PlantMo: Plant Monitoring

System Specification Document (SSD)

Cartago · Tonet · Toso · Zambonini

Document Status. The following document presents a preliminary and evolving version of the system specification. The described requirements, design choices, and architectural decisions may be subject to modification as the project advances. Accordingly, this document should be regarded as a working draft rather than a finalized specification.

Contents

1 Problem Definition

Effective plant cultivation requires consistent monitoring of environmental conditions to maintain optimal growth and detect potential issues early. Manual observation and periodic measurements provide only snapshots of environmental data, lacking the continuity necessary for comprehensive analysis.

Without continuous and reliable data collection, gradual environmental changes and anomalous patterns often go undetected until they have already impacted plant health. This reactive approach limits the ability to intervene proactively and maintain ideal growing conditions. This project aims to develop an automated environmental monitoring system that collects, processes, and analyzes sensor data.

The system will provide insights into environmental conditions and enable early detection of anomalous behavior, facilitating timely intervention and optimized plant care.

1.1 Business Problem

The goal of this project is to design a smart plant monitoring system capable of collecting, aggregating and analyzing environmental data using sensors and machine learning techniques. The system aims to support **data-driven decision making**, detect drifts, and provide real-time visualization of plant conditions.

1.2 ML Formulation

From a Machine Learning perspective, the problem can be formulated as a **time-series analysis and monitoring**.
The Machine Learning model will consider the following tasks:

- Statistical aggregation of sensor data;
- Detection of data drift and anomalous trends;
- Predictive modeling of plant watering conditions.

The Machine Learning model will consider the following input features (sensors):

- Light intensity;
- Temperature;
- Humidity;
- Soil moisture.

1.3 KPIs

The global effectiveness of the system will be evaluated using the following KPIs.

- **Data Availability**: number of successfully stored sensor data;
- **Data Latency**: delay between data acquisition and cloud availability;
- **System Reliability**: stability of sensor readings over time;

- **Drift Detection Accuracy**: ability to identify and plot significant changes in data distributions;
- **Dashboard Responsiveness**: load and refresh time of the visualization interface.

2 Data Specification

2.1 Data Sources

Data are collected from **four environmental sensors** attached to a plant:

- Light Sensor;
- Temperature Sensor;
- Humidity Sensor;
- Soil Moisture Sensor.

Each sensor is connected to an Arduino microcontroller interface, which acts as the data acquisition unit. Table 1 summarizes the measurement ranges and units associated with each sensor.

Table 1: Sensor measurement ranges and scales

Sensor	Measurement Range	Unit / Scale
Humidity	0 – 100	Percentage (%)
Temperature	0 – 50	Degrees Celsius (°C)
Light	0 – 1023	ADC scale (0 = dark, 1023 = max light)
Soil Moisture	0 – 1023	ADC scale (0 = dry soil, 1023 = fully wet)

Light intensity and soil moisture readings are expressed on the native analog-to-digital converter (ADC) scale of the sensors and should therefore be interpreted as **qualitative indicators** of environmental conditions rather than absolute physical measurements. For soil moisture, a reference threshold value of approximately 840 is expected under optimal watering conditions.

2.2 Data Flow

The data flow describes the end-to-end pipeline through which sensor measurements are acquired, processed, stored, and made available for analysis and visualization. Data are progressively transformed through a sequence of processing stages designed to ensure reliability, temporal consistency, and accessibility. This structured flow enables continuous data acquisition, aggregation over fixed time windows, and centralized cloud logging, while also supporting secure access and real-time monitoring through dedicated visualization tools

1. Sensors sample raw signals every 10 seconds, which are summarized through a 1-minute temporal window;
2. The Arduino board computes local statistics and transmits feature-based data every 5 minutes;

3. Processed sensor data are logged to the cloud using **Weights and Biases** (W&B);
4. Authorized clients access the data through secure API keys;
5. Data are visualized and continuously monitored via a **Streamlit Dashboard**.

2.3 Quality

Data quality is essential to ensure that the collected measurements accurately represent the environmental conditions of the plant. Basic checks are applied to identify potential issues that may affect the reliability of the data.
To ensure data reliability, the following aspects are considered:

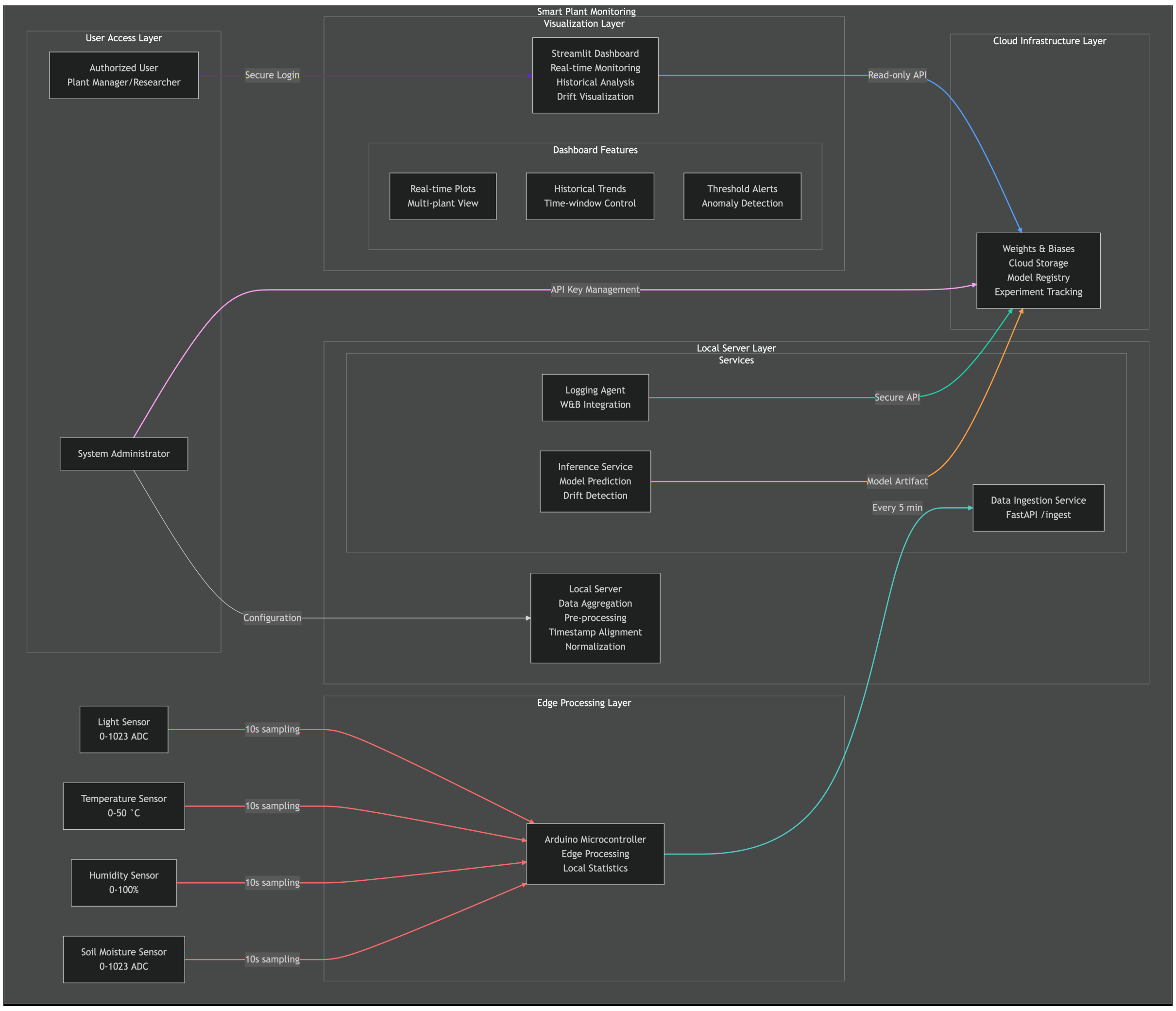
- Handling missing or corrupted sensor readings;
- Noise reduction through temporal aggregation;
- Detection of out-of-range values;
- Timestamp consistency across sensors.

2.4 Pre-processing

Preprocessing steps include:

- Timestamp alignment;
- Weighted averaging over fixed time windows;
- Normalization of sensor values;
- Formatting for cloud logging and visualization

2.5 UML Architecture



Smart Plant Monitoring
Visualization Layer
User Access Layer
Authorized User Plant Manager/Researcher
Secure Login
Streamlit Dashboard Real-time Monitoring Historical Analysis Drift Visualization
Read-only API
Cloud Infrastructure Layer
Dashboard Features
Real-time Plots Multi-plant View
Historical Trends Time-window Control
Threshold Alerts Anomaly Detection
Weights & Biases Cloud Storage Model Registry Experiment Tracking
API Key Management
Local Server Layer
Services
Logging Agent W&B Integration
Secure API
System Administrator
Inference Service Model Prediction Drift Detection
Model Artifact
Every 5 min
Data Ingestion Service FastAPI /ingest
Configuration
Local Server Data Aggregation Pre-processing Timestamp Alignment Normalization
Edge Processing Layer
Light Sensor 0-1023 ADC
10s sampling
Temperature Sensor 0-50 °C
10s sampling
Humidity Sensor 0-100%
10s sampling
Soil Moisture Sensor 0-1023 ADC
10s sampling
Arduino Microcontroller Edge Processing Local Statistics

Figure 1: Environmental sensors sample data every 10 seconds and transmit them to an Arduino-based edge layer for local processing and statistical aggregation. Preprocessed data are handled by a local server that performs normalization, inference, and drift detection before periodic ingestion into the cloud via secure APIs. Cloud services support model storage and experiment tracking, while a Streamlit dashboard enables real-time monitoring, historical analysis, anomaly detection, and multi-plant visualization for authorized users.

3 Functional Requirements

FR01	The Arduino Microcontroller collects environmental data from four sensors measuring light, temperature, humidity, and soil moisture.
FR02	Sensor measurements are sampled every 10 seconds for each sensor.
FR03	Sensor data are summarized using a 1-minute temporal window to compute statistical features.
FR04	A representative set of features is generated and transmitted every 5 minutes for each sensor.
FR05	The Arduino Microcontroller forwards collected data to a local server for processing.
FR06	The Arduino performs local processing of sensor data and transmits the resulting features to the local server.
FR07	Aggregated sensor data is uploaded to a cloud-based platform (W&B) for long-term storage.
FR08	The data pipeline integrates with an experiment tracking platform to log time-series measurements.
FR09	Access to cloud-stored data is controlled through API keys assigned to authorized users.
FR10	Authorized clients can retrieve and upload sensor data through secure interfaces.
FR11	Visualization of sensor data is provided through a web-based dashboard.
FR12	Historical sensor data can be explored through the dashboard for trend analysis.
FR13	Changes in data distributions are monitored to identify potential data drift.
FR14	Abnormal patterns or drift events are signaled to the user interface.
FR15	System operations and detected anomalies are recorded for monitoring and debugging purposes.

4 Non Functional Requirements

NFR01	The system is designed to operate continuously under normal environmental and network conditions.
NFR02	Data processing and cloud logging occur with minimal delay after acquisition.
NFR03	The system architecture supports scalability for additional sensors or monitored plants.
NFR04	Secure communication mechanisms are used for data exchange between system components.
NFR05	Stored data remains protected against unauthorized modification or loss.
NFR06	Temporary network or power failures do not permanently compromise data availability.
NFR07	Software components follow a modular structure to simplify maintenance and updates.
NFR08	The system is designed to be deployable across different operating systems without functional degradation.
NFR09	The user interface prioritizes clarity and ease of use for effective data interpretation.
NFR10	Logging and monitoring capabilities facilitate performance evaluation and troubleshooting.

5 Project Architecture

5.1 Training Phase

The training phase is designed to support an incremental and robust development of the Machine Learning model, accounting for the limited availability of real-world data during the early stages of the project.

Initially, an early demonstration is performed using a synthetic dataset that simulates realistic environmental sensor patterns. This synthetic data acts as a fallback solution, enabling the definition, testing, and debugging of the training pipeline before sufficient real sensor data are collected from the Arduino-based system. Although synthetic data do not fully capture real environmental variability, they provide a controlled baseline to validate data ingestion, preprocessing, and model behavior.

Once real data become available, a full training demo is conducted using measurements collected directly from the monitored plant. These data represent time-series observations of environmental conditions and are used to train the predictive and monitoring models under realistic operating conditions.

Given the temporal nature of the data, the dataset is split according to chronological order rather than random sampling. Specifically, a 70/30 split is applied, where the first 70% of observations are used for training and the remaining 30% are reserved for evaluation. This approach preserves temporal consistency and reflects real-world deployment scenarios, where future values must be predicted based solely on past observations.

All training experiments, including dataset versions, hyperparameters, and intermediate results, are logged and tracked using Weights and Biases (W&B) to ensure reproducibility and traceability.

5.2 Validation Phase

The validation phase aims to quantitatively assess the performance of the trained model and determine its readiness for deployment.

Model evaluation is performed using the Mean Squared Error (MSE), which measures the average squared difference between predicted and observed values over the validation set. The MSE metric is particularly suitable for continuous time-series data, as it penalizes larger prediction errors and provides a clear indication of predictive accuracy.

A predefined acceptance threshold is established for the MSE. If the model achieves an error below this threshold, it is considered valid and suitable for promotion. Models that fail to meet this criterion are discarded or retrained with updated parameters or additional data.

Validated models are automatically logged to the Weights and Biases Model Registry, where versioning is applied to track model evolution over time. This mechanism ensures that only approved model versions are made available for deployment, while maintaining a complete history of previous experiments for auditing and comparison purposes.

5.3 Deployment phase

Component / Task	Deployment Actions and Configuration
Model Promotion	The validated model is transitioned to the `Production` stage in the Weights & Biases (W&B) Model Registry. The specific versioned artifact (e.g., `model_prod:v4.2`) is retrieved via API to ensure full traceability and reproducibility of the production system.
Edge Device Finalization	The Arduino microcontroller is flashed with the frozen, validated firmware. This locks all operational parameters: sensor sampling rates (e.g., IMU at 10 Hz), temporal window length (5 seconds), feature extraction logic, and weighted aggregation formulas. A final burn-in test confirms stability.
Server Provisioning	The local server is provisioned with a dedicated Python environment containing all dependencies (scikit-learn, FastAPI, W&B SDK). The model artifact and its associated preprocessor (e.g., StandardScaler `.pkl` file) are placed in a secure, version-controlled directory.
Service Integration	Core services are launched: 1) A **Data Ingestion Service** (e.g., FastAPI endpoint at `/ingest`) listens for Arduino data. 2) The **Inference Service** loads the model and executes predictions and drift detection. 3) A **Logging Agent** streams outputs to the cloud database using encrypted credentials.
Dashboard Deployment	The Streamlit dashboard is containerized using Docker and deployed as a service. It is configured with read-only cloud database credentials via `secrets.toml` and exposed on a designated port (e.g., `:8501`) behind the network firewall.
Post-Deployment Verification	A formal smoke test executes: simulated data is sent to the ingestion endpoint; cloud logs and dashboard updates are verified for latency and correctness; and a final live connection test with the Arduino device is performed to confirm end-to-end operation.

Table 2: Deployment Phase Component Configuration and Tasks

Upon successful completion of all tasks detailed in Table 2, the system is formally designated as *Operational*. Authority is transferred from the development team to the monitoring and maintenance team, initiating the continuous monitoring phase.

5.4 Monitoring phase

The monitoring phase enables real-time observation of sensor data, system status, and environmental conditions through a dedicated dashboard. The monitoring infrastructure is designed to ensure data continuity, anomaly detection, and clear user awareness during system operation.

The system supports multiple data sources, including cloud-based logging through Weights & Biases (W&B), synthetic data for testing and demonstration, and randomly generated data for debugging and fallback scenarios. The connection status to W&B is explicitly shown in the interface, allowing users to verify whether the system is operating in online or offline mode.

The dashboard allows simultaneous monitoring of multiple plants, which can be dynamically added or removed. Each plant is equipped with a complete set of sensors (light, temperature, humidity, and soil moisture). Plant configurations and sensor metadata are retrieved via an API from JSON configuration files, ensuring modularity and scalability.

Sensor values are visualized as continuously updated time-series plots. Users can interactively select the number of historical samples displayed and adjust the visualization time window. For selected sensors, such as light intensity, current values are compared against configurable minimum and maximum thresholds. Threshold violations are highlighted through visual indicators and interpreted as potential anomalies.

To ensure efficient memory usage, the monitoring application maintains a fixed-size local buffer containing the most recent sensor readings currently displayed. The buffer is limited to 150 samples per sensor, with automatic removal of the oldest data when the limit is exceeded.

Feature	**Description**
Data Sources	W&B logging, synthetic data, random fallback data
Multi-Plant Monitoring	Dynamic management of multiple plants with full sensor sets
Visualization	Real-time time-series plots with adjustable history window
Anomaly Detection	Threshold-based monitoring with visual alerts
Configuration	API-based JSON configuration for plants and sensors
Data Buffering	Fixed buffer of 150 samples per sensor

6 Risk Analysis

Risk	**Description**
Sensor Failure	One or more sensors may malfunction, degrade over time, or provide inaccurate measurements, leading to unreliable data collection.
Data Loss	Interruptions in communication, power outages, or hardware issues may cause partial or complete loss of sensor data.
Noise in Data	Environmental disturbances or sensor limitations may introduce noise and fluctuations in the collected measurements.
Security Breach	Unauthorized access to the cloud platform or APIs may compromise data integrity and confidentiality.
Scalability Issues	The system may face performance or management challenges when integrating additional sensors or monitored plants.