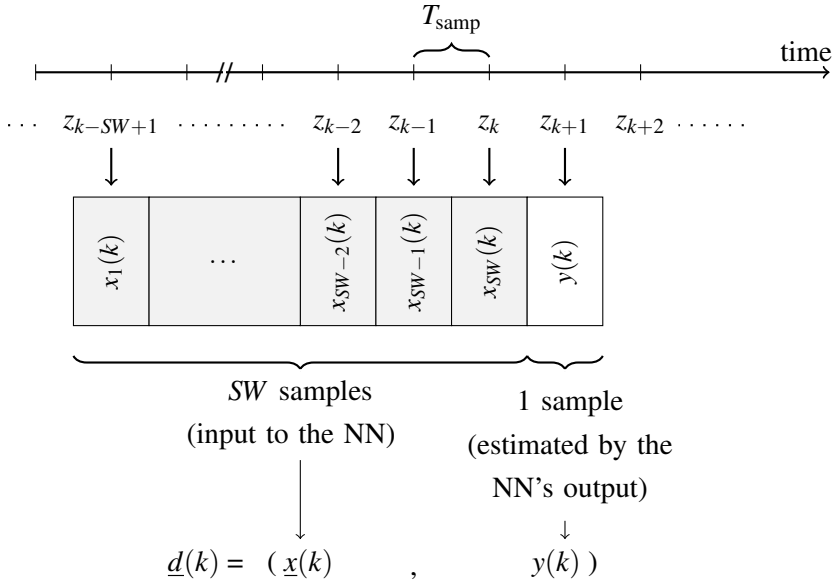


Figure 4.10: The k-th sample $\underline{d}(k)$ in the data set $\mathscr{D}_{SW}^{(T_{\text{samp}})}$ includes SW air temperature values (resulting in the SW-dimensional vector $\underline{x}(k)$ of model's input variables) and an output variable $y(k)$ (related to the air temperature at epoch $k+1$, which has to be predicted starting from $\underline{x}(k)$).

included in the data sets. Moreover, data sets are (randomly) split into a training and a test subset with a ratio equal to 3 : 1. More information related to the obtained data sets, in terms of values of SW and T_{samp} and the number of samples in the set (split between training and test subset), are collected in Table 4.2.

Model Training

As anticipated, three types of NN architectures (namely, LSTMs, RNNs, and ANNs) are evaluated to build the forecasting models with the time series-oriented approach. The resulting models, based on LSTMs, RNNs, and ANNs, are, respectively, shown in Figure 4.11a, Figure 4.11b and Figure 4.11c. As can be noticed, the designed models have a similar structure with respect to the number

Data Set	T_{samp} [min]	SW [samples]	Size [samples]	Training Subset Size [samples]	Test Subset Size [samples]	Data Set	T_{samp} [min]	SW [samples]	Size [samples]	Training Subset Size [samples]	Test Subset Size [samples]
$\mathscr{D}_1^{(10)}$	10	1	36330	27248	9082	$\mathscr{D}_{10}^{(10)}$	10	10	32696	24522	8174
$\mathscr{D}_2^{(10)}$	10	2	35828	26871	8957	$\mathscr{D}_3^{(10)}$	10	3	35363	26523	8840
$\mathscr{D}_4^{(10)}$	10	4	34923	26193	8730	$\mathscr{D}_5^{(10)}$	10	5	34512	25884	8628
$\mathscr{D}_6^{(10)}$	10	6	34118	25589	8529	$\mathscr{D}_7^{(10)}$	10	7	33734	25301	8433
$\mathscr{D}_8^{(10)}$	10	8	33366	25025	8341	$\mathscr{D}_9^{(10)}$	10	9	33020	24765	8255
$\mathscr{D}_1^{(120)}$	120	1	2985	2239	746	$\mathscr{D}_{10}^{(120)}$	120	10	2457	1843	614
$\mathscr{D}_2^{(120)}$	120	2	2912	2184	728	$\mathscr{D}_3^{(120)}$	120	3	2843	2133	710
$\mathscr{D}_4^{(120)}$	120	4	2781	2086	695	$\mathscr{D}_5^{(120)}$	120	5	2723	2043	680
$\mathscr{D}_6^{(120)}$	120	6	2666	2000	666	$\mathscr{D}_7^{(120)}$	120	7	2611	1959	652
$\mathscr{D}_8^{(120)}$	120	8	2558	1919	639	$\mathscr{D}_9^{(120)}$	120	9	2507	1881	626
$\mathscr{D}_1^{(20)}$	20	1	18102	13577	4525	$\mathscr{D}_{10}^{(20)}$	20	10	16006	12005	4001
$\mathscr{D}_2^{(20)}$	20	2	17812	13359	4453	$\mathscr{D}_3^{(20)}$	20	3	17539	13155	4384
$\mathscr{D}_4^{(20)}$	20	4	17286	12965	4321	$\mathscr{D}_5^{(20)}$	20	5	17050	12788	4262
$\mathscr{D}_6^{(20)}$	20	6	16822	12617	4205	$\mathscr{D}_7^{(20)}$	20	7	16605	12454	4151
$\mathscr{D}_8^{(20)}$	20	8	16399	12300	4099	$\mathscr{D}_9^{(20)}$	20	9	16196	12147	4049
$\mathscr{D}_1^{(30)}$	30	1	12067	9051	3016	$\mathscr{D}_{10}^{(30)}$	30	10	10618	7964	2654
$\mathscr{D}_2^{(30)}$	30	2	11862	8897	2965	$\mathscr{D}_3^{(30)}$	30	3	11678	8759	2919
$\mathscr{D}_4^{(30)}$	30	4	11504	8628	2876	$\mathscr{D}_5^{(30)}$	30	5	11341	8506	2835
$\mathscr{D}_6^{(30)}$	30	6	11184	8388	2796	$\mathscr{D}_7^{(30)}$	30	7	11030	8273	2757
$\mathscr{D}_8^{(30)}$	30	8	10885	8164	2721	$\mathscr{D}_9^{(30)}$	30	9	10749	8062	2687
$\mathscr{D}_1^{(40)}$	40	1	9008	6756	2252	$\mathscr{D}_{10}^{(40)}$	40	10	7690	5768	1922
$\mathscr{D}_2^{(40)}$	40	2	8829	6622	2207	$\mathscr{D}_3^{(40)}$	40	3	8656	6492	2164
$\mathscr{D}_4^{(40)}$	40	4	8495	6372	2123	$\mathscr{D}_5^{(40)}$	40	5	8341	6256	2085
$\mathscr{D}_6^{(40)}$	40	6	8196	6147	2049	$\mathscr{D}_7^{(40)}$	40	7	8062	6047	2015
$\mathscr{D}_8^{(40)}$	40	8	7931	5949	1982	$\mathscr{D}_9^{(40)}$	40	9	7806	5855	1951
$\mathscr{D}_1^{(50)}$	50	1	7220	5415	1805	$\mathscr{D}_{10}^{(50)}$	50	10	6185	4639	1546
$\mathscr{D}_2^{(50)}$	50	2	7079	5310	1769	$\mathscr{D}_3^{(50)}$	50	3	6944	5208	1736
$\mathscr{D}_4^{(50)}$	50	4	6816	5112	1704	$\mathscr{D}_5^{(50)}$	50	5	6695	5022	1673
$\mathscr{D}_6^{(50)}$	50	6	6584	4938	1646	$\mathscr{D}_7^{(50)}$	50	7	6479	4860	1619
$\mathscr{D}_8^{(50)}$	50	8	6376	4782	1594	$\mathscr{D}_9^{(50)}$	50	9	6280	4710	1570
$\mathscr{D}_1^{(60)}$	60	1	6006	4505	1501	$\mathscr{D}_{10}^{(60)}$	60	10	5146	3860	1286
$\mathscr{D}_2^{(60)}$	60	2	5886	4415	1471	$\mathscr{D}_3^{(60)}$	60	3	5772	4329	1443
$\mathscr{D}_4^{(60)}$	60	4	5668	4251	1417	$\mathscr{D}_5^{(60)}$	60	5	5570	4178	1392
$\mathscr{D}_6^{(60)}$	60	6	5478	4109	1369	$\mathscr{D}_7^{(60)}$	60	7	5389	4042	1347
$\mathscr{D}_8^{(60)}$	60	8	5306	3980	1326	$\mathscr{D}_9^{(60)}$	60	9	5224	3918	1306

Table 4.2: The created data sets in terms of number of samples, SW, and T_{samp}.

of layers and neurons per layer, except for the first hidden layer. Indeed, the first hidden layer is not feed-forward for the RNN-based and the LSTM-based models but

composed of RNN or LSTM cells, respectively; conversely, it is feed-forward for the ANN-based model. Furthermore, in line with the selected values of *SW*, the number of neurons belonging to the input layer may vary from 1 to 10.

Each of the three models is trained with the 70 data sets created in the previous phases (more exactly, on the training subsets derived from these data sets), adopting the BP algorithm [79] and selecting the RMSE as loss function. From a technical perspective, Python v3.8.6 and the Keras framework v2.4.3 [140] have been employed. Furthermore, the values of learning parameters of the model have been selected as follows: *learning rate* = 0.002, *batch size* = 20, and *number of epochs* = 20. The values of the learning parameters have not been changed for the three models (in other words, have not been individually optimized for each model) in order to fairly evaluate in comparative terms the obtained NN models. As a last comment, the number of hidden layers and the number of their neurons are kept fixed likewise in all NN models: they are set to 3 and to 32/8/3, respectively.

Pure ML Model Re-training

With the goal of performing a fair comparison among the models proposed in this Sub-section (namely, developed with the time series-oriented approach) and the model presented in Sub-section 4.3.4 (namely, developed with the pure ML approach and, for shortness, denoted as $\text{ANN}_{10}^{\text{ML}}$), the latter has been re-trained with additional data (and denoted as $\text{ANN}_{16}^{\text{ML}}$). In detail, these data include weather data and sensor data (referred to air temperatures) collected from the beginning of August 2019 and the end of November 2020. As a comment, the addition of the aforementioned data, gathered during 6 more months with respect to the time period considered in Sub-section 4.3.4, increase the number of samples of the data set of the model developed with the pure ML approach from 5346 to 7919.

Experimental Results

Sliding Window and Sampling Interval. With purpose of evaluating how the values of *SW* and T_{samp} influence the prediction performance of the three proposed

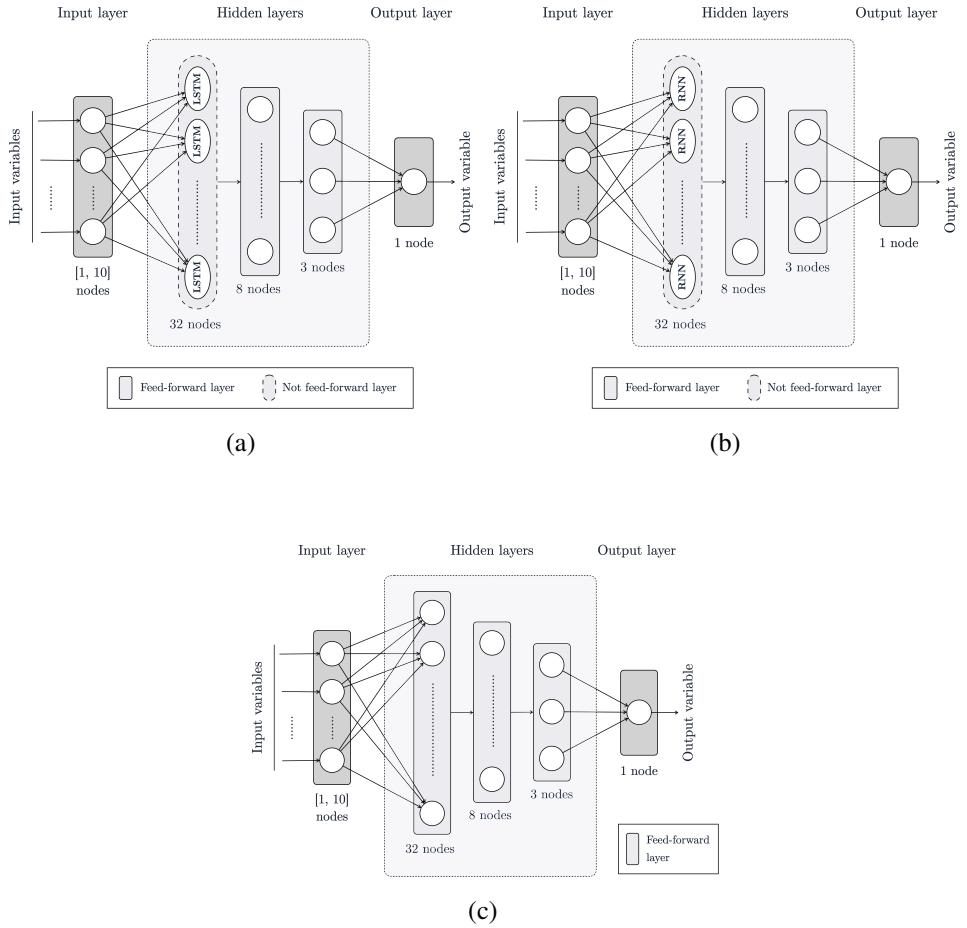


Figure 4.11: NN models evaluated with the time series-oriented approach: (a) LSTM-based; (b) RNN-based; (c) ANN-based.

models (and displayed in Figure 4.11), for each of the 210 models generated in the training phase (namely, 3 models for each of the 70 created data sets), the RMSE, the MAPE, and R^2 have been computed. Figure 4.12 shows the values of the three selected metrics, calculated for the different models, with a three-dimensional plot chart. As a remark, the considered values of RMSE, MAPE, and R^2 displayed in Figure 4.12, together with the others presented in this chapter, are computed

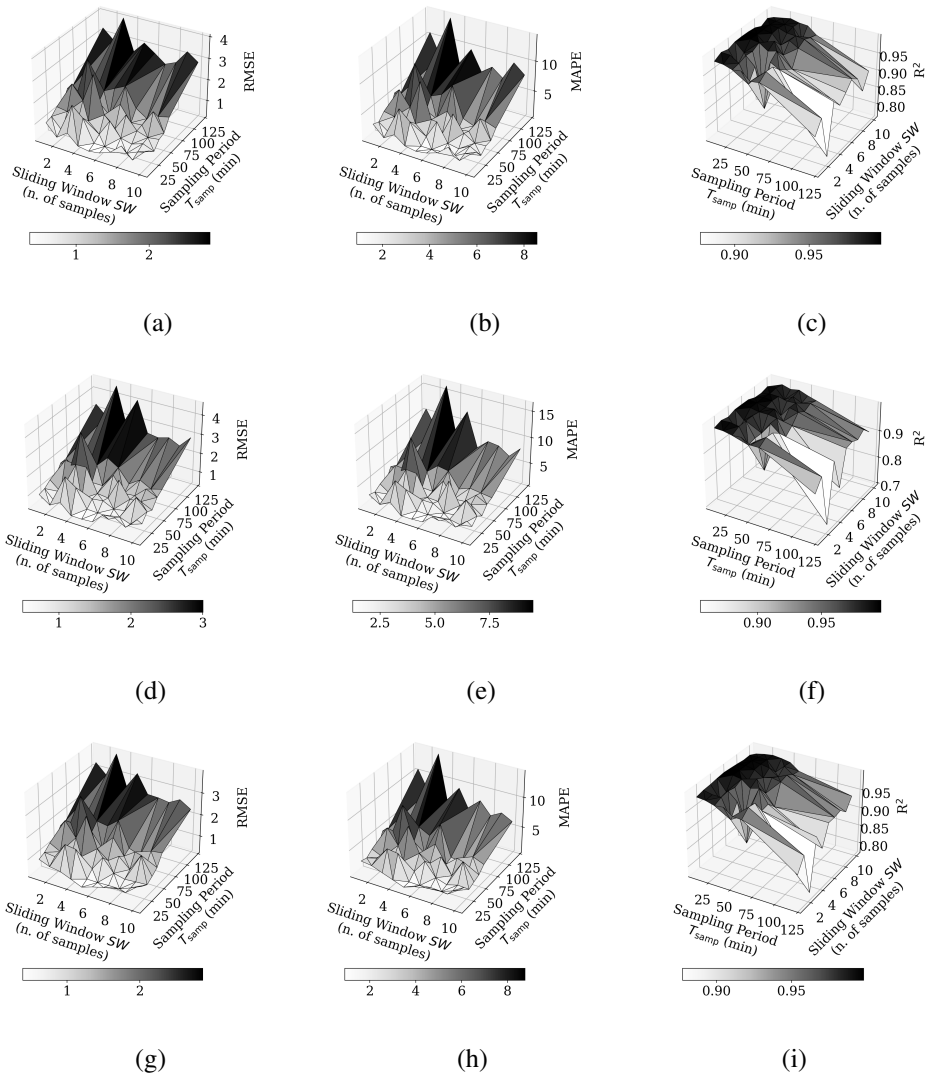
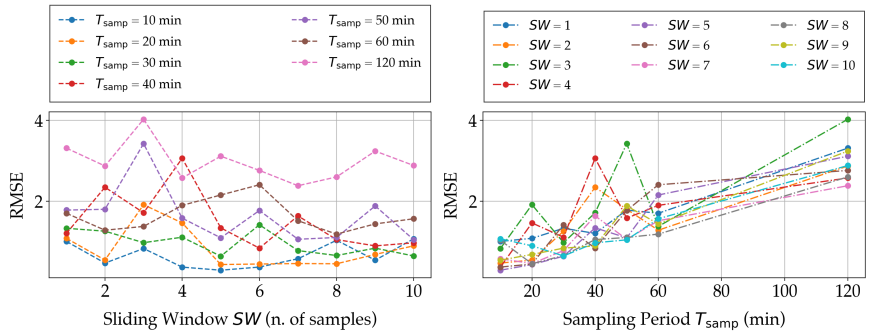
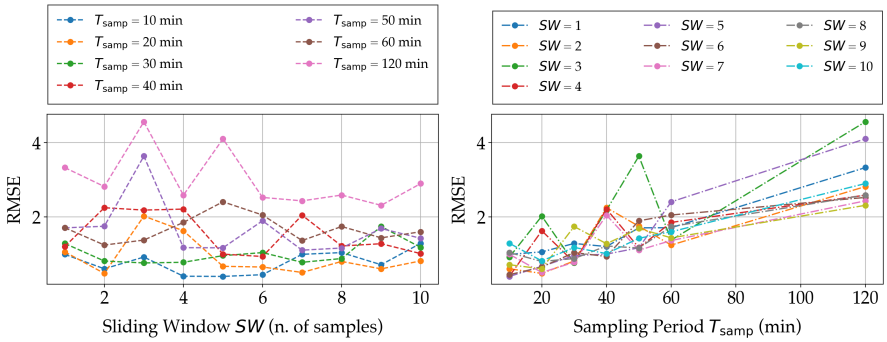


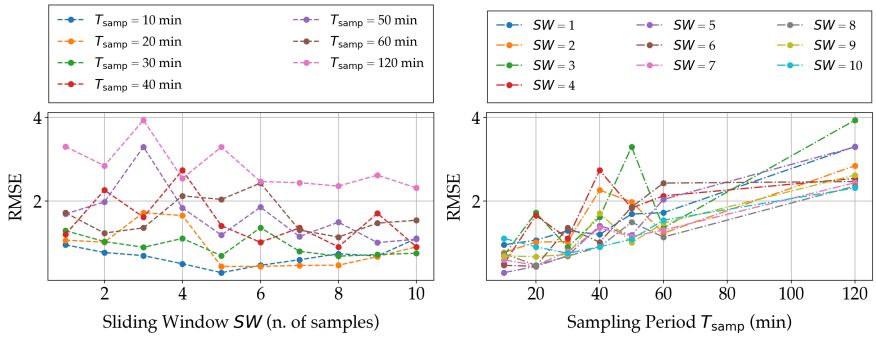
Figure 4.12: Experimental results with the three proposed NN-based models, namely, LSTM (a, b, c), ANN (d, e, f), and RNN (g, h, i), for different values of SW and T_{samp}, in terms of RMSE (a, d, g), MAPE (b, e, h) and R^2 (c, f, i).



(a) LSTM-based models.



(b) ANN-based models.



(c) RNN-based models.

Figure 4.13: Experimental results, in terms of RMSE, on the three proposed NN-based models, namely, (a) LSTM, (b) ANN, and (c) RNN, for different values of SW and T_{samp}.

evaluating the various models on the test sets (namely, set of samples which have not been exploited while training the models). Consequently, the associated performance analysis directly refer to the test sets.

Analyzing Figure 4.12, the following considerations, targeting the trends of the considered evaluation metrics, are valid for all the presented NNs models. *First*, despite the value of *SW*, low values of T_{samp} are linked to low values of RMSE (i.e., a RMSE < 1 °C for $T_{\text{samp}} = 10$ or 20 or 30 min, as shown in Figure 4.12a, Figure 4.12d, and Figure 4.12g). Conversely, high values of RMSE and, thus, lower prediction performance, are achieved with higher values of T_{samp} (for instance, with $T_{\text{samp}} = 120$ min). *Second*, the same trend is shown in all cases by the MAPE metric, as shown in Figure 4.12b, Figure 4.12e, and Figure 4.12h. *Lastly*, the higher the R^2 is, the smaller T_{samp} is (namely, $T_{\text{samp}} = 10$ min). Moreover, R^2 decreases for rising values of T_{samp}. Furthermore, an increase in the value of *SW* leads to an increment in R^2, as can be seen by Figure 4.12c, Figure 4.12f, and Figure 4.12i.

The bi-dimensional charts displayed in Figure 4.13, in which the performance of the three NN-based models, in terms of RMSE, are arranged as functions of both *SW* (with T_{samp} as a parameter) or T_{samp} (with *SW* as a parameter), confirm the first trend. As a remark, the plots displayed in Figure 4.13 are obtained by projecting the three-dimensional plots in Figure 4.12 onto the two vertical planes.

In addition, the minimum, maximum, and average values of the three selected evaluation metrics, over the 210 developed models, are collected in Table 4.3, together with the relative design parameters (namely, *SW* and T_{samp}) referring to minimum and maximum values. According to Figure 4.12 and Table 4.3, among the considered models some reach satisfactory performance on test sets while, alternatively, some others are less accurate. For instance, an example of model having a low accuracy is the ANN-based model trained with (T_{samp}, *SW*) = (120, 3), which has RMSE = 4.561°C, MAPE = 16.35%, and R^2 = 0.699 on the test set.

Although their experimental results are reported in Figure 4.12, low-performing models, in details, models with RMSE > 1°C, MAPE > 3%, and $R^2 < 0.980$ on at least one test set, are not included in the following analysis. As a clarification, less-performing models have been, nevertheless, considered in the previous paragraphs

NN arch. type		RMSE [°C]			MAPE [%]			R^2		
		Value	T_{samp}	SW	Value	T_{samp}	SW	Value	T_{samp}	SW
ANN	Min	0.402	10	5	1.03	10	4	0.699	120	3
	Max	4.561	120	3	16.35	120	3	0.998	10	4, 5
	Avg	1.52	N/A	N/A	4.29	N/A	N/A	0.96	N/A	N/A
RNN	Min	0.290	10	5	0.87	10	5	0.776	120	3
	Max	3.933	120	3	14.14	120	3	0.999	10	5
	Avg	1.45	N/A	N/A	4.10	N/A	N/A	0.96	N/A	N/A
LSTM	Min	0.294	10	5	0.89	10	5	0.766	120	3
	Max	4.024	120	3	14.08	120	3	0.999	10	5
	Avg	1.46	N/A	N/A	4.17	N/A	N/A	0.96	N/A	N/A

Table 4.3: Minimum (min), maximum (max), and average (avg) values of RMSE, MAPE, and R^2 obtained over the 210 trained models.

and included in Figure 4.12 since the author of this doctoral thesis wants to provide an exhaustive discussion concerning the influence of the value of the sampling interval and the dimension of the sliding window on the model prediction performance.

Type of NN Architecture. The performance of the selected architectures (i.e., LSTM, RNN, and ANN) are evaluated in comparative terms over a group of selected sets over the 70 data sets generated in the previous phases. In detail, 17 models, trained on the same 17 data sets, have been considered for each type of NN architecture. More exactly, all the data sets $\mathscr{D}_{SW}^{(T_{\text{samp}})}$ in correspondence to which the LSTM, RNN, and ANN models trained with $T_{\text{samp}} \in \{10, 20, 30, \ldots, 120\}$ and $SW \in \{1, 2, \ldots, 10\}$ have RMSE ≤ 1 °C, MAPE $\leq 3\%$, and $R^2 \geq 0.980$ on test sets, have been chosen. The prediction performance related to the selected models are collected in Table 4.4.

With respect to the experimental results achieved by the three NN-based models for the chosen values of SW and T_{samp}, they are shown, in terms of RMSE, MAPE,

Data Set	T_{samp}	SW	RMSE [°C]			MAPE [%]			R^2		
			LSTM	RNN	ANN	LSTM	RNN	ANN	LSTM	RNN	ANN
$\mathscr{D}_2^{(10)}$	10	2	0.470	0.769	0.608	1.42	2.80	1.91	0.997	0.992	0.995
$\mathscr{D}_3^{(10)}$	10	3	0.830	0.696	0.923	2.51	2.14	2.98	0.991	0.994	0.989
$\mathscr{D}_4^{(10)}$	10	4	0.370	0.501	0.407	1.19	1.52	1.03	0.998	0.997	0.998
$\mathscr{D}_5^{(10)}$	10	5	0.294	0.289	0.402	0.89	0.87	1.04	0.999	0.999	0.998
$\mathscr{D}_6^{(10)}$	10	6	0.371	0.464	0.449	1.16	1.53	1.18	0.998	0.997	0.997
$\mathscr{D}_7^{(10)}$	10	7	0.577	0.598	0.997	1.78	1.94	2.95	0.996	0.996	0.987
$\mathscr{D}_9^{(10)}$	10	9	0.542	0.685	0.717	1.69	2.08	2.23	0.996	0.994	0.993
$\mathscr{D}_5^{(20)}$	20	5	0.434	0.438	0.674	0.90	0.93	1.68	0.998	0.998	0.994
$\mathscr{D}_6^{(20)}$	20	6	0.447	0.439	0.655	0.93	0.91	1.63	0.997	0.998	0.995
$\mathscr{D}_7^{(20)}$	20	7	0.458	0.461	0.509	1.13	1.16	1.17	0.997	0.997	0.997
$\mathscr{D}_8^{(20)}$	20	8	0.453	0.466	0.805	0.98	1.07	2.06	0.997	0.997	0.992
$\mathscr{D}_9^{(20)}$	20	9	0.684	0.674	0.606	2.00	1.77	1.64	0.994	0.994	0.995
$\mathscr{D}_{10}^{(20)}$	20	10	0.897	0.907	0.820	2.80	2.94	2.08	0.990	0.990	0.992
$\mathscr{D}_3^{(30)}$	30	3	0.974	0.894	0.765	2.97	2.66	1.61	0.987	0.989	0.992
$\mathscr{D}_5^{(30)}$	30	5	0.640	0.693	0.961	1.72	1.89	2.90	0.995	0.994	0.988
$\mathscr{D}_7^{(30)}$	30	7	0.778	0.799	0.782	1.73	1.97	1.68	0.993	0.992	0.992
$\mathscr{D}_8^{(30)}$	30	8	0.657	0.682	0.883	1.39	1.38	1.99	0.995	0.994	0.990

Table 4.4: Prediction performances of the three proposed models on a reduced set of selected values of SW and T_{samp} (those which are better performing).

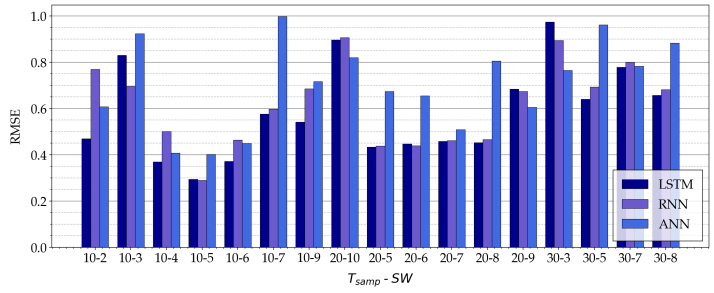
and R^2, in Figure 4.14. As displayed in Figure 4.14a and Figure 4.14b, for fixed values of SW and T_{samp}, there not exists a NN architecture which is better-performing, in terms of RMSE and MAPE, than the others in all cases. For instance, the LSTM model has the best performance in terms of RMSE with $T_{\text{samp}} = 10$ and $SW = 4$, conversely, the RNN model is the most accurate for $T_{\text{samp}} = 10$ and $SW = 5$. Generally, better results, in terms of selected evaluation metrics, are achieved with the

following couples of values of (T_{samp}, SW): $(10,4)$, $(10,5)$, $(10,6)$, $(20,5)$, $(20,6)$, $(20,7)$, and $(20,8)$. Indeed, RNN and LSTM models have RMSE $\leq 0.5\,^{\circ}$C, MAPE $< 1.6\%$, and $R^2 \geq 0.997$, for the above listed values.

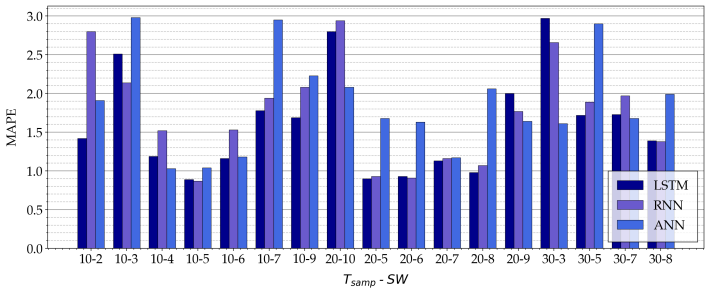
As a final comment, the best performance, according to the considered metrics, are reached by the RNN model trained with (T_{samp}, SW) $= (10,5)$. In detail, this model ensures a RMSE = $0.289\,^{\circ}$C, a MAPE = 0.87%, and a R^2 = 0.999. It is remarked that very similar results have been reached with the LSTM model with the identical values of SW and T_{samp} (i.e., RMSE $= 0.294\,^{\circ}$C, MAPE $= 0.98\%$, and $R^2 = 0.999$).

Performance Analysis and Literature Comparison. In Table 4.5 are shown the experimental results related to (i) the model presented in Sub-section 4.3.4 and obtained with a data set of data gathered during a time period of 10 months (namely, $\text{ANN}_{10}^{\text{ML}}$); (i) the results of the same model re-trained with a larger data set including samples collected during 6 additional months (namely, $\text{ANN}_{16}^{\text{ML}}$); and (iii) the three models (ANN, RNN, and LSTM) built with the data set $\mathscr{D}_5^{(10)}$. According to results shown in Table 4.5, the re-trained model $\text{ANN}_{16}^{\text{ML}}$ has degraded prediction performance with respect to the $\text{ANN}_{10}^{\text{ML}}$ model. This can be justified by the fact that, despite the model re-training involves supplementary samples (and, for this reason, should be, ideally, more accurate), it may happen that the additional data are more unbalanced (for instance, in terms of number of gathered samples per month, as shown in Figure 4.7), thus reducing the final accuracy. Furthermore, if the $\text{ANN}_{10}^{\text{ML}}$ model is compared, in terms of prediction performance, with the three models with lowest RMSE obtained with the data set $\mathscr{D}_5^{(10)}$ (namely: $\text{LSTM}_5^{(10)}$, $\text{RNN}_5^{(10)}$, and $\text{ANN}_5^{(10)}$), one can conclude that the RMSEs of the latter are slightly lower. Indeed, the $\text{ANN}_{10}^{\text{ML}}$ model has, with respect to the last three shown in Table 4.5, a RMSE higher by at least $1\,^{\circ}$C, a MAPE higher by approximately 3%, and a R^2 lower by approximately 0.24.

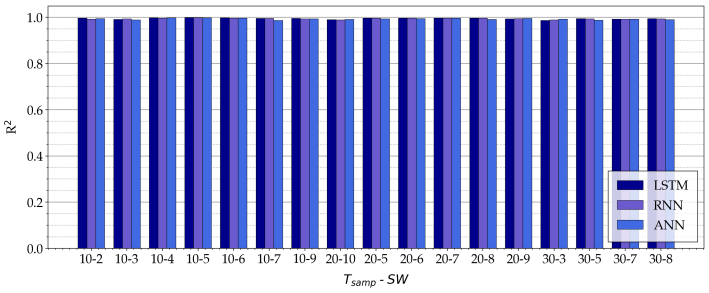
In Table 4.5, the architectural and computational complexities of the selected models are evaluated. More precisely, the computational and architectural complexities can be described, respectively, in terms of MAC operations and number



(a)



(b)



(c)

Figure 4.14: Experimental results on the three proposed NN-based models, namely, LSTM, ANN, and RNN, for a relevant reduce number of pairs of (T_{samp}, SW), expressed in terms of (a) RMSE, (b) MAPE, and (c) R^2.

Model	RMSE [°C]	MAPE [%]	R^2	Accuracy [%]	MAC Operations	Parameters Number	NetScore
ANN_{10}^{ML}	1.50	4.91	0.965	48.87	1018	1018	17.05
ANN_{16}^{ML}	2.28	6.54	0.931	34.91	1018	1018	3.60
$LSTM_5^{(10)}$	0.294	0.89	0.999	99.18	22192	4625	−0.59
$RNN_5^{(10)}$	0.289	0.87	0.999	99.28	5712	1361	25.25
$ANN_5^{(10)}$	0.402	1.04	0.998	97.28	464	464	60.31

Table 4.5: Prediction performance and complexity of a reduced set of evaluated models, in terms of RMSE, MAPE, R^2, accuracy (namely, number of samples predicted with RMSE lower than 1 °C), MAC operations, number of parameters, and NetScore of the models.

of parameters characterizing the model (see Sub-section 1.4.4). According to the results shown in Table 4.5, the most architecturally and computationally complex model is based on the LSTM model, i.e., $LSTM_5^{(10)}$, with respect to the other NN architecture. The RNN-based model, i.e., $RNN_5^{(10)}$, ranks second and is then followed by the ANN_{10}^{ML} model and its re-trained version (ANN_{16}^{ML}). The "lightest" model is $ANN_5^{(10)}$ model, which has only 464 parameters and 464 MAC operations. In other terms, $ANN_5^{(10)}$ is the model with the lowest architectural and computational complexities with respect to the others.

To conclude, the NetScore metric (see Sub-section 4.3.3) is computed for all the 5 models collected in Table 4.5. This in order to define which model achieves the best trade-off between prediction performance and complexity. According to Eq. (4.4), given a target NN model $\mathcal{N}$, one of the parameters needed to compute the NetScore is the model's accuracy $a(\mathcal{N})$. Accuracy is an evaluation metric usually used in classification tasks, which, considering a target data set, can be expressed as the percentage of samples correctly attributed to their classes by a model. Since the air temperature forecasting is, in practise, a regression task and, due to this, the typical definition of the accuracy metrics cannot be adopted for the evaluation of the models, the concept of accuracy is redefined—and, thus, the meaning of $a(\mathcal{N})$ in Eq. (4.4)—with the purpose of computing the NetScore metric for the models. More

precisely, the following two elements are defined: (i) a threshold parameter $\mathscr{T}$; and (ii) an indicator function $\mathscr{I}$. For each sample $\underline{d}(k) = (\underline{x}(k), y(k))$ in the test subset of the model to be evaluated, $\mathscr{I}$ may assume a binary value as follows:

$$\mathscr{I}(\underline{d}(k)) \;=\; \mathscr{I}(y(k),\hat{y}(k)) = \mathscr{U}(\mathscr{T} - |y(k) - \hat{y}(k)|) \quad = \quad \begin{cases} 1 & \text{if } |y(k) - \hat{y}(k)| < \mathscr{T} \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

where: $\mathscr{U}(\cdot)$ is the unit step function; $|\cdot|$ is the modulo operator; $\underline{d}(k)$ is the k-th entry in the test subset; $\underline{x}(k)$ is the vector of the input samples of the test subset entry $\underline{d}(k)$; $y(k)$ and $\hat{y}(k)$ are the actual (in the entry $\underline{d}(k)$) and the forecast by the NN (with input $\underline{x}(k)$) values of air temperature for the k-th test subset entry, respectively; and $\mathscr{T}$ is a threshold value.

Therefore, the accuracy $a(\mathscr{N})$ for the NN-based model $\mathscr{N}$ over all the samples included in the test subset is defined as follows:

$$a(\mathscr{N}) \quad = \quad \frac{100}{N_{\text{tst}}} \sum_{r=1}^{N_{\text{tst}}} \mathscr{I}(y(k),\hat{y}(k)) \tag{4.9}$$

where: N_{tst} is the number of samples in the test subset; $y(k)$ and $\hat{y}(k)$ are, respectively, the real and the forecast values (of air temperature) for the k-th test subset entry.

The value of $\mathscr{T}$ is set to 1 °C. This means that, conceptually, a future value of air temperature $\hat{y}(k)$ forecast by the model is treated as *correct* (e.g., $\mathscr{I}(y(k),\hat{y}(k)) = 1$) if the absolute value of the difference between real and forecast air temperature values ($|y(k) - \hat{y}(k)|$) is smaller than 1 °C. The exponential coefficients of the NetScore metric are set to their default values, i.e., $\alpha = 2$ and $\beta = \gamma = 0.5$. The computed NetScore values are listed in Table 4.5.

As shown in Table 4.5, the $\text{ANN}_5^{(10)}$ model reaches the highest NetScore, followed by the $\text{RNN}_5^{(10)}$ model, and, then, by the $\text{ANN}_{10}^{\text{ML}}$ model. Conversely, the re-trained version of this last model ($\text{ANN}_{16}^{\text{ML}}$) and the $\text{LSTM}_5^{(10)}$ model gain the lowest NetScore. This means that these models reach a rather unbalanced trade-off between their prediction performance and complexity. In fact, the performance of the $\text{ANN}_{16}^{\text{ML}}$ model is lower than that of the $\text{LSTM}_5^{(10)}$ model (e.g., with RMSE equal to

2.28°C and 0.294°C, respectively). In contrast, the ANN_{16}^{ML} model has computational and architectural complexities considerably lower than those of the $LSTM_5^{(10)}$ model. Indeed, the ratio between the number of MAC operations of the first model and $LSTM_5^{(10)}$ is approximately equal to $1:22$. Globally, the preferred model to be executed on edge devices is $ANN_5^{(10)}$.

Moreover, the three best models evaluated in the previous paragraphs, in details, $LSTM_5^{(10)}$, $RNN_5^{(10)}$, and $RNN_5^{(10)}$, have prediction performance comparable with those of literature works listed in Table 1.9. More precisely:

- the proposed NN-based models have a RMSE in the range $0.289 \div 0.402$°C, which is lower than that in [84, 85, 88, 89] and is moderately higher than that in [86, 87];
- the selected NN-based models have a MAPE in the range $0.87 \div 1.04\%$, thus smaller than that in [86, 89];
- the value of R^2 of the selected NN-based models is higher than those of all papers collected in Table 1.9.

As a final remark, the literature papers listed in Table 1.9 do not provide information concerning the complexity of the developed models. Due to this, a direct comparison between literature works and the proposed NN-based models, from a complexity-performance trade-off perspective, cannot be performed.

Possible Application Scenario and Reference Architecture

Practically, the NN model selected in the previous paragraph, in other words, $ANN_5^{(10)}$, can be effectively deployed on a real edge device, denoted as Smart GW. As detailed for the algorithm developed with the pure ML approach and presented in Sub-section 4.3.4 (i.e., ANN_{10}^{ML}), the Smart GW can be built around a RPi [48], which is a valid option in the deployment of AI algorithms at the edge [138, 139].

As said, the Smart GW can be placed inside a greenhouse, which further hosts a WSN aiming at monitoring the inner air temperature, and acts as a sink node for the SNs of the WSN, as displayed in Figure 4.15. In detail, sensed air temperature

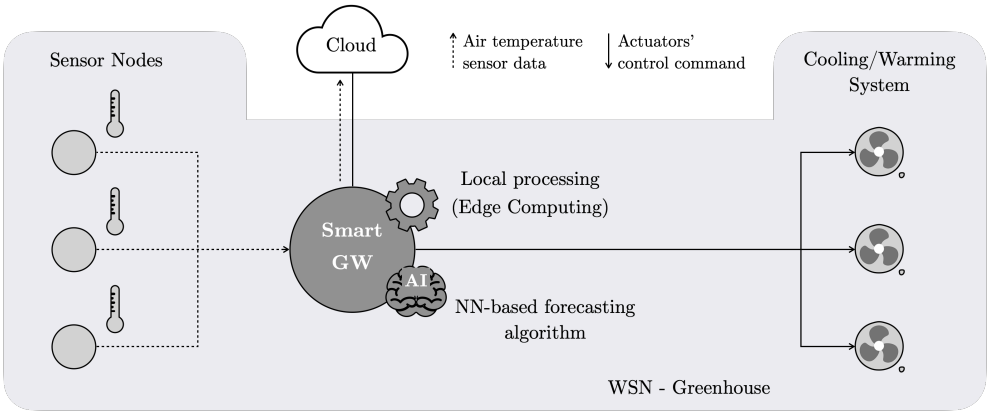


Figure 4.15: Possible reference architecture and application scenario for the developed forecasting model based on the time series-oriented approach.

values are forwarded by SNs installed in the greenhouse to the Smart GW through an available wireless connectivity in the greenhouse, such as, a WiFi network, which may have to be specifically introduced if not already present. Thereafter, the GW temporarily stores, within its internal memory, the received sensor data and, if required and if an Internet AP is available close to the greenhouse, forwards them to the Cloud.

In the case of some sort of cooling/warming system is installed inside the greenhouse, the Smart GW can be designed to control it in order to regulate the values of the internal air temperature and humidity. (This approach can be exploited, for example, in a demonstration greenhouse in the AFarCloud project [141].) If required and when a proper number of successive air temperature values are gathered from SNs—which are supplied as inputs to the model (e.g., 5 air temperature values for $\mathrm{ANN}_5^{(10)}$)—the Smart GW runs the forecasting algorithm, which, in turn, provides a predicted future value of the internal air temperature. Based on the forecast value, the Smart GW determines if the cooling/warming system has to be turned on, disabled, or left in its current operational state. For instance, if the value of the forecast air temperature exceed a fixed threshold which cannot be overcome within the greenhouse, the GW can communicate to the inner cooling system to activate fans

to avoid the internal air temperature reaching the unwanted (and unsafe) predicted value.

To conclude, as already said, the introduction of EdgeAI applications has an extremely positive impact on greenhouses' management. In fact, predicting the trend of inner environmental variables, as air temperature, allows to preemptively schedule some actions. This means that the greenhouse's actuators can be more efficiently controlled to maintain (within the greenhouse) the best growing conditions for the internal cultivation. Moreover, running AI algorithms and performing decision-making at the edge, instead of in the Cloud, increase the robustness (against connectivity problems) of a greenhouse's management and reduces latency.

4.3.6 Conclusions

In this section, two possible NN-based approaches, expedient to build a forecasting algorithm which can be embedded into a Smart GW, acting as sink node and boarder router for SNs monitoring the internal air temperature of a greenhouse, have been presented. In detail, in order to build the above-mentioned constrained device-friendly forecasting algorithm, the pure ML approach and the time series-oriented approach have been evaluated and discussed, respectively, in Sub-section 4.3.4 and Sub-section 4.3.5.

In the *first* case, the external meteorological conditions of the greenhouse (namely, apparent temperature, dew point, air humidity and temperature, and UV index) and a time reference (namely, hour of the day and harvest month), have been selected as input variables for an ANN-based forecasting model. In detail, meteorological data can be gathered from locally deployed sensors (namely, weather stations) or from the Cloud, namely, from the DarkSky repository (Sub-section 4.3.4). Moreover, experimental results, performed on a test set, shown that the model can forecast a greenhouse internal air temperature with a RMSE of 1.50 °C, a MAPE of 4.91%, and a R^2 of 0.965. Furthermore, the ANN-based developed prediction model is based on only 4 hidden layers and 1018 parameters and has been especially designed to be lightweight and executable by a constrained IoT device (i.e., a Smart GW built on a RPi3).

In the *second* evaluated approach (i.e., time series-oriented), instead of weather conditions external to the greenhouse, indoor air temperature values, collected inside the greenhouse, have been considered as input sources for the developed model (Sub-section 4.3.5). Moreover, three types of NN architectures (i.e., LSTM, RNN, and ANN) have been evaluated to discover the one with the best performance and the lowest complexity, from both computational and architectural perspective. This aspect is crucial while running NN models on edge devices having constrained capabilities (i.e., a Smart GW).

The achieved experimental results show that the three proposed models, based on LSTM, RNN, and ANN, have prediction performance comparable to that of literature works and improved than the model developed with the pure ML approach. In detail, the three best performing models, achieved adopting $SW = 5$ input variables and with a sampling interval $T_{\text{samp}} = 10$ min, have a RMSE in the range of $0.289 \div 0.402$°C, a MAPE in the range of $0.87 \div 1.04\%$, and a $R^2 \geq 0.997$. Furthermore, according to obtained results, the lower computational complexity, expressed as the number of MAC operations and parameters, is reached by the ANN-based model. More exactly, the best performance-complexity trade-off is guaranteed by $\text{ANN}_5^{(10)}$.

In general, the design and deployment of NN-based accurate forecasting algorithms, having a computational complexity compatible with the computing resources of edge devices, is an attractive and promising research topic, not comprehensively explored in recent literature. This is still more true for the tasks of forecasting of a greenhouse's inner variables, which this chapter has tried to address in the more comprehensive possible way.

Finally, some examples of possible future research directions for the presented application are listed in the following. *First*, the performance, based on relevant metrics for the online execution of an algorithm, of the proposed NN-based algorithms can be evaluated on diverse types of edge devices. Some interesting performance metrics in this context include: (i) the time required by the device to execute the algorithm in a real use case; (ii) the (flash) memory space of the device needed to store the model, along with the RAM required to run it; and (iii) the power consumed by the edge device while it runs the NN-based algorithm to predict a

future value of air temperature. *Second*, the same methodologies followed to build an air temperature forecasting model can be used to develop algorithms aiming at forecasting other types of internal variables, rather than air temperature, which are relevant for greenhouse cultivation and, thus, have to be monitored and regulated, as air humidity. *Finally*, the proposed approaches, based on the Smart GW, can be effectively integrated in the LoRaFarM architecture (presented in Section 3.2) as an additional module.

Conclusions

This doctoral thesis aims at supporting the development of Smart Agriculture (SA) and future smart farms. This goal is achieved by proposing a reference framework to build general-purpose Internet of Things (IoT)/SA-oriented architectures, which are modular, scalable, and can efficiently handle typical challenges of these systems. In detail, these challenges include: security and privacy, connectivity issues, heterogeneity management, interoperability, integration, agricultural harsh environments, standardization, developing easy-to-use and plug-and-play solutions, and physical device's proper power source selection and management.

The proposed design framework has been conceptualized and validated through the following methodological steps. *First*, in order to better understand the background of the proposed framework (namely, a framework for the design of IoT systems targeting SA applications), nowadays agriculture's limitations and future trends (impacting on it and leading to the digitalization of farms' production processes, e.g., global population growth, climate changing) have been investigated. On the basis of this analysis, the main technologies to adopt and the main challenges to handle in making the agricultural sector smarter and greener have been highlighted. The identified technologies include (just to name a few examples) autonomous vehicles, robots, sensors and actuators, Artificial Intelligence (AI) tools, and IoT: the last one plays a crucial role. Indeed, IoT technologies allows to build IoT/SA-oriented systems able to collect, exchange, process, and store (i.e., manage) relevant data for future smart farms, integrating a wide and heterogeneous set of other technologies

within the same Farm Management System (FMS). For this reason, literature related to key aspects, architectural models, building components, and main challenges to consider while designing and developing general-purpose IoT systems (which can, then, be specialized for SA scenarios) have been surveyed.

To complete the *first* contribution of this thesis, an additional literature review, aiming at discovering possible research gaps of the presented IoT/SA-oriented systems (to be covered by this doctoral thesis), has been performed. The found gaps are the following.

- General-purpose IoT/SA-oriented systems have not being already presented in the literature. Indeed, reviewed works focus on the management of a reduced set of farm's activities (i.e., agricultural scenarios), but future farms will have to manage a multiplicity of them.
- A reference design framework to be followed, aiming at simplifying the development of such type of systems (namely, general-purpose IoT/SA-oriented systems), has not been already proposed.
- The execution of AI algorithms on IoT edge devices with limited computing capabilities (which is a novel research trend—usually denoted as EdgeAI—providing several advantages, e.g., creation of self-managed smart environments), to enhance the management of farm's activities, has not been satisfactorily explored in the context of SA.

Second, a layered reference framework to develop general-purpose IoT/SA-oriented systems has been presented. In detail, the goal has been achieved describing the main building components of the proposed framework, together with a few examples of the more promising technologies to implement them. The proposed framework, in terms of building components, their implementation, and their interactions, is a powerful tool the future developers and researchers can adopt in order to build robust, modular, and smart IoT/SA-oriented systems.

Third, on the basis of the proposed framework, two IoT/SA-oriented systems, namely, VegIoT Garden [12] and LoRaFarM [13], have been implemented. In

particular, LoRaFarM is a general-purpose architecture aiming at supporting the management of future smart farms in a highly customizable way. It has been tested in an evaluation farm (the Podere Campáz [120]) and has been operational for at least 2 years (namely, from June 2019 to September 2021). As this highlights, LoRaFarM is robust and can effectively enhance the management of a farm's activities over a long period of time, it could evolve towards a commercial product in the future. This is an additional contribution of this doctoral thesis (i.e., the development of a prototypical architecture as a first step in the process of building a commercial IoT/SA-oriented system).

Finally, two different methodologies (i.e., pure Machine Learning (ML)-based and time series-oriented) to be used to build Neural Network (NN)-based algorithms aiming at forecasting future values of environmental variables (e.g., air temperature trends within a greenhouse) have been presented. These methodologies can be followed in the future to develop novel forecasting algorithms, in addition to those proposed in this doctoral thesis. Moreover, key aspects to be considered and key approaches to be adopted while developing and deploying such type of algorithms on real (constrained) IoT edge devices, with a focus on SA applications, have been discussed. Since this last aspect is typically poorly discussed in literature, this can be considered an additional contribution of this thesis (in the field of EdgeAI and SA). Indeed, as discussed in Sub-section 1.4.4, the embedding of data analytical techniques within edge devices (and, thus, edge computing) leads to several benefits. They include, just to name few examples, the reduction of latency to perform both AI algorithms-oriented tasks (as data pre-processing, data fusion, and model inference), since they are executed at the edge, and data transmission. Security and privacy, indeed, if data are locally processed by AI algorithms, no (sensible) information (or, eventually, a reduced amount of these) have to be forwarded to the Cloud to be analyzed. Sustainability, since EdgeAI-powered applications can ideally work even when network access is not available due to, for instance, temporary lack of Internet connectivity or networking infrastructure.

Furthermore, since assets and things (as tractors, robots, and micro-controllers) enriched with "intelligence" (i.e., namely AI algorithms) are able to perform smart

operations without requiring a Cloud link, the level of automation of several applications can be increased by EdgeAI. In addition, automate decision making can be made faster thanks to real-time (edge) processing. To conclude, together with data inference, whenever additional information to build a more accurate model are collected, AI algorithms deployed on edge devices can be locally updated (in other terms re-trained) based on the new acquired data. This means that new knowledge coming from the new available data can be automatically exploited to increase the performance of an already deployed model.

Future research directions for the covered topics can be summarized as follows. *First*, the proposed reference model can be validated developing additional IoT/SA-oriented systems (for example, targeting the monitoring of cows' behavior). *Second*, the presented LoRaFarM architecture can be extended with additional modules and made even more modular, introducing, for example, a microservice-based architecture at the middleware layer. *Third*, data collected from LoRaFarM can be used to build additional data forecasting algorithms. As an example, together with air temperature, an algorithm forecasting future values of air humidity within the greenhouse can be deployed in order to automatically regulate in a smart way the internal microclimate. In general, the use of AI and EdgeAI algorithms aiming at increasing the degree of autonomy of many cyber-physical systems (and, thus, IoT/SA-oriented applications) is an interesting research topic to investigate. In fact, this trend will significantly impact on the way IoT solutions are used and how their value for agricultural production is perceived by farmers. *Fourth*, different microcomputers (instead of the used RPi) can be deployed to run the developed air temperature forecasting algorithm on the basis of relevant metrics for its online execution (e.g., time to run it, RAM required to the device), such as STMicroelectronics boards [96].

To conclude, the author wants to remark that the final goal of this manuscript is to provide the reader (and the scientific community) with an effective reference aiming at supporting the design and development of future IoT/SA-oriented architectures, providing a solid theoretical framework enriched with lessons learned from practical experience in the field.

List of Publications

- G. Codeluppi, A. Cilfone, L. Davoli, and G. Ferrari. VegIoT Garden: a modular IoT Management Platform for Urban Vegetable Gardens. In *2019 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 121–126, Portici, Italy, 2019.
- Gaia Codeluppi, Antonio Cilfone, Luca Davoli, and Gianluigi Ferrari. LoRaFarM: A LoRaWAN-Based Smart Farming Modular IoT Architecture. *Sensors*, 20(7), 2020.
- G. Codeluppi, A. Cilfone, L. Davoli, and G. Ferrari. AI at the Edge: a Smart Gateway for Greenhouse Air Temperature Forecasting. In *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 348–353, 2020.
- Gaia Codeluppi, Luca Davoli, and Gianluigi Ferrari. Forecasting Air Temperature on Edge Devices with Embedded AI, *Sensors*, 21(12), 2021.

Bibliography

[1] FAO. The future of food and agriculture: Trends and challenges. *[Online pdf]*, 2017.

[2] De Clercq Matthieu, Vats Anshu, and Biel Alvaro. Agriculture 4.0: The future of farming technology, 2018.

[3] T. Truong, A. Dinh, and K. Wahid. An IoT Environmental Data Collection System for Fungal Detection in Crop Fields. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4, 4 2017.

[4] Sergio Trilles, Joaquin Torres-Sospedra, Ascar Belmonte, F. Javier Zarazaga-Soria, Alberto Gonzalez-Perez, and Joaquin Huerta. Development of an Open Sensorized Platform in a Smart Agriculture Context: A Vineyard Support System for Monitoring Mildew Disease. *Sustainable Computing: Informatics and Systems*, 2019.

[5] S. R. Jino Ramson, Walter D. León-Salas, Zachary Brecheisen, Erika J. Foster, Cliff T. Johnston, Darrell G. Schulze, Timothy Filley, Rahim Rahimi, Martín Juan Carlos Villalta Soto, Juan A. Lopa Bolivar, and Mauricio Postigo Málaga. A self-powered, real-time, lorawan iot-based soil health monitoring system. *IEEE Internet of Things Journal*, 8(11):9278–9293, 2021.

[6] Mustafa Alper Akka and Radosveta Sokullu. An IoT-based Greenhouse

Monitoring System with Micaz Motes. *Procedia Computer Science*, 113:603–608, 2017.

[7] Min-Sheng Liao, Shih-Fang Chen, Cheng-Ying Chou, Hsun-Yi Chen, Shih-Hao Yeh, Yu-Chi Chang, and Joe-Air Jiang. On Precisely Relating the Growth of Phalaenopsis Leaves to Greenhouse Environmental Factors by Using an IoT-based Monitoring System. *Computers and Electronics in Agriculture*, 136:125–139, 2017.

[8] Lean Karlo S. Tolentino, Edmon O. Fernandez, Romeo L. Jorda, Shayne Nathalie D. Amora, Daniel Kristopher T. Bartolata, Joshua Ricart V. Sarucam, June Carlo L. Sobrepeña, and Kristine Yvonne P. Sombol. Development of an iot-based aquaponics monitoring and correction system with temperature-controlled greenhouse. In *2019 International SoC Design Conference (ISOCC)*, pages 261–262, 2019.

[9] A. Ilapakurti and C. Vuppalapati. Building an IoT Framework for Connected Dairy. In *2015 IEEE First International Conference on Big Data Computing Service and Applications*, pages 275–285, 3 2015.

[10] Fiona Edwards-Murphy, Michele Magno, Padraig M. Whelan, John OHalloran, and Emanuel M. Popovici. b+WSN: Smart Beehive with Preliminary Decision Tree Analysis for Agriculture and Honey Bee Health Monitoring. *Computers and Electronics in Agriculture*, 124:211–219, 2016.

[11] Nicholas Zinas, Sotirios Kontogiannis, George Kokkonis, Stavros Valsamidis, and Ioannis Kazanidis. Proposed open source architecture for long range monitoring. the case study of cattle tracking at pogoniani. In *Proceedings of the 21st Pan-Hellenic Conference on Informatics*, PCI 2017, New York, NY, USA, 2017. Association for Computing Machinery.

[12] G. Codeluppi, A. Cilfone, L. Davoli, and G. Ferrari. VegIoT Garden: a modular IoT Management Platform for Urban Vegetable Gardens. In *2019*

IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor), pages 121–126, Portici, Italy, 2019.

[13] Gaia Codeluppi, Antonio Cilfone, Luca Davoli, and Gianluigi Ferrari. LoRaFarM: A LoRaWAN-Based Smart Farming Modular IoT Architecture. *Sensors*, 20(7), 2020.

[14] G. Codeluppi, A. Cilfone, L. Davoli, and G. Ferrari. Ai at the edge: a smart gateway for greenhouse air temperature forecasting. In *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 348–353, 2020.

[15] Gaia Codeluppi, Luca Davoli, and Gianluigi Ferrari. Forecasting air temperature on edge devices with embedded ai. *Sensors*, 21(12), 2021.

[16] International Labour Organization. *Safety and health in agriculture*. International Labour Organization, third edition, December 1999.

[17] World meters. `http://www.worldometers.info`. Accessed: 2021-09-06.

[18] LI Min and ZHAO Liange. Agricultural labor force aging phenomenon and the effect on agricultural production:evidence from liaoning province. October 2009.

[19] Godwin Idoje, Tasos Dagiuklas, and Muddesar Iqbal. Survey for smart farming technologies: Challenges and issues. *Computers & Electrical Engineering*, 92:107104, 2021.

[20] European Commission (EC). Industry 4.0 in agriculture: Focus on iot aspects. *Digital Transformation Monitor*, July 2017.

[21] Jirapond Muangprathub, Nathaphon Boonnam, Siriwan Kajornkasirat, Narongsak Lekbangpong, Apirat Wanichsombat, and Pichetwut Nillaor. Iot and agriculture data analysis for smart farm. *Computers and Electronics in Agriculture*, 156:467–474, 2019.

[22] Manlio Bacco, Paolo Barsocchi, Erina Ferro, Alberto Gotta, and Massimiliano Ruggeri. The digitisation of agriculture: a survey of research activities on smart farming. *Array*, 3-4:100009, 2019.

[23] Michael J. O'Grady and Gregory M.P. O'Hare. Modelling the smart farm. *Information Processing in Agriculture*, 4(3):179–187, 2017.

[24] Subramania Ananda Kumar and Paramasivam Ilango. The impact of wireless sensor network in the field of precision agriculture: A review. *Wireless Personal Communications*, 98:685–698, 2018.

[25] M. Balasubramaniyan and C. Navaneethan. Applications of internet of things for smart farming – a survey. *Materials Today: Proceedings*, 2021.

[26] Panagiotis Radoglou-Grammatikis, Panagiotis Sarigiannidis, Thomas Lagkas, and Ioannis Moscholios. A compilation of uav applications for precision agriculture. *Computer Networks*, 172:107148, 2020.

[27] Juan Jesús Roldán, J. Cerro, D. Garzón-Ramos, PabloGarcia-Aunon, M. Garzón, Jorge de León, and A. Barrientos. Robots in agriculture: State of art and practical experiences. 2018.

[28] Antonis Tzounis, Nikolaos Katsoulas, Thomas Bartzanas, and Constantinos Kittas. Internet of Things in Agriculture, Recent Advances and Future Challenges. *Biosystems Engineering*, 164:31–48, 2017. doi:10.1016/j.biosystemseng.2017.09.007.

[29] Emerson Navarro, Nuno Costa, and António Pereira. A systematic review of iot solutions for smart farming. *Sensors*, 20(15), 2020.

[30] Zeynep Ünal. Smart farming becomes even smarter with deep learning—a bibliographical analysis. *IEEE Access*, 8:105587–105609, 2020.

[31] Alem Čolaković and Mesud Hadžialić. Internet of things (iot): A review of enabling technologies, challenges, and open research issues. *Computer Networks*, 144:17–39, 2018.

[32] Andrés Villa-Henriksen, Gareth T.C. Edwards, Liisa A. Pesonen, Ole Green, and Claus Aage Grøn Sørensen. Internet of things in arable farming: Implementation, applications, challenges and potential. *Biosystems Engineering*, 191:60–84, 2020.

[33] Mohamed Rawidean Mohd Kassim. Iot applications in smart agriculture: Issues and challenges. In *2020 IEEE Conference on Open Systems (ICOS)*, pages 19–24, 2020.

[34] Maanak Gupta, Mahmoud Abdelsalam, Sajad Khorsandroo, and Sudip Mittal. Security and privacy in smart farming: Challenges and opportunities. *IEEE Access*, 8:34564–34584, 2020.

[35] A.P. Barnes, I. Soto, V. Eory, B. Beck, A. Balafoutis, B. Sánchez, J. Vangeyte, S. Fountas, T. van der Wal, and M. Gómez-Barbero. Exploring the adoption of precision agricultural technologies: A cross regional study of eu farmers. *Land Use Policy*, 80:163–174, 2019.

[36] Bucci Giorgia, Bentivoglio Deborah, and Finco Adele. Factors affecting ict adoption in agriculture: A case study in italy. *Calitatea*, 20 (S2):122–129, 2019.

[37] Federica Caffaro and Eugenio Cavallo. The effects of individual variables, farming system characteristics and perceived barriers on actual use of smart farming technologies: Evidence from the piedmont region, northwestern italy. *Agriculture*, 9(5), 2019.

[38] J Blasch, B van der Kroon, P van Beukering, R Munster, S Fabiani, P Nino, and S Vanino. Farmer preferences for adopting precision farming technologies: a case study from Italy. *European Review of Agricultural Economics*, 12 2020. jbaa031.

[39] Olakunle Elijah, Tharek Abdul Rahman, Igbafe Orikumhi, Chee Yen Leow, and MHD Nour Hindia. An overview of internet of things (iot) and data

analytics in agriculture: Benefits and challenges. *IEEE Internet of Things Journal*, 5(5):3758–3773, 2018.

[40] Shivangi Vashi, Jyotsnamayee Ram, Janit Modi, Saurav Verma, and Chetana Prakash. Internet of things (iot): A vision, architectural elements, and security issues. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 492–496, 2017.

[41] Nallapaneni Manoj Kumar and Pradeep Kumar Mallick. The internet of things: Insights into the building blocks, component interactions, and architecture layers. *Procedia Computer Science*, 132:109–117, 2018. International Conference on Computational Intelligence and Data Science.

[42] In Lee. The internet of things for enterprises: An ecosystem, architecture, and iot service business model. *Internet of Things*, 7:100078, 2019.

[43] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015.

[44] Telecommunication Standardization sector of ITU (ITU-T). Recommendation itu-t y.2060, iot reference model, overview of the internet of things. Technical report, ITU-T, 2016.

[45] Alliance for Internet of Things Innovation WG03 (AIOTI). High level architecture (hla) release 2.1. Technical report, AIOTI, 2018.

[46] Mauro A. A. da Cruz, Joel José P. C. Rodrigues, Jalal Al-Muhtadi, Valery V. Korotaev, and Victor Hugo C. de Albuquerque. A reference model for internet of things middleware. *IEEE Internet of Things Journal*, 5(2):871–883, 2018.

[47] Arduino boards. `https://www.arduino.cc/en/main/boards`. Accessed: 2021-07-08.

[48] Raspberry Pi boards. `https://www.raspberrypi.org/products/`. Accessed: 2021-07-08.

[49] Pycom boards. `https://pycom.io/shop/#dev`. Accessed: 2021-07-08.

[50] Mike O. Ojo, Stefano Giordano, Gregorio Procissi, and Ilias N. Seitanidis. A review of low-end, middle-end, and high-end iot devices. *IEEE Access*, 6:70528–70554, 2018.

[51] ISO/IEC JTC 1 Information technology. Iso/iec 7498-1:1994 information technology — open systems interconnection — basic reference model: The basic model. ISO/IEC Standard 59, ISO/OSI, November 1994.

[52] Jeddou Sidna, Baina Amine, Najid Abdallah, and Hassan El Alami. Analysis and evaluation of communication protocols for iot applications. In *Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications*, SITA'20, New York, NY, USA, 2020. Association for Computing Machinery.

[53] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Comput. Surv.*, 51(6), January 2019.

[54] R. Peon M. Belshe and M. Thomson. Hypertext transfer protocol version 2 (http/2). rfc 7540. RFC, RFC Editor, 2015.

[55] Andrew Banks and Rahul Gupta (Ed.). Mqtt version 3.1.1. oasis standard. RFC, OASIS Standard, 29 October 2014.

[56] Object Management Group (OMG). Data distribution service (dds) version 1.4. RFC, OMG, March 2015.

[57] P. Saint-Andre. Extensible messaging and presence protocol (xmpp): Core. rfc 3920. RFC, RFC Editor, 2004.

[58] OASIS Standard. Advanced message queuing protocol (amqp) version 1.0. Technical report, OASIS Standard, 29 October 2012.

[59] K. Hartke Z. Shelby and C. Bormann. The constrained application protocol (coap). rfc 7252. RFC, RFC Editor, 2014.

[60] Google Cloud IoT. `https://cloud.google.com/solutions/iot`. Accessed: 2021-07-08.

[61] Amazon Web Services (AWS) IoT platform. `https://aws.amazon.com/it/iot/`. Accessed: 2021-07-08.

[62] IBM Watson IoT. `https://www.ibm.com/docs/en/watson-iot-platform?topic=getting-started`. Accessed: 2021-07-08.

[63] Microsoft azure iot hub. `https://azure.microsoft.com/en-us/overview/iot/`. Accessed: 2021-07-08.

[64] Openiot. `http://www.openiot.eu`. Accessed: 2021-07-08.

[65] Sitewhere. `https://sitewhere.io/en/`. Accessed: 2021-07-08.

[66] Openremote. `https://openremote.io`. Accessed: 2021-07-08.

[67] Thinger.io. `https://thinger.io`. Accessed: 2021-07-08.

[68] Fiware. `https://www.fiware.org`. Accessed: 2021-07-08.

[69] Souad Amghar, Safae Cherdal, and Salma Mouline. Which nosql database for iot applications? In *2018 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, pages 131–137, 2018.

[70] Sharvari Rautmare and D. M. Bhalerao. Mysql and nosql database comparison for iot application. In *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, pages 235–238, 2016.

[71] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[72] Haider Mahmood Jawad, Rosdiadee Nordin, Sadik Kamel Gharghan, Aqeel Mahmood Jawad, and Mahamod Ismail. Energy-Efficient Wireless Sensor Networks for Precision Agriculture: A Review. *Sensors*, 17(8), 2017.

[73] D. Davcev, K. Mitreski, S. Trajkovic, V. Nikolovski, and N. Koteli. IoT Agriculture System based on LoRaWAN. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 1–4, 6 2018.

[74] Li Wen, Zhang Qingfang, Huang Ke, Luo Xueke, and Guo Kai. Design of Agricultural Irrigation Hydropower Dual Control Intelligent Equipment Based on NB-IoT Technology. In *Proceedings of the 7th International Conference on Informatics, Environment, Energy and Applications*, IEEA '18, pages 42–48, New York, NY, USA, 2018. Association for Computing Machinery.

[75] Eli-Chukwu Ngozi Clara. Applications of artificial intelligence in agriculture: A review. *Engineering, Technology & Applied Science Research*, 9(4):4377–4383, 2019.

[76] N. N. Misra, Yash Dixit, Ahmad Al-Mallahi, Manreet Singh Bhullar, Rohit Upadhyay, and Alex Martynenko. Iot, big data and artificial intelligence in agriculture and food industry. *IEEE Internet of Things Journal*, pages 1–1, 2020.

[77] Andreas Kaplan and Michael Haenlein. Siri, siri, in my hand: Who's the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62(1):15–25, jan 2019.

[78] Y. Lee, P. Tsung, and M. Wu. Techology Trend of Edge AI. In *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–2, Hsinchu, Taiwan, 2018.

[79] Jesús Ferrero Bermejo, Juan F. Gómez Fernández, Fernando Olivencia Polo, and Adolfo Crespo Márquez. A Review of the Use of Artificial Neural Network Models for Energy and Reliability Prediction. A Study of the Solar PV, Hydraulic and Wind Energy Sources. *Appl. Sci*, 8:1844, 2019.

[80] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[81] Jenny Cifuentes, Geovanny Marulanda, Antonio Bello, and Javier Reneses. Air Temperature Forecasting Using Machine Learning Techniques: A Review. *Energies*, 13(4215), 2020.

[82] Eda Kavlakoglu. AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference? `https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks`. Accessed: 2021-03-02.

[83] M.W Gardner and S.R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627 – 2636, 1998.

[84] Sławomir Francik and Sławomir Kurpaska. The Use of Artificial Neural Networks for Forecasting of Air Temperature inside a Heated Foil Tunnel. *Sensors*, 20(3), 2020.

[85] Taewon Moon, Seojung Hong, Ha Young Choi, Dae Ho Jung, Se Hong Chang, and Jung Eek Son. Interpolation of Greenhouse Environment Data using Multilayer Perceptron. *Computers and Electronics in Agriculture*, 166:105023, 2019.

[86] Morteza Taki, Saman Abdanan Mehdizadeh, Abbas Rohani, Majid Rahnama, and Mostafa Rahmati-Joneidabad. Applied machine learning in greenhouse

simulation; new application and analysis. *Information Processing in Agriculture*, 5(2):253–268, 2018.

[87] Y. Yue, J. Quan, H. Zhao, and H. Wang. The Prediction of Greenhouse Temperature and Humidity Based on LM-RBF Network. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1537–1541, Changchun, China, 2018.

[88] Dae-Hyun Jung, Hyoung Seok Kim, Changho Jhin, Hak-Jin Kim, and Soo Hyun Park. Time-serial analysis of deep neural network models for prediction of climatic conditions inside a greenhouse. *Computers and Electronics in Agriculture*, 173:105402, 2020.

[89] Wang Hongkang, Li Li, Wu Yong, Meng Fanjia, Wang Haihua, and N.A. Sigrimis. Recurrent Neural Network Model for Prediction of Microclimate in Solar Greenhouse. *6th IFAC Conference on Bio-Robotics BIOROBOTICS 2018*, 51(17):790–795, 2018.

[90] Morteza Taki, Yahya Ajabshirchi, Seyed Faramarz Ranjbar, Abbas Rohani, and Mansour Matloobi. Heat transfer and MLP neural network models to predict inside environment variables and energy lost in a semi-solar greenhouse. *Energy and Buildings*, 110:314–329, 2016.

[91] Najmul Hassan, Kok-Lim Alvin Yau, and Celimuge Wu. Edge computing in 5g: A review. *IEEE Access*, 7:127276–127289, 2019.

[92] Yanjiao Chen, Baolin Zheng, Zihan Zhang, Qian Wang, Chao Shen, and Qian Zhang. Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions. *ACM Comput. Surv.*, 53(4), August 2020.

[93] Tejalal Choudhary, Mishra Vipul, Goswami Anurag, and Sarangapani Jagannathan. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 2020.

[94] STMicroelectronics. Ai expansion pack for stm32cubemx. `https://www.st.com/en/embedded-software/x-cube-ai.html`.

[95] Alexander Wong. Netscore: Towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge usage. In Fakhri Karray, Aurélio Campilho, and Alfred Yu, editors, *Image Analysis and Recognition*, pages 15–26, Cham, 2019. Springer International Publishing.

[96] Stm32 nucleo boards family. `https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html#overview`. Accessed: 2021-07-08.

[97] Launch pad^{tm} evaluation boards by texas instruments. `https://www.ti.com/design-resources/embedded-development/hardware-kits-boards.html`. Accessed: 2021-07-08.

[98] Pyboard Store. `https://store.micropython.org`. Accessed: 2021-07-08.

[99] Libelium plug&sense products family. `https://www.libelium.com/iot-products/plug-sense/`. Accessed: 2021-07-08.

[100] Mehmet Ali Erturk, Muhammed Ali Aydın, Muhammet Talha Buyukakkaslar, and Hayrettin Evirgen. A Survey on LoRaWAN Architecture, Protocol and Technologies. *Future Internet*, 11(10), 2019.

[101] Sigfox website. `https://www.sigfox.com/en`. Accessed: 2021-07-08.

[102] Rapeepat Ratasuk, Benny Vejlgaard, Nitin Mangalvedhe, and Amitava Ghosh. Nb-iot system for m2m communication. In *2016 IEEE Wireless Communications and Networking Conference*, pages 1–5, 2016.

[103] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. Overview of cellular lpwan technologies for iot deployment: Sigfox, lorawan, and nb-iot. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 197–202, 2018.

[104] Salim Jibrin Danbatta and Asaf Varol. Comparison of zigbee, z-wave, wi-fi, and bluetooth wireless technologies used in home automation. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–5, 2019.

[105] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi. In *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, pages 46–51, 2007.

[106] J.A. López-Riquelme, N. Pavón-Pulido, H. Navarro-Hellín, F. Soto-Valles, and R. Torres-Sánchez. A software architecture based on fiware cloud for precision agriculture. *Agricultural Water Management*, 183:123–135, 2017. Special Issue: Advances on ICTs for Water Management in Agriculture.

[107] J. Diego Franco, Tania A. Ramirez-delReal, Daniel Villanueva, Araceli Gárate-García, and Dagoberto Armenta-Medina. Monitoring of ocimum basilicum seeds growth with image processing and fuzzy logic techniques based on cloudino-iot and fiware platforms. *Computers and Electronics in Agriculture*, 173:105389, 2020.

[108] Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 1st edition, February 2015.

[109] R. T. FIELDING. Rest : Architectural styles and the design of network-based software architectures. *Doctoral dissertation, University of California*, 2000.

[110] Nitin Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7, 2017.

[111] Andreas Kamilaris, Andreas Kartakoullis, and Francesc X. Prenafeta-Boldú. A review on the practice of big data analysis in agriculture. *Computers and Electronics in Agriculture*, 143:23–37, 2017.

[112] Veg growing — guide edition. https://www.teagasc.ie/media/website/publications/2017/Veg_Growing_Guide_Edition_8.pdf. Accessed: 2021-09-01.

[113] CC3200 LAUNCHXL. http://www.ti.com/lit/ds/symlink/cc3200.pdf. Accessed: 2021-09-01.

[114] AM2032 Datasheet - Air Tumidity and Temperature Sensor. https://datasheetspdf.com/parts/AM2302.pdf?id=942482. Accessed: 2020-02-07.

[115] Soil moisture sensor module. https://www.sparkfun.com/products/13322. Accessed: 2020-02-07.

[116] DS18B20 Datasheet - Soil Temperature I2C Sensor. https://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html. Accessed: 2020-02-07.

[117] P. Adams and L. C. Ho. Effects of environment on the uptake and distribution of calcium in tomato and on the incidence of blossom-end rot. *Plant and Soil*, 154(1):127–132, Jul 1993. doi:10.1007/BF00011081.

[118] V. P. Venkatesan, C. P. Devi, and M. Sivaranjani. Design of a Smart Gateway Solution based on the Exploration of Specific Challenges in IoT. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 22–31, 2 2017.

[119] G. Aloi, G. Caliciuri, G. Fortino, R. Gravina, P. Pace, W. Russo, and C. Savaglio. A Mobile Multi-Technology Gateway to Enable IoT Interoperability. In *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 259–264, 4 2016.

[120] Podere Campáz - Produzioni Biologiche. https://www.poderecampaz.com. Accessed: 2021-02-07.

[121] LoPy Datasheet v1.0. https://cdn.sparkfun.com/assets/e/b/2/9/0/lopy4-specsheet.pdf. Accessed: 2021-02-24.

[122] DFRobot Soil Moisture Sensor Datasheet. https://docs.rs-online.com/8389/A700000007238462.pdf. Accessed: 2020-02-09.

[123] Solar LiPo Charger - Schematic. http://image.dfrobot.com/image/data/DFR0264/Solar%20Charger%20V1.0%20SCH.pdf. Accessed: 2020-02-07.

[124] The Things Industries. The Things Network (TTN). https://www.thethingsnetwork.org. Accessed: 2020-02-07.

[125] Texas Instruments SimpleLink CC3100/CC3200 Wi-Fi Internet-on-a-chip Networking Sub-system Power Management. http://www.ti.com/lit/an/swra462/swra462.pdf. Accessed: 2020-02-07.

[126] Anil Kumar, G. N. Tiwari, Subodh Kumar, and Mukesh Pandey. Role of Greenhouse Technology in Agricultural Engineering. *International Journal of Agricultural Research*, 5(9):779–787, 2010.

[127] Axel Escamilla-García, Genaro M. Soto-Zarazúa, Manuel Toledano-Ayala, Edgar Rivas-Araiza, and Abraham Gastélum-Barrios. Applications of Artificial Neural Networks in Greenhouse Technology and Overview for Smart Agriculture Development. *Applied Sciences*, 10(3835), 2020.

[128] G.P.A. Bot. Physical Modeling of Greenhouse Climate. *IFAC Proceedings Volumes*, 24(11):7–12, 1991.

[129] Laura Belli, Simone Cirani, Luca Davoli, Lorenzo Melegari, Màrius Mónton, and Marco Picone. An Open-Source Cloud Architecture for Big Stream IoT Applications. In Ivana Podnar Žarko, Krešimir Pripužić, and Martin Serrano, editors, *Interoperability and Open-Source Solutions for the Internet of Things: International Workshop, FP7 OpenIoT Project, Held in Conjunction with*

SoftCOM 2014, Split, Croatia, September 18, 2014, Invited Papers, pages 73–88. Springer International Publishing, Sep 2015.

[130] Aarti Kochhar and Naresh Kumar. Wireless sensor networks for greenhouses: An end-to-end review. *Computers and Electronics in Agriculture*, 163:104877, 2019.

[131] M. Abbasi, M. H. Yaghmaee, and F. Rahnama. Internet of things in agriculture: A survey. In *2019 3rd International Conference on Internet of Things and Applications (IoT)*, pages 1–12, Isfahan, Iran, 2019. doi:10.1109/IICITA.2019.8808839.

[132] Luca Davoli, Laura Belli, Antonio Cilfone, and Gianluigi Ferrari. Integration of Wi-Fi mobile nodes in a Web of Things Testbed. *ICT Express*, 2(3):95–99, 2016. Special Issue on ICT Convergence in the Internet of Things (IoT).

[133] Z. Tafa, F. Ramadani, and B. Cakolli. The Design of a ZigBee-Based Greenhouse Monitoring System. In *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4, Budva, Montenegro, 2018.

[134] M. T. W. Wiboonjaroen and T. Sooknuan. The Implementation of PI Controller for Evaporative Cooling System in Controlled Environment Greenhouse. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 852–855, Jeju, South Korea, 2017.

[135] Z. Zou, Y. Bie, and M. Zhou. Design of an Intelligent Control System for Greenhouse. In *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pages 1–1635, Xi'an, China, 2018.

[136] DarkSky Open Weather Data Repository. `https://darksky.net/dev`. Accessed: 2021-02-20.

[137] Monica Franzese and Antonella Iuliano. Correlation Analysis. In Shoba Ranganathan, Michael Gribskov, Kenta Nakai, and Christian Schonbach,

editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 706–721. Academic Press, Oxford, 2019.

[138] Vittorio Mazzia, Aleem Khaliq, Francesco Salvetti, and Marcello Chiaberge. Real-time apple detection system using embedded systems with hardware accelerators: An edge ai application. *IEEE Access*, 8:9102–9114, 2020.

[139] Dmitrii Shadrin, Alexander Menshchikov, Dmitry Ermilov, and Andrey Somov. Designing future precision agriculture: Detection of seeds germination using artificial intelligence on a low-power embedded system. *IEEE Sensors Journal*, 19(23):11573–11582, 2019.

[140] François Chollet et al. Keras. `https://keras.io`, 2015.

[141] Aggregate Farming in the Cloud (AFarCloud) H2020 Project. `http://www.afarcloud.eu`. Accessed: 2021-02-14.