**UNIVERSITÉ LIBRE DE BRUXELLES**
**Faculté des Sciences**
**Département d'Informatique**

# Digital Twin for Plant Health Monitoring

Contuliano Bravo Rayan

**Promotors :**
Prof. Gianluca Bontempi
Mr. Tribel Pascal

Preparatory work for the Master Thesis in Computer Sciences

**Academic year 2024 – 2025**

# Contents

# 1 Introduction

## 1.1 Background & Motivation

Agriculture has always been a central point of our society. It is the basis of our food supply and has a significant impact on our economy. Considering this, it is important to ensure that the agricultural sector and its processes keep getting as efficient and sustainable as possible. However one might face several challenges. The world population is growing and the demand for food is increasing. At the same time, the available agricultural land and growth are decreasing due to urbanization and climate change [6] [17]. Moreover, The Food and Agriculture Organization of the United Nations (FAO) estimates that plant diseases cost the global economy around $220 billion per year [19]. The ability to optimize agricultural processes (e.g. irrigation, fertilization, ...) and to detect and treat plant diseases in a timely manner is crucial to ensure agricultural sustainability.

Beyond the needs of agricultural sustainability, the preservation of biodiversity is also vital. Many plant species, including endangered species, face threats from environmental changes and human activity [29, 27]. Active and continuous monitoring plays a crucial role in conserving these species by making it possible not only to track their growth, but also to detect and address signs of stress or disease.

Traditionally, plant health monitoring has relied on manual inspections and visual assessments, which can be time consuming, often subjective [7] and surely error-prone. An important drawback of this approach is that it does not allow for real-time monitoring of the plants.

In response to these challenges, the agricultural sector is increasingly evolving towards digitalization and automation, this is often referred as *smart farming* or *Agriculture 4.0* [22]. This involves the use of advanced technologies such as the Internet of Things (IoT), artificial intelligence (AI), machine learning, and big data analytics to improve agricultural processes [10]. This is where the digital twin concept comes into play.

In this context, Digital twins stands out as a key area of research in modern agriculture. It gives an approach to detect early stage of diseases, plant stress, nutrient deficiencies or other factors affecting the plant's well-being. Its application extends beyond the simple detection of diseases as it provides farmers with the precise information they need to make informed decisions about their crops which facilitate the resource allocation (e.g. irrigation, lighting, fertilization, pest control, ...) and reduces waste and environments pollution.

## 1.2 Digital Twin Concept

A digital twin is a virtual representation of a physical object, that not only passively reflects the state of its physical counterpart, but also actively interacts with it through feedback and control mechanisms. In other words, the digital twin is not merely an observer: every change in the physical object is mirrored in the digital twin, and, decisions made by the digital twin can be communicated back to the physical entity to optimize its performance or response to environmental changes. To achieve this, the digital twin is connected in real-time to the physical object.

A digital twin consists of multiple elements:

- **Physical Entity**: The physical object that is being monitored or controlled that exists in the real world. In the context of agriculture, this could be a plant, a field,

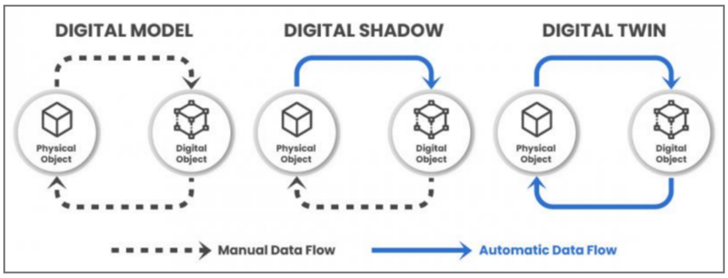


Figure 1: Different levels of digital twinning. Source: Open Health Systems Lab

or an entire farm.

- **Digital Entity** : The virtual representation of the physical entity. It is a digital model that simulates the behavior and characteristics of the physical entity. It is often created using data from sensors and other sources.
- **Data**: The information collected from the physical entity. This can include sensor readings and other relevant information. The data is used to update and synchronize the digital entity and to make decisions about the physical entity. The data can be issued from several sources such as
  - **Sensors**: Devices that collect data from the physical entity. In agriculture, this can include soil moisture , temperature , humidity , and other types of sensors.
  - **External Data**: Data from external sources that can be used to enhance the digital twin's context such as weather data [28].
  - **Historical Data** : Data from previous observations and measurements that can be used to improve the digital twin's accuracy and reliability.
- **Communication** : The real-time transmission of data and control commands between the physical and digital entities, typically over IoT networks [24]. This bidirectional communication is fundamental, enabling the digital twin not only to monitor but also to act, for example by sending irrigation or lighting adjustments to optimize plant health. This capacity for decision-making and feedback distinguishes the digital twin from earlier paradigms such as digital models or digital shadows [9].
- **Services** : The digital twin can provide various services to the user, such as monitoring, data analysis, and (remote) control. These services are designed to enhance the lifecyle of the physical entity and to make better decisions. For example, the digital twin can provide real-time monitoring of the plant's health and growth, provide recommendations for irrigation, fertilization, and pest control, or even run "what-if" scenarios to predict the impact of different management strategies on the plant's performance [16, 24] .

Digital twins are used in various industries, including manufacturing, healthcare, and transportation.

## 1.3 Project Aim and Scope

The aim of this work is to design and implement a modular, scalable digital twin prototype for plant health monitoring. This prototype will enable real-time tracking of plant health parameters, and provide a user-accessible dashboard for data visualization and interaction with the digital twin. The system will support the acquisition and integration of environmental data, and lay the groundwork for future decision support (e.g., recommendations for irrigation, lighting, and heating). The primary focus of this work is on the hardware and software implementation of a flexible monitoring platform, which will serve as a foundation for advanced analytics and decision-making tools in future researches.

## 1.4 Report Structure

In the following sections, we will review the current state of the art of the digital twin technology in agriculture, outline key considerations for designing a digital twin system, describe the system architecture and conclude with a discussion of future work and a conclusion.

# 2 State of the Art

This section will present the state of the art of the digital twin technology and its application in agriculture. We will also discuss the Internet of Things (IoT) and its role in plant monitoring, as well as data management in real-time environments and machine learning in plant health monitoring.

## 2.1 Digital Twins in agriculture

Digital twins have shown great potential in various fields such as manufacturing and healthcare. In manufacturing, digital twins are widely used for predictive maintenance, equipment monitoring, and process optimization [26] . In healthcare, they support applications ranging from personalized medicine to medical device design and hospital management [12] In contrast, the adoption of digital twins in agriculture has been more limited, with most documented applications still in early development and not widely adopted in the industry. One of the main challenges is the complexity involved in accurately modeling living organisms (like plants) and their interactions with non-living elements (such as soil,sunlight, water, or equipment). Indeed, living organisms can affect their environment (like soil, water, ...) and those environmental factors can also affect the living organisms. Due to this bidirectional interaction, it is harder to build a digital twin for agriculture than for areas where only machines are involved [23].

In the following section we describe some relevant works in the field of digital twins for agriculture.

### 2.1.1 Plant Digital Twins

**Multi-agent DT for Wheat** The Multi-agent DT for Wheat [25] project is an approach to model the entire wheat life-cycle using a knowledge base and a multi-agent system. The multi-agent approach was chosen because plant growth and development are ruled by multiple biological process that vary across different growth stage. It aims to overcome the limitation of data-driven models which can become ineffective at predicting and classifying crop states as the models and reality diverge due to climate change, soil degradation, and other factors.

The systems consists of multiple agents that are in charge of tracking different stages of the wheat life-cycle. They compare incoming sensor readings, such as temperature, soil moisture, or weather conditions, against scientifically established thresholds for healthy development at that stage. If the data diverges from the expected values, the system will alert the user and suggest actions to take.

Another important aspect is that the system's suggestion can be corrected by experts in the field. This feedback is then used to improve the system's performance and accuracy.

**3D Functional Plant Modeling** A significant advancement in plant-level digital twins is proposed by [14] by incorporating, additionally to the "simple" environmental monitoring, **Functional Structural Plant Models** (FSPMs) into the digital twin framework. The goal is to enhance the predictive, monitoring and decision-making capabilities of the digital twin by incorporating a detailed, dynamic 3D representation of the plant's structure and its physiological functions (like photosynthesis, nutrient uptake, and growth).

The current limitation of most agricultural digital twins is that they lack the ability to predict the yield or disease progression accurately because they do not consider the plant's structure and its interactions with the environment. The proposed framework aims to address this limitation by integrating FSPMs into the digital twin, allowing for a more comprehensive understanding of the plant's growth and development.

This approach enables the researchers not only to monitor plant health, but also to predict plant's growth, simulate interventions and optimize growing conditions based on the plant's physiological responses with its environment.

### 2.1.2 Greenhouse Digital Twins

Greenhouses are controlled environments that are used to grow plants in a protected way (i.e by shielding them from external factors such as extreme weather, pests and diseases using regulated climate control and physical barriers). They are at a smaller scale than farms and are often used to grow plants that are not native to the region.

**GreenhouseDT** The GreenhouseDT [11] project is a digital twin framework for greenhouses that aims to provide a modular and extensible architecture for **low-costs** greenhouse management. It is designed to be easily reproducible and adaptable to different greenhouse environments in order to facilitate the study and comparison of different self-adaptive control strategies without the need for expensive equipment and/or proprietary software. It uses the plants, sensors and water pumps as a physical twin connected to a Raspberry PI (a single board computer). The data is stored in a InfluxDB (a high-performance time-series database) and visualized in a web-based dashboard enabling the user to monitor, control and simulate the greenhouse environment.

### 2.1.3 Field and Crop Digital Twins

Field and crop digital twins are used to monitor and manage crops in a larger scale than greenhouses. They are used to monitor the health of crops, detect diseases, and optimize resources used to manage those crops. Because of the larger scale, they are often used in conjunction with drones and satellites to collect data from the field.

**AgriLoRa** The AgriLoRa [4] project was created to offer a digital twin platform for monitoring and managing crops in real time, with an emphasis on scalability and efficient resource utilization.

The system relies on wireless sensor networks using the **Long Range Wide Area Network** (LoRaWAN) protocol (a wireless communication technique for battery-operated IoT devices), which is well-suited for covering large fields while keeping power usage low. These sensors gather data directly from the fields and send it back for analysis. To turn this data into practical insights, the system applies machine and deep learning and computer-vision algorithms, helping farmers make decisions.

All data is stored in the cloud, where it's organized, processed, and made accessible through a web dashboard. In addition to sensor readings, the system also adds drone imagery and satellite data to give a complete view of crop health and field conditions. AgriLoRa not only helps monitor crops but also offers useful recommendations for watering, fertilization, and disease management.

**TWINSOR** The TWINSOR [13] project provides a digital twin framework for small and medium-sized farms where the goal is to help farmers visualize and analyze multi-sensor data from their fields and integrate predictive models, such as disease prediction. TWINSOR is co-founded by the European Union supporting the "Farm to Fork" strategy aiming to reduce the use of pesticides and fertilizers to enhance the sustainability of agriculture.

The system is designed to be modular and extensible, allowing farmers to add new sensors and models as needed. Additionally, the development of the system is user-centered, meaning that farmers, agricultural technicians are involved in the design. This approach aims to ensure that the system meets every needs of the users and is easy to use even for those with limited technical knowledge. The mobile application (Agri-Dash) can be used to receive alerts and management recommendations from the data analysis service, it allows farmers and agricultural technicians to communicate and share information about the field and the crops.

## 2.2 IoT for Plant Monitoring

The Internet of Things (IoT) is a network of physical devices, vehicles, buildings, and other objects that are embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet. The IoT is an essential components of digital twins, as it enables the real-time monitoring and control of physical assets and processes.

The IoT is used in agriculture to monitor and manage crops, livestock, and equipment. It enables farmers to collect and analyze data from their fields, greenhouses in real time, allowing them to make informed decisions about their operations. It relies on a wide variety of sensors to capture critical environmental and plant health data. The selection of sensors depends on the specific application and the type of data that needs to be collected.

### 2.2.1 Environmental Sensors

Environmental sensors play a vital role in monitoring the key conditions that affect plant growth. These sensors help to ensure that plants are cultivated in optimal environments by continuously tracking various parameters. Common types of environmental sensors include:

- **Temperature** and **humidity** sensors: Devices such as the DHT11 and DHT22 are the most common sensors used in low-cost IoT applications. They are used to measure the temperature and humidity of the air which are critical factors influencing plant growth [11, 2, 3].

- **Soil moisture** sensors: They measure the amount of water contained in the soil to ensure plants receive adequate water for optimal growth. The most common soil moisture sensors are capacitive and resistive sensors. Capacitive sensors measure the dielectric constant of the soil, while resistive sensors measure the electrical resistance of the soil. Capacitive sensors are generally more accurate and reliable than resistive sensors, but they are also more expensive [11].

- **Light**: To monitor how much sunlight plants are receiving, light sensors such as Light Dependent Resistors (LDRs) and Digital Light Sensors (DLS) are commonly used. LDRs offer a simple and low-cost solution, whereas DLS devices provide more precise measurements of light intensity, which is vital for processes like photosynthesis and healthy plant growth [11].

- **pH** sensors: Measuring the acidity or alkalinity of soil is crucial, as it directly affects nutrient availability. They are typically based on electrochemical principles and can be either analog or digital [2].

- **CO2** sensors: These sensors are used to measure the concentration of carbon dioxide in the environment, which is a critical factor for photosynthesis, maintaining appropriate levels is critical for plant productivity [5].

- **Nutrient levels** sensors: NPK sensors are used to measure the levels of essential nutrients like nitrogen, phosphorus, and potassium in the soil. These sensors help ensure that plants receive the necessary nutrients for optimal growth. It is essential for optimizing fertilization and preventing nutrient deficiencies or excesses [2].

### 2.2.2 Plant Health Sensors

Plant health sensors are used to monitor the health and growth of plants by tracking various physiological parameters of the plants. Here are some examples of plant health sensors:

- **RGB, Multispectral/Hyperspectrals, Thermals Cameras**: These type of cameras are used to capture images of plants. They can be used to monitor plant growth, shape, size, color, detect diseases, and assess overall plant health. They are often used in conjunction with computer vision algorithms to analyze the images and extract relevant information about the plants [14].

- **LiDAR** (Light Detection and Ranging) sensors: They are used to measure the distance between the sensor and the plant, allowing for the creation of 3D models of the plants. This information can be used to create accurate representations of the plants' structure and growth patterns [14]. However, **LiDAR sensors can be significantly more expensive than simpler imaging solutions**, with commercial terrestrial LiDAR systems often costing around $10,000 [8]. Nonetheless, it is possible to use low-cost LiDAR sensors (e.g., $100-$500) [18], but these generally have limited range and resolution compared to professional systems . Hence, the use of LiDAR in research or commercial agriculture is often justified only for applications where detailed 3D structural data is essential.

### 2.2.3 Connectivity

The connectivity of the sensors and the communication between the sensors and the digital twin is a critical aspect of the IoT as it determines the effectiveness of the monitoring and control system. Several connectivity options are available for IoT applications in agriculture:

- **LoRaWAN** (Long Range Wide Area Network) is a low-power, wide-area network technology. Its ability to transmit data over several kilometers makes it particularly suitable for agricultural fields where sensors are distributed over wide areas, far from the main control center [4]

- **NB-IoT** (Narrowband Internet of Things) is a cellular technology that is designed for low-power, wide-area IoT applications. Like LoRaWAN, it allows for long-range communication between devices, making it ideal for agricultural applications where sensors may be located far from the main control system. [13]

- **Wi-Fi** is a wireless communication protocol that is of a common usage in homes an regular networks. It can be used in agricultural applications where sensors are located close to the main control system under the same Wi-Fi. [5]

- **Bluetooth** is a short-range wireless communication protocol that is commonly used in consumer electronics. It can also be used in agricultural applications where sensors are located close to the main control system.

- **Zigbee** is a low-cost, low-power, wireless communication protocol that is designed for short-range communication between devices. It is often used in home automation and industrial applications. It can also be used in agricultural applications where sensors are battery-powered and located close to the main control system. [5]

- **Wired connections**, despite the increasing use of wireless technology, are still widely used in greenhouses and small environments. They are often easier to implement and maintain, providing stable and interference-free data transmission. [11] [5]

# 3 Analysis

Designing and implementing a Digital Twin system can be a complex task, as it demands the effective integration of both hardware, such as sensors and actuators, and software components, such as data preprocessing, machine learning, storage and visualization. Additionally, the system must support efficient real-time data management and offer a user-friendly interface for interaction and visualization. This section describes the design considerations and architectural choices for the digital twin system, addressing the requirements for streaming data management (real-time data collection, preprocessing and storage) and the future integration of machine learning algorithms for data analysis and decision support. The practical implementation details of the system are described later in the Section 4.

## 3.1 Requirements and Challenges

While designing the digital twin system, one must consider the following requirements and challenges:

- **Real-Time Data Collection**: The system must be capable of collecting and processing data in real-time to ensure timely decision-making. Immediate data collection and processing ensure that there are no missed opportunities for intervention, such as adjusting irrigation or lighting based on environmental changes.
- **Reliability**: The fidelity of the decision making and the overall system performance depends on the reliability of the data collection. Sensors drifts, failures or calibration issues can lead to inaccurate data and misleading decisions.
- **Modular Design**: To support flexibility and future expansion, the architecture should be modular. This enables straightforward integration of additional sensors, actuators, or data processing modules. Clear separation of data collection, processing, and storage functionalities facilitates maintenance and incremental upgrades, allowing the system to evolve alongside new requirements or technological advances without necessitating a complete redesign.
- **Scalability**: As the system expands, incorporating more sensors or managing larger datasets, it should remain responsive and efficient. The architecture must be capable of scaling horizontally to handle increasing data loads without sacrificing real-time performance.

### 3.1.1 Formal definition of the problem and Objectives

A precise definition of system objectives forms the basis for architectural choices.

Given a physical plant $P$ evolving in a continuously changing environment $E$. The goal is to develop a digital twin $D$ that is a dynamic, real-time virtual replica of the environmental context and state of the plant.

Let $t$ denote the current time instant at which measurements are observed.

We can define the state of the environment as

$$E(t) = \{L(t), M(t), T(t), H(t)\}$$

where:

- $L(t)$: **Light Intensity**, measured in **Lux** (lx), is the amount of light that the plant receives at time $t$. $L(t) \in [0, 65000]$
- $M(t)$: **Soil Moisture**, expressed in **percentage** (%), is the amount of water that is present in the soil at time $t$. $M(t) \in [0, 100]$
- $T(t)$: **Temperature**, measured in **degrees Celsius** (°C), is the temperature of the environment of the plant at time $t$. $T(t) \in [-40, 80]$
- $H(t)$: **Humidity**, expressed in **percentage** (%), is the amount of water vapor that is present in the air at time $t$. $H(t) \in [0, 100]$

The plant $P$ can be characterized by its observable state $O(t)$, which evolves with $t$, and can be described by a set of observable variables. The observable state of the plant is defined as

$$O(t) = \{H(t), A(t), C(t)\}$$

[1] where:

- $H(t)$: **Plant Height**, measured in **centimeter** (cm), is the vertical extent of the plant at time $t$.
- $A(t)$: **Leaf Area**, measured in **square centimeter** ($cm^2$), is the total observed area of leaves at time $t$.
- $C(t)$: **Global Coloring**, expressed as a color vector in **RGB** space, representing the distribution of the plant colors at time $t$. $C(t) \in ([0, 255]^3)^k$ where $k$ is the number of dominant colors extracted.

The choice of those observable variables was guided by both biological relevance [20, 21] and practical ability to measure them using computer vision algorithms [15].

All environmental state variables in $E(t)$ are directly measured via sensors integrated with a Raspberry Pi (e.g. light, temperature, humidity, and soil moisture). The plant observable state variables in $O(t)$ (height, leaf area, and global coloring) are estimated from images using computer vision algorithms. No hidden or latent state variables are considered at this stage, and all defined parameters are either directly observed or estimated from sensor or image data.

The state of the digital twin is then defined as $S(t)$, a function of the state of the environment and the observable state of the plant:

$$S(t) = f(E(t), O(t))$$

where:

- $f$ is a function that maps the state of the environment and the observable state of the plant to the state of the digital twin.

The digital twin system $D$ will consists of multiple modules that work together, for more details refer to the Section 3.2. The solution is intended to be modular and scalable, allowing for the addition of new sensors and machine learning algorithms in the future.

[1] Note that the range of measurement of $H(t)$ and $A(t)$ is not provided as it will highly depend on the monitored plant

## 3.2 System Architecture

The system architecture (Figure 2), is designed to meet the requirements and challenges outlined above. It has multiple components that work together to provide a seamless experience for monitoring and plant health analytics :

- **Sensors and Data Collection**: Data is captured by physical sensors positioned around the plant. These sensors are coordinated by a Sensor Manager running on a Raspberry Pi, which abstracts hardware specifics and ensure smooth communication with all connected devices.
- **Data Streaming and Preprocessing**: Sensor readings are continuously forwarded to a preprocessing module. This stage is crucial, as it cleans the incoming data, removing noise, handling missing entries, and standardizing formats, so that subsequent analytics are based on accurate and reliable information.
- **Data Storage**: Once preprocessed, all data is archived in a time-series database. This specialized storage solution is optimized for managing large volumes of timestamped records, supporting both rapid queries and long-term analysis.
- **Analytics & AI Module**: It is responsible for performing advanced analysis on the data collected. It can use machine learning algorithms to detect trends, anomalies and make predictions about the plant health.
- **Web dashboard**: The results and real-time data are presented to users through an intuitive web dashboard. This user interface focuses on accessibility, delivering clear visuals and real-time alerts when some events are detected. Moreover, it enables the user to interact directly with the system's functionalities such as controlling the actuators or machine learning models.

A key strength of this architecture is its independence between modules, each can be implemented, tested, and scaled separately. While the prototype is deployed on a Raspberry Pi, the modular design allows for straightforward migration to cloud services or more powerful servers if the scale of of the projectf increases in the future.

## 3.3 Data Streaming and Storage Strategy

As highlighted previously, the ability to collect and process data in real time is essential for an effective digital twin system. To meet these requirements, this system implements a robust architecture for data streaming and storage, supporting both real-time monitoring and long-term historical analysis.

### 3.3.1 Real-Time Data Streaming Collection

Monitoring plant health generates continuous streams of timestamped data from multiple sensors. Immediate data collection and processing ensure that there are no missed opportunities for intervention. In the system architecture (see Figure 2), sensor readings are initially gathered by a Sensor Manager application running on a Raspberry Pi. This data is then forwarded to an **Apache Kafka**[2] message broker, which acts as a central hub for data exchange.

[2] https://kafka.apache.org/

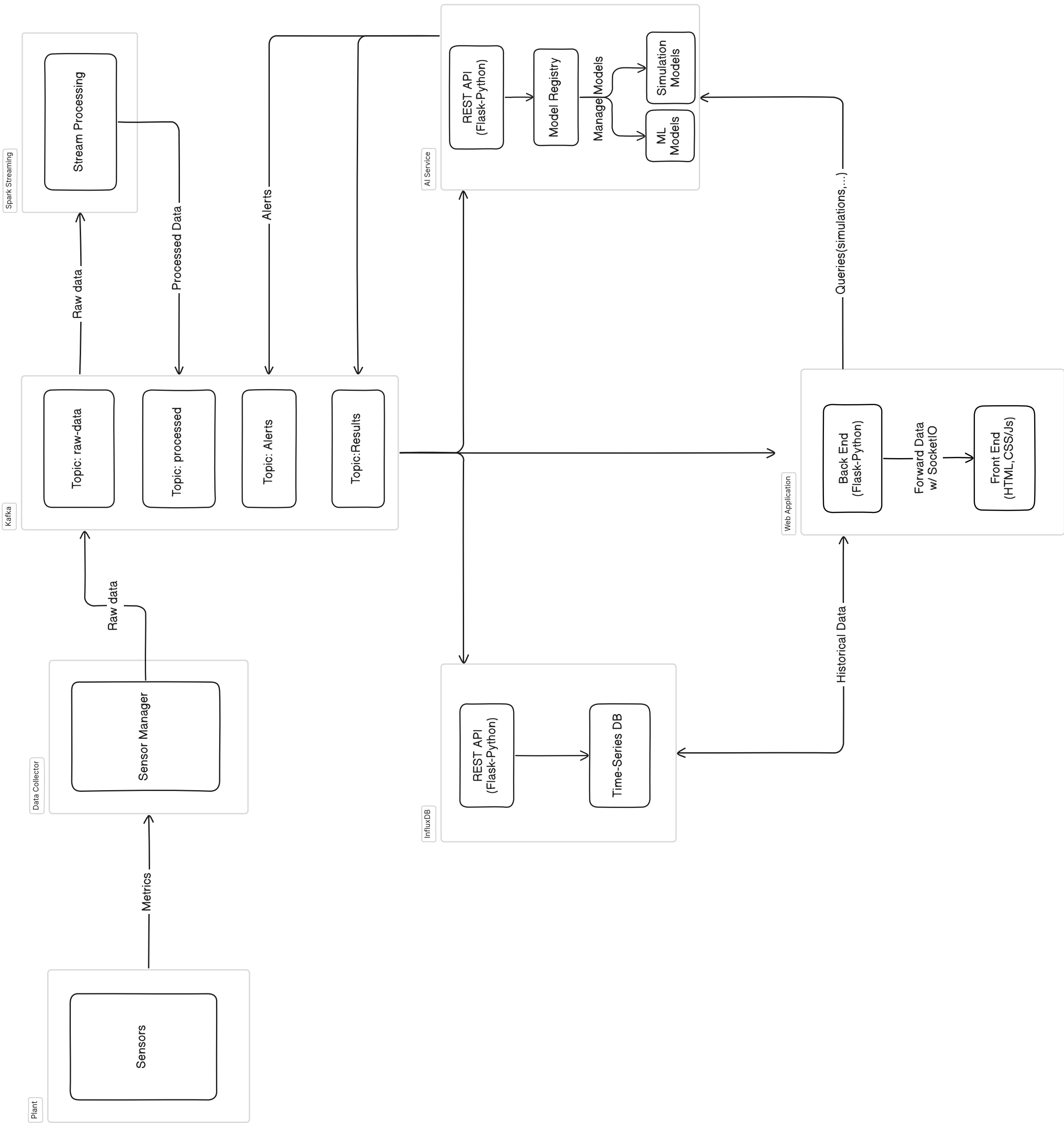


Figure 2: System architecture for the digital twin system.

Kafka is a distributed event streaming platform that is designed to handle high-throughput, low-latency data streams. Unlike simpler messaging protocols such as MQTT [3], Kafka provides advanced features including data replication, retention policies, and robust fault tolerance. Its publish-subscribe model allows independent modules, such as preprocessing, storage, analytics, or dashboards to consume and process data in real time. The integration of Kafka brings several advantages to the digital twin system:

- **Modularity**: Each subsystem, such as sensors, processing, storage, analytics, acts independently and communicates via standardized Kafka topics, making it easier to develop, test, and deploy each component separately.
- **Extensibility**: Additional analytics or visualization modules can be integrated simply by subscribing to relevant data streams, without disrupting existing workflows.
- **Scalability**: Kafka can handle large and growing data volumes by distributing workloads across multiple nodes, enabling the system to scale horizontally.
- **Fault Tolerance**: Kafka ensures resilience by replicating data across multiple nodes, which protects against data loss in the event of system failures.

Using multiple Kafka topics, the architecture can efficiently organize and direct various types of data for processing and storage.

### 3.3.2 Data Storage

Efficient storage and retrieval of time-series data is critical for both real-time analytics and historical trend analysis. For this purpose, the system leverages **InfluxDB**[4], an open-source time-series database. InfluxDB offers seamless integration with Python, a user-friendly web interface, and flexibility for cloud deployment. Given the rapid growth of real-time data, InfluxDB's provides features such as down-sampling, retention policies, and data aggregation that enable efficient long-term management by reducing storage costs and maintaining query performance.

Although more traditional relational databases (such as **MySQL**[5] or **SQLite**[6]) are familiar to many developers, they are less suited to the unique demands of time-series data due to potential performance bottlenecks when handling large, high-frequency datasets.

### 3.3.3 Data Preprocessing

Additionally to Kafka that provides a real-time data streaming platform, the need to preprocess the data in real-time is also important to have reliable data The system uses **Apache Spark**[7] Structured Streaming that allows to process large amount of data in real-time and provides a high-level API for processing different data streams, thus allowing to easily combine different data sources (Kafka, InfluxDB) to perform complex data preprocessing tasks based on real-time data and historical data.

[3] `https://mqtt.org/`
[4] `https://docs.influxdata.com/influxdb/v2/`
[5] `https://www.mysql.com/`
[6] `https://sqlite.org/`
[7] `https://spark.apache.org/)`

## 3.4 ML Applications

While this preparatory work focuses on robust data collection and visualization, the next steps will be to integrate machine learning (ML) algorithms and compare them. Machine learning is a key component of the digital twin system, as it allows to analyze the data collected and make predictions or decision about the plant health. As the volume of historical data grows, the system can take advantage of increasingly sophisticated models to improve both accuracy and functionality. There are several promising areas where ML can play a key role in plant health monitoring:

### 3.4.1 Anomaly Detection

ML-based anomaly detection algorithms can continuously analyze both streaming and historical sensor readings to automatically identify unusual patterns or outliers. For instance, a sudden drop in temperature or spike in humidity may signal environmental issues or sensor malfunctions. When such anomalies are detected, the system can proactively alert users, helping to minimize risks to plant health. Simple statistical methods (such as moving averages or thresholds) may be used at first, with more advanced models deployed as data availability increases.

### 3.4.2 Predictions

By forecasting future conditions based on current and historical data, the system can support smarter decision-making. For example, predictive models could estimate when soil moisture levels will fall below optimal thresholds, indicating when to water the plant. Regression and time-series forecasting algorithms are well-suited to this task, enabling proactive rather than reactive interventions.

### 3.4.3 Classification

Another important application for ML is the automated classification of plant health. By analyzing patterns across various sensor readings, the system can determine whether a plant is likely healthy, stressed, or unhealthy. This classification enables the system to suggest or even take corrective actions, such as changing environmental parameters to improve plant well-being.

### 3.4.4 Disease Detection

Machine learning, particularly computer vision techniques, can assist in identifying early signs of plant disease. By processing images from cameras placed near plants, the system could detect symptoms like unusual spots or changes in leaf coloration. Early detection allows for rapid intervention, reducing the spread and impact of disease. Convolutional neural networks (CNNs) are a common approach for this kind of image-based analysis.

### 3.4.5 Automated Decision Making

As the digital twin system matures, it could eventually move beyond providing recommendations to actually performing actions automatically, such as activating irrigation or adjusting lighting levels. Techniques like rule-based systems or reinforcement learning could enable this level of autonomy, improving both efficiency and responsiveness.

### 3.4.6 Model Management

To harness these ML applications effectively, robust model management is essential. This means not only training and evaluating different models but also tracking versions and deploying the best ones. Industry tools such as MLflow [8]or DVC [9] help manage this lifecycle, allowing the system to maintain high performance over time through continuous monitoring and improvement. Integrating a model registry supports modularity and facilitates seamless upgrades as new ML models emerge.

[8]`https://mlflow.org/`
[9]`https://dvc.org/`

# 4 Implementation

This chapter describes the practical implementation of the digital twin for plant health monitoring. It covers the hardware setup, software stack, data flow, dashboard design, and challenges encountered during the project.

## 4.1 Hardware Setup and Sensor Choices

### 4.1.1 Raspberry Pi 4 B

The monitoring system centers around a **Raspberry Pi 4 B**, selected for its computational capabilities and extensive connectivity. It is is a powerful single-board computer known for its ease of use, flexibility and low cost. The Raspberry Pi 4 is equipped with a quad-core ARM Cortex-A72 CPU, 4GB of RAM, 32GB micro-sd card of storage and multiple connectivity options such as Wi-Fi, Bluetooth, USB ports. With its 40 pins, the device readily supports interfacing with a range of sensors and actuators, including I2C, SPI, and UART devices, enabling versatile monitoring and control scenarios.

The different types of pins are:

- **Power pins**(4/40): These pins provide power to the connected devices. There are 5V and 3.3V power pins, two of each.
- **Ground pins**(8/40): These pins are used to complete the circuit and provide a return path for the current.
- **General Purpose input/output (GPIO) pins**(28/40): These pins can be configured as either input or output. In our case, they will be used to read data from the sensors and send signals to the actuators. Some of thoses pins are multiplexed with other functions, such as I2C, SPI, UART and PCM.
  - **I2C pins**(4/28): I2C (Inter-Integrated Circuit) is a communication protocol that allows multiple devices to communicate via a bus, each device having a unique address.
  - **SPI pins**(11/28): SPI (Serial Peripheral Interface) is a communication protocol that allows for high-speed data transfer between devices.
  - **UART pins**(4/28): These pins are used for serial communications and are used to send data between the Raspberry Pi and other devices.
  - **PCM pins**(4/28): PCM (Pulse Code Modulation) is used for audio data transfer.

When wiring the sensors to the Raspberry Pi, one must be careful to connect the correct pins to avoid damaging the board or the sensors and ensure proper functionality.

### 4.1.2 Sensors

To monitor the key variables defined in Section 3.1.1, the following sensors were integrated with the Raspberry Pi:

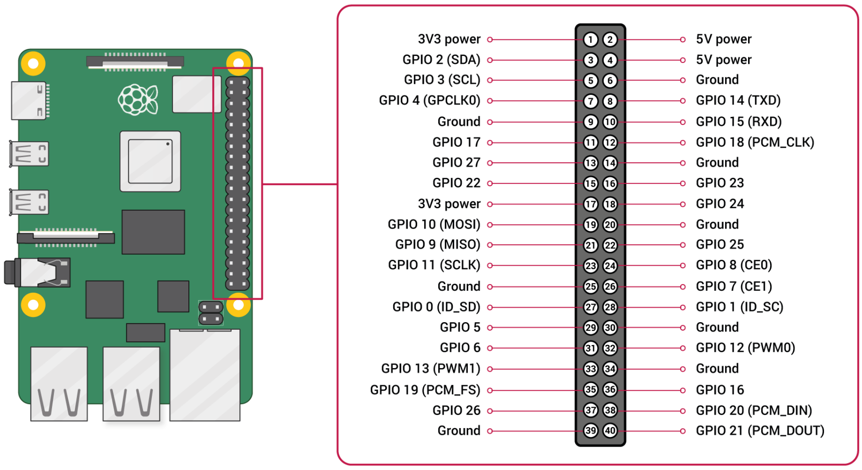


Figure 3: Pinout diagram of the Raspberry Pi 4 B. The GPIO pins are labeled with their respective functions. Source: Raspberry Pi Foundation.

- **Adafruit STEMMA Soil Sensor**: Capable of measuring both soil moisture (from 200 [very dry] up to 2000 [very wet]) and temperature (±2°C accuracy), this sensor operates over the I2C protocol. Note that this sensor cannot be calibrated to use standard moisture units such as volumetric water content (VWC). It provides only a raw capacitive measurement that must be interpreted relatively rather than absolutely [1].
- **DHT22 Sensor**: Measures environmental temperature (-40°C to 80°C) and relative humidity (0% to 100%). Known for robustness in IoT applications, it communicates via general GPIO.
- **BH1750 Sensor**:A digital ambient light sensor, with a range from 1 to 65,535 lux, communicating via I2C.
- **Camera Module**: Records images and video (up to 3280x2464 pixels and 1080p, respectively), connected via the CSI interface.

These sensors were chosen for their accuracy, ease of integration with the Python ecosystem, and cost-effectiveness.

## 4.2 Software Stack

The Python language has been chosen for the implementation of the digital twin. It is a high-level programming language known for it simplicity and is widely used in the field of data science and machine learning. Moreover, it has a large amount of libraries and frameworks that can be used for data analysis, machine learning and data visualization.

### 4.2.1 Libraries Used

This project utilizes several Python libraries that together provide a robust and flexible software stack for a digital twin implementation:


Figure 4: The sensors used in the current system. (1) Adafruit STEMMA Soil Sensor, (2) DHT22 Sensor, (3) BH1750 Sensor, (4) Camera Module.

- **Poetry**[10]: A comprehensive Python dependency management and packaging tool, Poetry enables the creation of reproducible environments and simplifies the installation and maintenance of project dependencies.
- **Adafruit CircuitPython**[11]: This suite of libraries streamlines the process of interfacing with a range of hardware components, particularly sensors. Each sensor model is supported by a dedicated module that abstracts low-level communication, providing a straightforward Python interface for data acquisition. Some examples include:
  - Adafruit CircuitPython DHT
  - Adafruit CircuitPython BH1750
  - Adafruit CircuitPython Seesaw
- **RPI.GPIO**[12]: Essential for controlling and reading the Raspberry Pi's GPIO pins, which serve as the main communication channels with most sensors and actuators.
- **Flask**[13]: A lightweight Python web framework used to build the project's web dashboard. Flask handles server-side logic, data routing, and serves front-end content, providing a user interface for live data visualization and system interaction.

  For the dashboard, JavaScript libraries such as Socket.io and Plotly.js enable real-time updates and interactive graphics respectively, delivering an intuitive user experience.
- **PySpark**[14]: Act as the Python API for Apache Spark, PySpark facilitates the processing of large sensor datasets and supports key components for streaming analytics

[10] https://python-poetry.org/
[11] https://docs.circuitpython.org/en/latest/docs/library/index.html
[12] https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/
[13] https://flask.palletsprojects.com/en/stable/
[14] https://spark.apache.org/docs/latest/api/python/index.html


Figure 5: Current environment of the plant. Raspberry Pi and sensors are placed closed to the plant.

(Spark Structured Streaming), SQL queries, and scalable machine learning through Spark MLlib. These will be usefull to preprocess the data from the sensors and to build machine learning models like depicted in the Section 3.4

- **Kafka**: Provides a resilient and scalable backbone for real-time event streaming, acting as a bridge between data producers (sensors) and various data consumers (ai service, storage, web application).
- **InfluxDB**: Serves as the project's time-series database, optimized for efficient storage, retrieval, and aggregation of continuous sensor readings.

#### 4.2.2 Application Architecture

The application architecture is based on a microservices approach (Figure 2). This architecture breaks the system down into distinct, loosely coupled services that communicate through **Representational State Transfer** (REST) API (a stateless communication interface using HTTP methods such as POST, GET, ...) and Kafka, enhancing modularity and scalability.

The main components of the architecture are:

- **Data Collector Module** (`dt/main.py`): This service is responsible for interfacing directly with the plant's sensors, collecting raw data in real time. It uses the Adafruit CircuitPython and RPI.GPIO libraries to read sensor values. Once acquired, data is streamed to the rest of the system via Kafka.

- **Data Preprocessing Module** (`dt/data/preprocess/main.py`): This module transforms raw sensor readings into clean, structured data suitable for subsequent analysis and machine learning. It is built using PySpark to efficiently handle streaming data and transmits processed results back to Kafka.

- **Data Storage Module** (`dt/data/database/app.py`): Responsible for persistent storage, this service ingests sensor readings and organizes it within InfluxDB, a time-series database tuned for high-performance read/write operations. It also exposes a REST API, allowing other services to query current and historical plant data.

- **AI Service Module** (`dt/ai/app.py`): This component handles all machine learning operations, including running and simulating predictive models. Equipped with a model registry, this service supports the training, evaluation, storage, and deployment of various models, and offers a REST API for accessing predictions and simulation results.

- **Flask Web Application module** (`dt/webapp/app.py`): This web application displays live and historical sensor readings alongside the real-time health status of the plant. It also interacts with the AI Service to retrieve analysis, alerts, and recommendations for the end user.

Each module is implemented using object-oriented programming principles. This design encourages code reuse, abstraction, and ease of maintenance, making it straightforward to modify or extend individual components as the system evolves.

## 4.3 Data Flow Example

The data flow is as follows (see Figure 6):

- The data collector module continuously polls the connected sensors, gathering raw measurements. This incoming data is immediately forwarded to Kafka, ensuring prompt delivery and decoupling of system components.

- Once received, the data processing module cleans and preprocesses the raw measurements, removing noise and filtering outliers. The processed data is then published to a dedicated Kafka topic, making it available to consumers.

- Several system modules are subscribed to this Kafka topic and simultaneously receive the processed data:
  - The data storage module stores the data in InfluxDB.
  - The AI service module can use the incoming data to update and train machine learning models, as well as trigger alerts if abnormal patterns are detected.
  - The Flask web application retrieves the latest data to display real-time updates on the dashboard and reflect the current status of the plant.

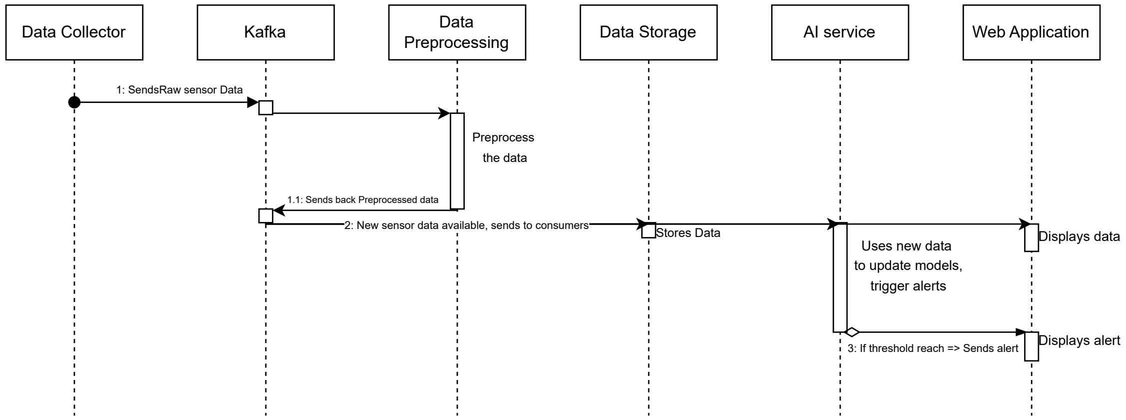


Figure 6: Sequence Diagram demonstrating the flow of the sensor readings in the system

## 4.4 Dashboard Design

The web-based dashboard serves as the main interface for users to observe and interact with the digital twin system. Although the dashboard is still under active development and not fully finalized, its core layout is organized into several sections:

- **Plant Digital Twin Overview**:This section provides a concise summary of the current status of the plant, aggregating the most recent sensor readings. It also displays recent system alerts, helping users quickly identify issues or actions that may be required. Planned future enhancements include a graphical representation of the plant's state.
- **Parameters Controls**: Here, users can interactively adjust system parameters or trigger simulation scenarios. This area is designed to give the user an active role in managing or testing the plant environment.
- **Real-Time Monitoring**: This section displays live data streams from the plant's sensors, presented through interactive graphs and charts built with Plotly.js. Users can choose different time ranges to explore historical trends or zoom in on specific periods of interest, making it easy to track changes over time.

This dashboard aims to deliver a comprehensive and intuitive platform for both real-time and historical analysis, supporting informed decision-making and seamless interaction with the digital twin system.

## 4.5 Challenges Encountered

Several challenges arose during the implementation of the system, each providing valuable learning experiences for future development:

- **Sensor Calibration**: The soil moisture sensor did not deliver the expected accuracy out of the box. Instead of measuring across the advertised range from 200 to 2000, it typically produced values within a narrower range, from about 300 to 1015. Furthermore, readings could sometimes fluctuate even when the soil conditions remained unchanged. To address this, careful calibration was required: baseline values

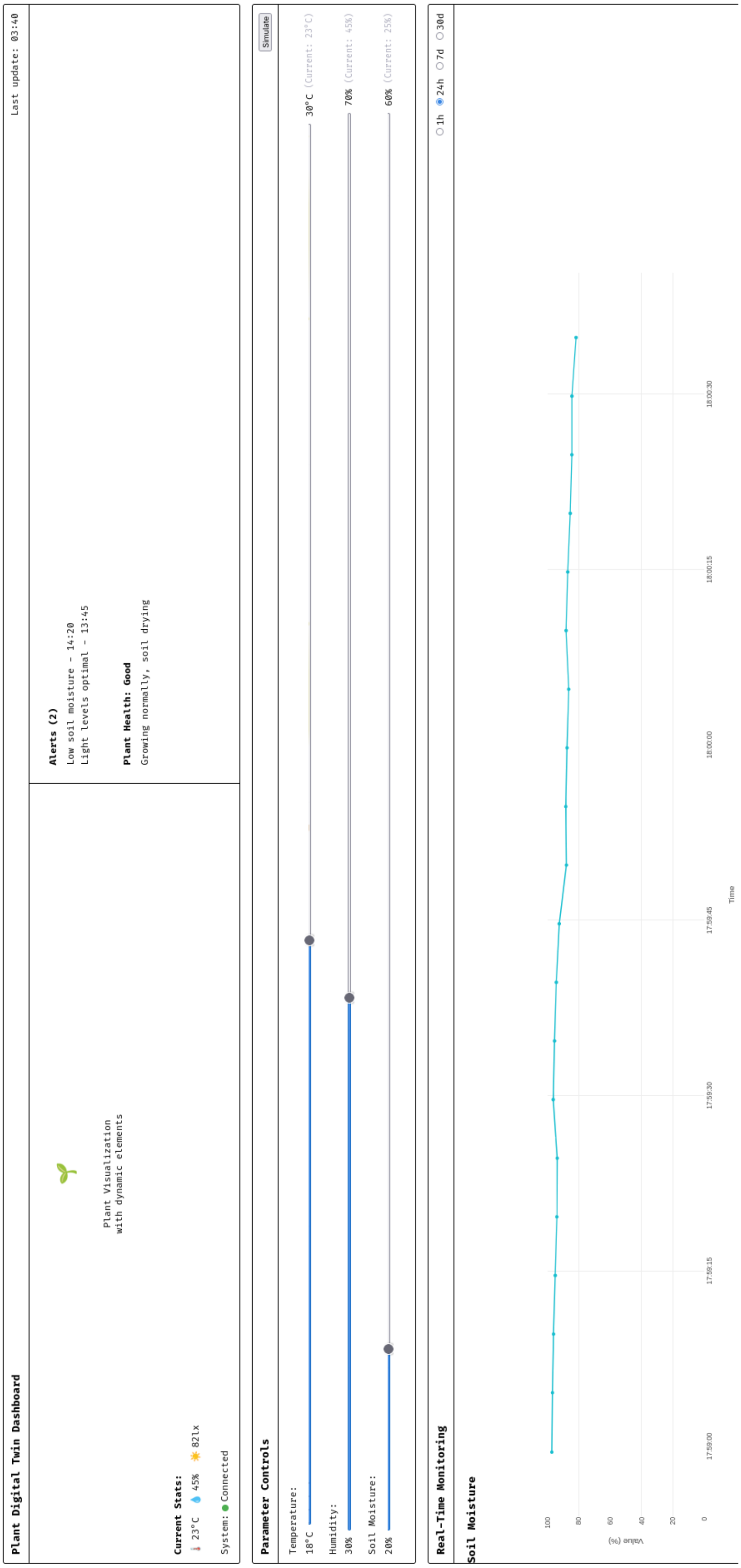


Figure 7: The dashboard of the digital twin. It displays the current status of the plant and sensors readings.

were obtained by measuring soil that was completely dry and then completely saturated. This process helped ensure that the sensor readings accurately reflected actual soil moisture levels.

- **Sensor Malfunction**: Hardware reliability presented its own obstacles. The DHT22 environmental sensor failed immediately upon connection to the Raspberry Pi, likely due to a fault or compatibility issue. Similarly, the STEMMA soil sensor, which initially operated correctly, became undetectable by the Raspberry Pi after only a day of use. Both faulty sensors had to be replaced to restore system functionality, highlighting the importance of robust hardware selection and testing in IoT projects.

- **Spark Structured Streaming**: While integrating real-time analytics with Spark Structured Streaming, implementing algorithms like Recursive Least Squares ( a simple online learning algorithm) has been challenging. The available documentation was limited and often unclear, requiring significant time and effort to fully understand the framework and develop a working solution.

- **Migrating from a mid-Scale to a large-scale system**: The project initially relied on MQTT for sensor communication and SQLite for data storage, both suitable choices for a mid-scale prototype. However, as the need for scalability became evident, the system was migrated to Kafka for messaging and InfluxDB for time-series data storage. This transition was necessary for future growth but came with a learning curve, as adopting new technologies introduced additional complexity and temporarily slowed development progress.

- **AI Service**: The AI service module, including features such as a model registry and automated model training and deployment, remains under development. Defining a robust workflow for managing, evaluating, and integrating machine learning models is an ongoing challenge that will be addressed in future iterations of the system.

## 4.6 Current State of the system

As of May 2025, the development of the Digital Twin platform for plant health monitoring has reached a stage where the core hardware and software components are operational:

- **Hardware Integration**: The foundational hardware setup is complete and fully functional. The Raspberry Pi is successfully interfaced with key environmental sensors: the DHT22 (temperature and humidity), BH1750 (light intensity), and a soil moisture sensor. All of these sensors are reliably collecting data in real time.

- **Data Acquisition and Messaging**: The Data Collector module is fully implemented and operates as expected. It collects sensor readings and serves as the interface between the physical sensors and the system. The Data Collector is capable of publishing sensor readings to Kafka, enabling real-time data streaming to consumers.

- **Data Storage**: The Data Storage module, along with its REST API, is operational. It can both ingest and store real-time sensor data and provides endpoints that allow retrieval of historical data. Users can request data either for specific sensor types or define a custom time range for their queries.

- **Web Application**: The web application is currently live and functional. It visualizes both real-time and historical sensor data, allows users to interactively explore trends, and displays alerts when defined thresholds are exceeded.
- **Ongoing Development**: The Data Preprocessing module and the AI Service (which will deliver advanced analytics and predictive insights) are under active development. An initial implementation of online learning, using the Recursive Least Squares (RLS) algorithm, has been started, but these components are not yet fully integrated into the live system.

In its current state, the system successfully demonstrates seamless sensor-to-dashboard data flow, as well as real-time and historical visualizations. Advanced analysis and decision-support features are planned for future releases as the preprocessing and AI modules are finalized.

# 5 Future work

The ultimate objective of this work is to create a scalable, robust digital twin for plant health monitoring. This chapter outlines the next steps planned for the continuation of the project in the coming year, aimed at transforming the current prototype into a digital twin system.

## 5.1 Data Preprocessing Module

Although the implementation of the data preprocessing module was postponed due to the recent transition to a large-scale system architecture, its integration remains a high priority. This module will be responsible for handling sensor anomalies, such as missing values or inconsistent readings. Incorporating robust preprocessing logic will greatly enhance data reliability, which is crucial for analytics and decision-making.

## 5.2 Advanced Analytics/ML

While basic functionality such as Recursive Least Squares (RLS) estimation has been established, primarily to gain familiarity with the PySpark ecosystem, more sophisticated machine learning models will be developed next. These will fulfill the goals outlined in Section 3.4, including predictive analytics and plant state classification.

The introduction of a model registry is also planned, allowing systematic tracking, training, deployment, and monitoring of machine learning models. An initial in-house solution will help define the requirements, after which the application of established tools such as MLflow will be explored to assess their suitability for the project.

**Planned Analytics Tasks for the Coming Year:** To ensure a structured evolution of the system, the following analytics tasks are planned for the coming year:

- **Predictions:** Implement simple linear regression models to forecast short-term trends (e.g. the next day's soil moisture or temperature from the pas week's data). These predictions can be compared directly with actual sensor measurements to evaluate performance and suggest simple, actionable recommendations (e.g. "consider irrigating tomorrow").
- **Health State Classification:** Start with rule-based classification before progressing towards basic supervised learning models using aggregated features. The classifier will provide a simple 3 state health indicator (e.g. healthy - stressed - unhealthy)

## 5.3 New DT Environment

Currently, the testing environment is minimal (as shown in 5), featuring only basic sensors. Future enhancements could introduce additional sensor types,such as NPK (nitrogen, phosphorus, potassium), $CO_2$, and pH sensors, to gain a more global understanding of plant health.

To fully realize the digital twin concept, the system will be upgraded with actuators, including devices for watering, lighting, and ventilation control, to allow for automated interventions. This will establish a closed-loop system, where the digital twin continuously

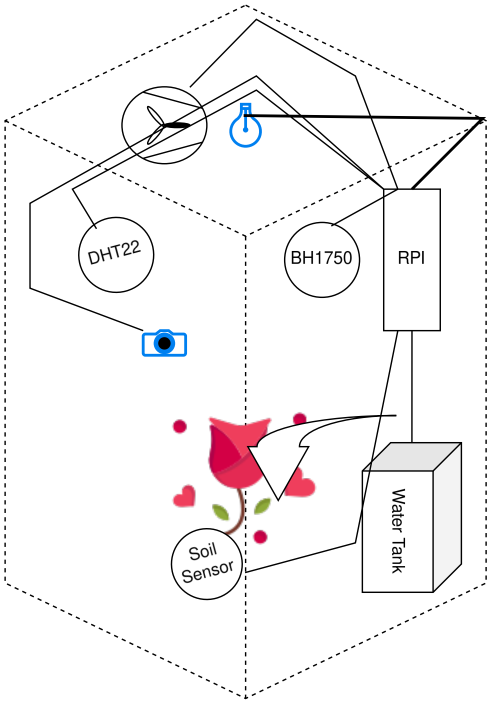


Figure 8: Schema of the final Digital Twin environment. It contains all the sensors previously mentioned and a fan, water pump/tank and light in a closed environment. The diagram has been created using draw.io and exported as SVG.

monitors, analyzes, and regulates the physical environment to promote optimal plant health.

The Figure 8 illustrates what the final environment could look like.

## 5.4 Better UI

The current user interface (UI) provides only the essential information required for monitoring. Planned improvements include enhanced data visualizations, more interactive controls (such as actuating system hardware and adjusting analytical model parameters), and a clean and modern look to facilitate user interaction.

## 5.5 Deployment

Upon achieving full system functionality, the next step will be real-world deployment to evaluate performance, reliability, and scalability under realistic conditions. Potential deployment scenarios include cloud-based hosting to enable remote access, monitoring, and management of the digital twin system.

# 6 Conclusion

This work has established a solid foundation for the development of a modular and scalable Digital Twin system for plant health monitoring. Using sensors connected to a Raspberry Pi 4 B, the system can collect and manage a range of environmental and physiological data from a plant. By leveraging the Python ecosystem, including libraries and tools such as CircuitPython, RPI.GPIO, PySpark, Kafka, InfluxDB, and Flask, the project achieves flexible sensor integration and facilitates future extensibility.

A major outcome of this work is the implementation of a real-time data pipeline capable of collecting, processing, storing, and visualizing data seamlessly. This pipeline, composed of interoperating modules, ensures a continuous and reliable data flow from the physical environment to the database and user interface.

While the current system is intentionally minimal, focusing on essential sensors and fundamental components, it provides a robust platform for further development. Immediate next steps include the integration of a Data Preprocessing module and an AI service for advanced data analysis and decision-making, the addition of new sensors and actuators to broaden environmental control, and enhancements to the user interface to improve user experience and data comprehension.

In summary, this work builds a solid and adaptable base for creating digital twins used in plant health monitoring. The progress made so far sets the stage for future improvements, bringing us closer to a more advanced, automatic, and easy-to-use system for monitoring plant health.

# Author's Note

Parts of the textual reformulation and formalization in this work were supported by the use of advanced language models (AI chatbots), specifically to refine the clarity, academic tone, and readability of technical sections. All research design, data analysis, and conclusions remain the author's original work.

# A Installation and Deployment Guide

This annex provides a comprehensive, step-by-step guide for setting up the software and system dependencies required to deploy the Digital Twin plant monitoring system. The instructions below assume a fresh installation of Raspberry Pi OS (or compatible Debian-based system) and are suitable for Raspberry Pi 4B devices. Adaptations for Ubuntu or other similar distributions are also possible.

## A.1 Requirements

Before beginning, please ensure you have the following:

- A Raspberry Pi 4B running Raspberry Pi OS (2022 release or newer)
- Python 3.11 or newer
- Bash shell access
- Reliable internet connection
- Java 17 (required for Apache Spark and Kafka)

## A.2 Step-by-Step Installation Instructions

### A.2.1 System Preparation

First, make sure your operating system and installed packages are up to date. Run the following commands in your terminal:

```
sudo apt update && sudo apt upgrade
sudo apt install python3 python3-pip git curl build-essential
```

### A.2.2 Python Environment and Poetry Setup

- **Poetry** is a tool for managing Python project dependencies and virtual environments. To install Poetry, run:

  ```
  curl -sSL https://install.python-poetry.org | python3
    ↪ -
  # Add Poetry to your PATH if necessary
  export PATH="$HOME/.local/bin:$PATH"
  ```

- Next, clone the Digital Twin project from GitHub `https://github.com/JustRayCB/digital-twin-master-thesis`:

  ```
  git clone https://github.com/JustRayCB/digital-twin-
    ↪ master-thesis digital-twin
  cd digital-twin
  ```

- Install the required Python packages using Poetry:

```
# This command will set up a venv and install all
    dependencies.
poetry install
# To activate the environment:
eval $(poetry env activate)
```

### A.2.3 Kafka and InfluxDB Installation

**Kafka** and **InfluxDB** are key services for data streaming and time-series storage respectively. For convenience, dedicated setup scripts are provided.

**Kafka:**

- To install and set up Kafka, run:

  ```
  bash setup_kafka.sh
  ```

- This script installs the latest version of Kafka (as of May 2025), configures it for the system, and ensures the service will start up automatically. (*Note: Kafka needs Java 17, so ensure Java is installed before running the script.*)
- After installation, create the necessary Kafka topics using the provided management script:

  ```
  python kafka_manager.py setup
  ```

- The `kafka_manager.py` script allows you to easily add or delete topics, adjust partitions, and modify retention or cleanup policies.

**InfluxDB:**

- Install and configure InfluxDB by running:

  ```
  bash setup_influxdb.sh
  ```

- Once InfluxDB is running, you can access its web dashboard at `http://localhost:8086` to visualize and inspect your time-series data.

## A.3 Running the System

To start the Digital Twin system, launch each module as needed. Since the system is modular, you can run different parts (such as the database, the web dashboard, and data collection modules) on separate machines if desired. Simply adjust the relevant endpoints and connection details in your configuration file located at `dt/utils/config.py`.

The provided **Makefile** automates common tasks:

- To start the Data Collector: `make main`[15]

[15]Note that this module needs to run on the Raspberry Pi to access the pins and send the sensor readings.

- To run the Web Application: `make web`
  Access the dashboard in your web browser at `http://localhost:5000`

- To launch the Data Storage module: `make db`. Access the Influx dashboard in your web browses at `http://localhost:8086`, when accessing it for the first time, it will ask you to configure. Create a username, organization and bucket that will be used in the system, e.g username: digital-twin, pswd: digital-twin, org: dt-ulb and bucket: plant-health-monitoring. (*Note: Remember that the generated token will be used by the system to communicate with the database*)

# B System demo

To illustrate the current capabilities of the Digital Twin platform, the following scenario demonstrates how one can observe and interact with the system in real time:

**Demo Setup** :

- The Raspberry Pi and all environmental sensors (temperature, humidity, light intensity, and soil moisture) are physically connected to the Raspberry Pi. The platform is running, streaming data continuously.

- The web dashboard is accessed via a standard browser

**Demo Steps**

1. **Sensor Data Acquisition**: The system begins by collecting real-time sensor readings from all connected sensors.

2. **Dashboard Visualization**: On the dashboard, live values for temperature, humidity, light intensity, and soil moisture are displayed, updating every few seconds.

3. **Exploring Data**: Historical data can be displayed by selecting a custom time window. For example, a user can review how soil moisture has evolved over the past 24 hours.

# References


[1] Lady ada. *Adafruit STEMMA Soil Sensor - I2C Capacitive Moisture Sensor*. URL: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-stemma-soil-sensor-i2c-capacitive-moisture-sensor.pdf.

[2] Kriti Agarwal et al. *Enhancing IoT based Plant Health Monitoring through Advanced Human Plant Interaction using Large Language Models and Mobile Applications*. 2025. arXiv: 2409.15910 [cs.AI]. URL: https://arxiv.org/abs/2409.15910.

[3] Rafael Gomes Alves et al. "A digital twin for smart farming". en. In: *2019 IEEE Global Humanitarian Technology Conference (GHTC)*. Seattle, WA, USA: IEEE, Oct. 2019, pp. 1–4. ISBN: 978-1-7281-1780-5. DOI: 10.1109/GHTC46095.2019.9033075. URL: https://ieeexplore.ieee.org/document/9033075/ (visited on 02/22/2025).

[4] Pelin Angin et al. "AgriLoRa: A Digital Twin Framework for Smart Agriculture". In: *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* 11 (2020), pp. 77–96. URL: https://api.semanticscholar.org/CorpusID:231858660.

[5] Natasja Ariesen-Verschuur, Cor Verdouw, and Bedir Tekinerdogan. "Digital Twins in greenhouse horticulture: A review". In: *Computers and Electronics in Agriculture* 199 (2022), p. 107183. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2022.107183. URL: https://www.sciencedirect.com/science/article/pii/S0168169922005002.

[6] Veronique Beckers et al. "The impact of urbanization on agricultural dynamics: a case study in Belgium". In: *Journal of Land Use Science* 15 ().

[7] Kaikang Chen et al. "Digital Twins in Plant Factory: A Five-Dimensional Modeling Method for Plant Factory Transplanter Digital Twins". In: *Agriculture* 13.7 (2023). ISSN: 2077-0472. DOI: 10.3390/agriculture13071336. URL: https://www.mdpi.com/2077-0472/13/7/1336.

[8] Ion Ciobotari et al. "LiDAR data acquisition and processing for ecology applications". In: *arXiv preprint arXiv:2401.05891* (2024).

[9] Hao Feng et al. "Introduction to Digital Twin Engineering". In: *2021 Annual Modeling and Simulation Conference (ANNSIM)*. 2021, pp. 1–12. DOI: 10.23919/ANNSIM52504.2021.9552135.

[10] Mohd Javaid et al. "Enhancing smart farming through the applications of Agriculture 4.0 technologies". In: *International Journal of Intelligent Networks* 3 (2022), pp. 150–164. ISSN: 2666-6030. DOI: https://doi.org/10.1016/j.ijin.2022.09.004. URL: https://www.sciencedirect.com/science/article/pii/S2666603022000173.

[11] Eduard Kamburjan et al. "GreenhouseDT: An Exemplar for Digital Twins". In: *2024 IEEE/ACM 19th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 2024, pp. 175–181. DOI: 10.1145/3643915.3644108.

[12] Evangelia Katsoulakis et al. "Digital twins for health: a scoping review". In: *npj Digital Medicine* 7.1 (2024), p. 77. ISSN: 2398-6352. DOI: 10.1038/s41746-024-01073-0. URL: https://doi.org/10.1038/s41746-024-01073-0.

[13] Rui Neves Madeira et al. "Towards Digital Twins for Multi-Sensor Land and Plant Monitoring". en. In: *Procedia Computer Science* 210 (2022), pp. 45–52. ISSN: 18770509. DOI: 10.1016/j.procs.2022.10.118. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050922015745 (visited on 02/22/2025).

[14] Christos Mitsanis, William Hurst, and Bedir Tekinerdogan. "A 3D functional plant modelling framework for agricultural digital twins". en. In: *Computers and Electronics in Agriculture* 218 (Mar. 2024), p. 108733. ISSN: 01681699. DOI: 10.1016/j.compag.2024.108733. URL: https://linkinghub.elsevier.com/retrieve/pii/S0168169924001248 (visited on 02/22/2025).

[15] Paula F. Murakami et al. *An instructional guide for leaf color analysis using digital imaging software*. 2005. DOI: 10.2737/ne-gtr-327. URL: http://dx.doi.org/10.2737/NE-GTR-327.

[16] U.S Government Accountability Office. *Science & Tech Spotlight: Digital Twins—Virtual Models of People and Objects*. Accessed: 2025-03-31. 2023. URL: https://www.gao.gov/products/gao-23-106453.

[17] Ariel Ortiz-Bobea et al. "Anthropogenic climate change has slowed global agricultural productivity growth". In: *Nature Climate Change* 11.4 (2021), pp. 306–312. ISSN: 1758-6798. DOI: 10.1038/s41558-021-01000-1. URL: https://doi.org/10.1038/s41558-021-01000-1.

[18] Karim Panjvani et al. "LiDARPheno: A Low-Cost LiDAR-based 3D Scanning System for Plant Morphological Trait Characterization". PhD thesis. University of Saskatchewan, 2018.

[19] "Pathogens, precipitation and produce prices". In: *Nature Climate Change* 11.8 (2021), pp. 635–635. ISSN: 1758-6798. DOI: 10.1038/s41558-021-01124-4. URL: https://doi.org/10.1038/s41558-021-01124-4.

[20] Plagron. *How to see if your plant is healthy*. URL: https://plagron.com/en/hobby/grow-topics/how-to-see-if-your-plant-is-healthy.

[21] PlantIn. *How to identify a healthy plant*. URL: https://myplantin.com/blog/how-to-identify-a-healthy-plant.

[22] Warren Purcell and Thomas Neubauer. "Digital Twins in Agriculture: A State-of-the-art review". en. In: *Smart Agricultural Technology* 3 (Feb. 2023), p. 100094. ISSN: 27723755. DOI: 10.1016/j.atech.2022.100094. URL: https://linkinghub.elsevier.com/retrieve/pii/S2772375522000594 (visited on 02/22/2025).

[23] Warren Purcell and Thomas Neubauer. "Digital Twins in Agriculture: A State-of-the-art review". In: *Smart Agricultural Technology* 3 (2023), p. 100094. ISSN: 2772-3755. DOI: https://doi.org/10.1016/j.atech.2022.100094. URL: https://www.sciencedirect.com/science/article/pii/S2772375522000594.

[24] Rashmi R et al. "Exploring Digital Twins for Plant Growth Monitoring". In: *2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*. 2023, pp. 1–6. DOI: 10.1109/CSITSS60515.2023.10334087.

[25] Petr Skobelev et al. “Multi-agent approach for developing a digital twin of wheat”. en. In: *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*. Bologna, Italy: IEEE, Sept. 2020, pp. 268–273. ISBN: 978-1-7281-6997-2. DOI: 10.1109/SMARTCOMP50058.2020.00062. URL: https://ieeexplore.ieee.org/document/9239641/ (visited on 02/22/2025).

[26] Mohsen Soori, Behrooz Arezoo, and Roza Dastres. “Digital twin for smart manufacturing, A review”. In: *Sustainable Manufacturing and Service Economics* 2 (2023), p. 100017. ISSN: 2667-3444. DOI: https://doi.org/10.1016/j.smse.2023.100017. URL: https://www.sciencedirect.com/science/article/pii/S2667344423000099.

[27] D Tilman and C Lehman. “Human-caused environmental change: impacts on plant diversity and evolution”. en. In: *Proc Natl Acad Sci U S A* 98.10 (May 2001), pp. 5433–5440.

[28] Cor Verdouw et al. “Digital twins in smart farming”. In: *Agricultural Systems* 189 (2021), p. 103046. ISSN: 0308-521X. DOI: https://doi.org/10.1016/j.agsy.2020.103046. URL: https://www.sciencedirect.com/science/article/pii/S0308521X20309070.

[29] Amy Wrobleski et al. “The impact of climate change on endangered plants and lichen”. In: *PLOS Climate* 2.7 (July 2023), pp. 1–20. DOI: 10.1371/journal.pclm.0000225. URL: https://doi.org/10.1371/journal.pclm.0000225.