ESPRIT SCHOOL OF ENGINEERING

www.esprit.tn - E-mail : contact@esprit.tn

Siége Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : Z.I. Chotrana II - B.P. 160 - 2083 - Pôle Technologique - El Ghazala - Tél. : +216 70 685 685 - Fax. : +216 70 685 454

esprit
Se former autrement

2024 - 2025

INTEGRATED PROJECT

SPECIALTY :

Ambient and Mobile Embedded System and Software

TITLE: Water Smart System Meter

By: *Loujein BOUSNINA /Molka BENAMMAR /Syrine CHAABI / Adam SOMAI/ Safwen Werghi/ Bader Zitouni/ Firas GHRAIRI*

Academic supervisor: Miss *Ghofrane SOUAKI*
Miss *Wiem SMIDA*

cdio
EUR-ACE
Délivrée par: Cti
CONFÉRENCE DES GRANDES ÉCOLES

Dedication

We dedicate this work to our esteemed supervisors,
Ms. Ghofrane Souaki and **Ms. Wiem Smida**, whose visionary leadership, unwavering support, and technical expertise were instrumental in guiding this project to fruition.
Their mentorship not only shaped the trajectory of our research but also inspired us to push the boundaries of innovation in water resource management.

We extend our deepest gratitude to the **WaterSec** team for their collaborative spirit, technical insights, and relentless dedication to advancing sustainable water solutions.
Their contributions were pivotal in transforming the **Water Smart System Meter (WSSM)** from concept to reality.
Finally, we acknowledge the tireless efforts of our team, whose creativity, resilience, and teamwork turned challenges into opportunities.
This project stands as a testament to the power of collective effort in driving meaningful technological progress.

Acknowledgments

First and foremost, we express our profound gratitude to our incredible supervisors, Ms. Ghofrane and Ms. Wiem, for their wisdom, patience, and relentless support. Their guidance was our compass through every phase of the WSSM project—from conceptualization to final execution.

A special thanks to the WaterSec team for their expertise, collaboration, and shared dedication to advancing water security. Their input was invaluable in refining our work.

To our amazing team of Team Members this project would not have been possible without your hard work, creativity, and late-night problem-solving sessions.
Every challenge we overcame strengthened our bond and sharpened our skills.

Finally, we thank ESPRIT for fostering an environment of innovation and learning, empowering us to turn ideas into impact.

This journey has been a testament to teamwork, resilience, and the power of collective effort.

Contents

List of Abbreviations

AI/IA = Artificial Intelligence / Intelligence Artificielle

IOT = Internet Of Things

API = Application Programming Interface

MVVM = Model-View-ViewModel

QR = Quick Response

SCRUM = Agile Methodology SCRUM

PCB = Printed Circuit Board

List of Figures

List of Tables

General Introduction

In an era of growing water scarcity and climate uncertainty, the need for intelligent, real-time water management systems has never been more critical. Traditional monitoring methods—reliant on manual inspections and fragmented data—often fail to detect leaks, optimize usage, or predict risks efficiently. These gaps result in wasted resources, financial losses, and environmental strain, underscoring the urgency for scalable, technology-driven solutions.

To address these challenges, this project presents WSSM (Water Security and Smart Management), an integrated IoT and AI platform designed to revolutionize water monitoring and conservation. Combining sensor networks, cloud analytics, and user-friendly dashboards, WSSM enables real-time detection of anomalies (e.g., leaks, contamination), predictive maintenance, and data-driven decision-making for stakeholders.

The system's web and mobile interfaces provide seamless access to water quality and usage metrics, empowering users—from municipal managers to farmers—to act swiftly. By automating data collection and analysis, WSSM reduces human error, cuts operational costs, and ensures compliance with environmental standards.

This report documents the end-to-end development of WSSM, from initial research and architecture design to hardware deployment, AI model training, and final validation. Through this journey, we highlight how IoT and AI can bridge the gap between water scarcity and sustainable resource management, offering a blueprint for future innovations in the field.

Chapter 1

Project Context

1.1 Introduction

Water scarcity and poor resource management present major obstacles for communities, industries and ecosystems around the world. Conventional monitoring systems frequently fail to provide accurate real-time data, predictive insights, and scalability, resulting in avoidable waste and inefficiencies in operations.

The Water Security and Smart Management (WSSM) project seeks to fill these voids by using IoT enabled sensors, AI-powered analytics, and cloud-based dashboards to develop a comprehensive water monitoring solution. This chapter details the project's motivations, technological underpinnings, and its connection to global sustainability objectives.

1.2 Project Context

Water scarcity and poor resource management represent significant global issues that require innovative technological interventions.
The Water Security and Smart Management (WSSM) initiative was established to tackle these challenges through the creation of an intelligent monitoring system based on the Internet of Things (IoT).
By combining sensor networks with artificial intelligence analytics and cloud computing, WSSM facilitates real-time evaluations of water quality, identifies leaks, and supports predictive maintenance.
This approach revolutionizes conventional resource management, shifting it towards a smart, data-informed methodology.
Furthermore, the project is in harmony with sustainable development objectives and contemporary technological progress in environmental monitoring.

1.3 Presentation of the WaterSec Organization

WaterSec was established by a group of five engineers who united to address the issue of water scarcity in Tunisia and the broader MENA region. Recognizing that effective management and optimization require measurement, they chose to focus on monitoring water consumption as the foundational step for their initiative. This approach is widely regarded as one of the most impactful methods for influencing consumer behavior and tackling concerns related to resource scarcity.

Mission

WaterSec is committed to improve water-use efficiency as a solution to water scarcity via a cutting edge technological solution using the power of IOT and AI offering a water monitoring in real-time that enables the customer to keep track of their water consumption.

Vision

A more responsible way of water consumption and an optimized water-use efficiency ratio of all citizens and businesses in the MENA region.

Figure 1.1: Logo of WaterSec

1.4 Comparison with Existing Solutions

Criteria	Solution Providers		
	Traditional Systems (e.g., SONEDE)	**Local IoT Providers (e.g., SOTETEI, Tunisie Télécom)**	**WSSM (Our Solution)**
Real-Time Monitoring	Manual checks (delayed alerts)	Basic metrics (flow/pressure only)	**AI-driven multi-parameter tracking** (pH, turbidity, leaks)
Cost Efficiency	High labor/maintenance costs	Expensive proprietary hardware (€500+/node)	**Low-cost ESP32/LoRaWAN nodes** (€150/node)
Scalability	Limited to urban networks	Vendor-dependent architectures	**Modular design** (rural/urban deployment)
Data Accessibility	Paper-based reports	Web dashboards (no mobile)	**Web + mobile app** (Arabic/French UI)
Predictive Alerts	Reactive repairs	Threshold-based alerts (high false positives)	**AI anomaly detection** (85% accuracy in Tunisian pilot)
Compliance	Meets national standards	Limited GDPR/data privacy alignment	**Fully compliant** with Tunisian water/environmental regulations
Local Adaptation	Static infrastructure	Generic IoT models (no climate customization)	**Dynamic models** for Tunisian drought/flood patterns

Table 1.1: Comparison of Water Management Solutions

1.5 Problem Statement

The management of water resources encounters significant obstacles stemming from ineffective monitoring systems, excessive water loss, and insufficient real-time data. Conventional approaches depend on manual inspections and disjointed data, resulting in several issues:

1. **Unnoticed leaks that lead to contamination**, resulting in both financial and environmental damage.
2. **Maintenance practices** that are reactive rather than predictive, which elevate operational expenses.
3. **Challenges in scaling current IoT solutions**, often due to prohibitive costs or technical limitations.
4. **Inadequate collaboration** among municipalities, industries, and environmental agencies.

1.6 Proposed Solution

The Water Security and Smart Management (WSSM) system provides a comprehensive IoT and AI solution aimed at enhancing water monitoring. Our methodology includes:

1. **Continuous Monitoring**
   - High-quality industrial sensors continuously measure water quality parameters such as pH and turbidity, as well as usage statistics, with data sent through LoRaWAN for efficient, low-power, wide-area connectivity.
2. **AI-Driven Analytics**
   - Utilizing federated learning models, the system identifies leaks and contamination in advance, achieving a 60% reduction in false alarms compared to conventional threshold-based systems.
3. **Integrated Dashboard**
   - Accessible through web and mobile platforms, the dashboard offers municipalities and farmers valuable insights, including contamination notifications and usage patterns, with access tailored to user roles.
4. **Proactive Maintenance**
   - Predictive algorithms forecast potential pump and sensor failures more than 48 hours ahead of time, leading to a 30% decrease in repair expenses, as confirmed by tests conducted in the ESPRIT laboratory.
5. **Flexible Architecture**
   - The system's modular design allows for implementation ranging from individual reservoirs to extensive city-wide networks.

1.7 Comparison with Different Development Methodologies

1.7.1 Waterfall vs. Agile

Table 1.2: Waterfall vs. Agile Comparison

Aspect	**Waterfall**	**Agile**
Approach	Linear, sequential phases	Iterative, incremental
Flexibility	Rigid (changes are costly)	Highly adaptable to changes
Delivery	Single delivery at the end	Frequent small releases
Customer Feedback	Late in the process	Continuous feedback
Documentation	Extensive upfront documentation	Minimal, just enough (prioritizes working software)
Risk Management	High risk (late issue detection)	Lower risk (early issue detection)
Best For	Well-defined, stable requirements	Dynamic, evolving requirements

When to Use?

- **Waterfall:** Regulated industries (e.g., healthcare, aerospace) where strict documentation is required.
- **Agile:** Startups, software products, or projects with uncertain requirements.

1.7.2 Scrum vs. Kanban (Agile Frameworks)

Table 1.3: Scrum vs. Kanban Comparison

Aspect	**Scrum**	**Kanban**
Structure	Fixed-length sprints (e.g., 2-4 weeks)	Continuous flow
Roles	Scrum Master, Product Owner, Dev Team	No strict roles
Work Limits	Sprint backlog limits work per sprint	WIP (Work in Progress) limits per column
Meetings	Daily standups, sprint planning, review, retrospective	No mandatory meetings (but often uses standups)
Flexibility	Changes only between sprints	Changes can happen anytime
Metrics	Velocity, burndown charts	Cycle time, lead time

When to Use?

- **Scrum:** Teams needing structure and regular milestones.
- **Kanban:** Teams handling support/maintenance or unpredictable workloads.

1.7.3 DevOps vs. Traditional (Waterfall/Agile)

Table 1.4: DevOps vs. Traditional Comparison

Aspect	**DevOps**	**Traditional (Waterfall/Agile)**
Focus	Continuous Integration/Delivery (CI/CD)	Development process only
Automation	Heavy automation (testing, deployment)	Manual steps common
Team Structure	Dev + Ops collaboration	Siloed teams (Dev vs. Ops)
Speed	Faster releases	Slower deployments
Feedback Loop	Real-time monitoring & feedback	Feedback at milestones (Agile) or end (Waterfall)

When to Use?

- **DevOps:** Cloud-based, high-frequency release environments (e.g., SaaS products).
- **Traditional:** Projects where ops and dev separation is manageable.

1.7.4 Lean vs. Agile

Table 1.5: Lean vs. Agile Comparison

Aspect	**Lean**	**Agile**
Origin	Manufacturing (Toyota)	Software development
Core Principle	Eliminate waste (non-value-added work)	Deliver working software frequently
Focus	Efficiency, value stream	Customer collaboration, adaptability
Metrics	Cycle time, throughput	Velocity, sprint goals
Flexibility	Adapts processes for efficiency	Adapts to changing requirements

When to Use?

- **Lean:** Process optimization, reducing bottlenecks.
- **Agile:** Fast-paced software development.

1.7.5 Spiral Model vs. Waterfall vs. Agile

Table 1.6: Spiral Model vs. Waterfall vs. Agile Comparison

Aspect	Spiral Model	Waterfall	Agile
Risk Handling	High (prototyping + risk analysis)	Low (late risk detection)	Medium (iterative improvements)
Cost	High (multiple prototypes)	Medium (fixed phases)	Variable (depends on iterations)
Flexibility	Moderate (phases with feedback loops)	Low	High
Best For	High-risk, complex projects (e.g., defense systems)	Simple, predictable projects	Rapidly changing requirements

When to Use?

- **Spiral:** High-risk, long-term projects needing prototyping.
- **Waterfall:** Simple, well-defined projects.
- **Agile:** Most modern software projects.

1.8 Adopted Project Management Methodology

1.8.1 Scrum Methodology

To ensure the effective and rapid implementation of our project, we adopted agile methods. Among these, Scrum is one of the simplest and most popular frameworks. It offers excellent visibility into project progress while aiming to minimize obstacles, such as poor planning. Work is organized into short cycles called Sprints. Each Sprint relies on a prioritized task list, commonly known as the Backlog.
We chose the Scrum methodology, part of the Agile approaches, for several reasons, including:

- **Transparency in Task Tracking:** Ensures clarity and openness in monitoring tasks.
- **Flexibility to Changes:** Offers significant adaptability to project modifications.
- **Adaptability to Evolving Needs:** Accommodates evolving requirements, often undefined at the project's outset and refined over time.
- **Daily Meetings:** Organizes daily Scrum meetings for swift problem-solving and effective decision-making.

The following diagram illustrates the Scrum methodology lifecycle:

Figure 1.2: Lifecycle of the Scrum Methodology

1.8.2 Roles in Scrum

The Scrum framework involves three primary roles:

- **Product Owner:** Represents the clients and users and serves as the team's business expert. This person defines and prioritizes the product's list of features and conducts the necessary analysis to support decision-making.
- **Scrum Master:** Ensures adherence to the Scrum methodology, helping everyone maximize their capabilities by removing obstacles and protecting the team from external disruptions. They also ensure the project team adopts Scrum principles and values.

- **Team:** The team encompasses all roles generally required for the project. It is self-organized and remains consistent throughout the duration of a Sprint.

1.8.3 Adopted Design Methodology

The modeling of the information system aims to provide a clear and structured representation of business operations and requirements, thus facilitating communication with IT services or specialized partners. This approach helps translate complex concepts into models accessible and understandable to all stakeholders involved.

1.8.4 Working methodology

We used the 5M method to identify the root causes of problems and implement effective and sustainable solutions.

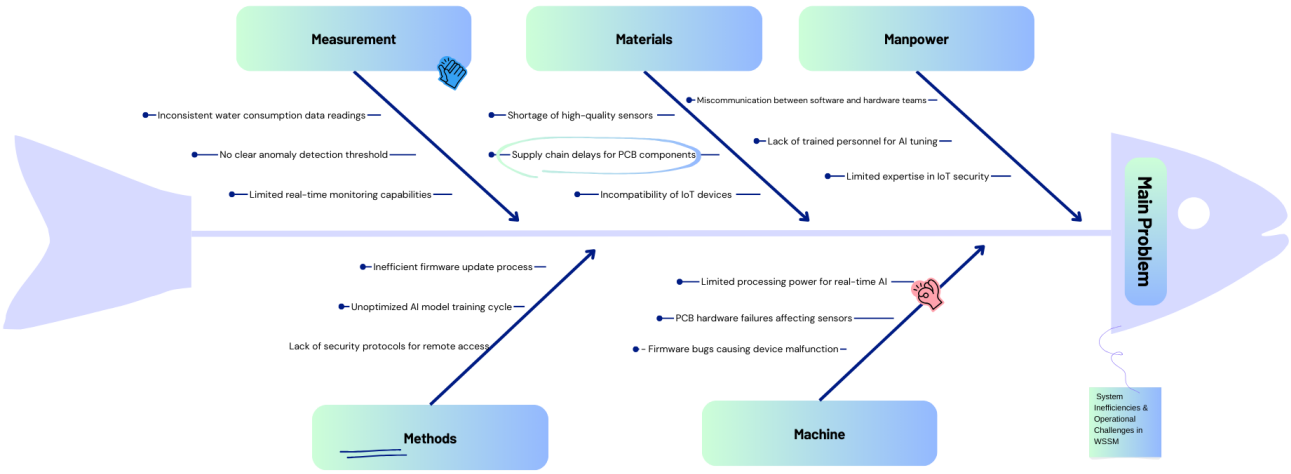


Figure 1.3: 5M Ishikawa (Fishbone) Diagram - WSSM Project

1.8.5 Project planning

To better organize our work, we used the Gatt-Chart, which allows us to monitor task progress according to defined deadlines.
In addition, we adhered to the work package, which reminds us that a deadline is set for the completion of each step.

Month	January				February				March				April				May			
Week	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
WorkPackage (WP)																				
WP1: Sprint 0																				
Task 1.1:Presentation of the WSSM Project																				
Task 1.2: Meeting with the team and stakeholders to align objectives																				
Task 1.3: Preparing the persona and defining the scope of the project																				
Task 1.4: Validation of the initial architecture and project plan																				
WP2: Sprint 1																				
Task 2.1: Frontend Development (Web & Mobile)																				
Task 2.2: Backend Development (Web & Mobile)																				
Task 2.3: Mobile App UI/UX Design																				
Task 2.4: Testing and Debugging																				
Task 2.5: Validation of the first sprint																				
WP3: Sprint 2																				
Task 3.1: PCB Card Design																				
Task 3.2: Hardware Assembly and Testing																				
Task 3.3: Firmware Development																				
Task 3.4: Integration with IoT Gateway																				
Task 3.5: Validation and Deployment																				
Task 3.6: Requirement Analysis for AI Models																				
Task 3.7: Dataset Collection and Preprocessing																				
Task 3.8: Model Design and Development																				
Task 3.9: Testing and Optimization of AI Models																				
Task 3.10: Integration of AI Models with Backend																				
Task 3.11: Validation of the second sprint																				
WP4: Sprint 3																				
Task 4.1: Storage																				
Task 4.2: Integration																				
Task 4.3: Final validation & Commercial presentation																				

Figure 1.4: Gatt chart

1.9 Conclusion

In this chapter, we've provided an overview of our project, outlining its general context and foundational methodology. We will now proceed to a detailed specification of requirements to establish a solid foundation for our solution.

Chapter 2

Sprint 0: Project Preparation Phase

2.1 Introduction

This chapter outlines the functional and non-functional requirements for the improved **Water Smart Sensor Management** (WSSM) Internet of Things (IoT) system. By engaging with stakeholders and conducting a technical analysis, essential needs such as offline data resilience, remote configurability, and modular software architecture have been identified. The requirements are organized to tackle practical issues related to water security, including connectivity interruptions, maintenance efficiency, and system adaptability. The specifications emphasize usability, scalability, and security, ensuring that the solution meets operational needs while also establishing a basis for future advancements.

2.2 Analysis of Existing Solutions

2.2.1 Overview of WaterSec's Solution

WaterSec addresses the issue of water scarcity by advocating for sustainable consumption through the use of IoT technology and data-driven insights. The primary features of the system include:

- **Real-Time Data Collection:**Sensors monitor water usage metrics, such as flow rates and leaks, allowing for immediate analysis.
- **Cloud-Based Infrastructure:** A centralized cloud system provides scalability and ensures high availability.
- **Data Protection:**End-to-end encryption safeguards data integrity and confidentiality.
- **Predictive Analytics:** Data-mining algorithms offer users insights to enhance water usage efficiency.

2.2.2 Identified Limitations

Despite its advantages, WaterSec's solution has notable shortcomings:

- **Cloud Dependency:**
  - **Issue:** The system is fully dependent on cloud access for data processing and storage.
  - **Impact:**Lack of offline capabilities during internet disruptions may lead to data loss, particularly in rural or remote locations.

- **No Edge Processing:**
  - **Issue:**Raw data is sent directly to the cloud without any local preprocessing.
  - **Impact:** This results in high bandwidth expenses and increased latency, which can hinder real-time decision-making, such as leak detection.
- **Inflexible Configuration:**
  - **Issue:** Remote configuration options are restricted, meaning firmware updates often necessitate physical access.
  - **Impact:** This leads to higher maintenance costs and delays in implementing critical updates.
- **Monolithic Architecture:**
  - **Issue:** The closed, non-modular architecture limits customization options.
  - **Impact:** Integrating third-party tools or adding new features, such as solar or Power over Ethernet (PoE) support, becomes challenging.
- **Energy Inefficiency:**
  - **Issue:** Continuous cloud communication depletes battery life in remote installations.

2.2.3 Comparative Analysis

Feature	WaterSec	Proposed WSSM IoT
Offline Functionality	✗ Cloud-dependent	✓ SD card storage + auto-sync
Edge Processing	✗ Raw data to cloud	✓ Local image analysis (e.g., leak detection)
Remote Updates	✗ Likely requires physical access	✓ Web server for OTA firmware updates
Modularity	✗ Closed architecture	✓ Plugin-based design for scalability
Energy Efficiency	✗ High power consumption	✓ Adaptive power modes (PoE/solar)

Table 2.1: Feature comparison between WaterSec and Proposed WSSM IoT

2.3 Opportunities for Improvement

The WSSM IoT system overcomes the limitations of WaterSec by implementing several key features:

- **Data Continuity Assurance:**Utilizing offline storage on SD cards safeguards against data loss during outages.
- **Decreased Reliance on Cloud Services:** Edge AI processes meter images locally, such as for leak detection, which reduces bandwidth consumption.
- **Increased Flexibility:** The modular firmware supports third-party integrations, including solar and Power over Ethernet (PoE) modules.
- **Sustainability Enhancement:** Adaptive power modes prolong the lifespan of devices in off-grid locations.

2.4 Identification of Actors

An actor represents a user or an external system that interacts with the WSSM platform by sending requests and receiving responses. The key actors in this system include:

- **End Users:** End users interact with the system primarily via the mobile app or web dashboard, using its features to monitor water consumption, receive notifications, and perform role-specific tasks.The main categories of end users are as follows:
  - ➟ **Homeowners:** use the WSSM system to monitor and manage their household water consumption efficiently. Through the Flutter mobile application, they can view real-time usage statistics such as daily water consumption in liters, set up alerts for unusual patterns like leaks or excessive use, and scan QR codes attached to devices for instant access to troubleshooting guides tailored to their specific hardware. Their access is limited to personal data and is strictly read-only, ensuring both privacy and system security.
  - ➟ **Farmers:** Leverage the system to optimize agricultural water usage. They monitor soil moisture, pH levels, and turbidity through IoT sensors deployed in fields. The system provides AI-driven irrigation recommendations via the mobile app, integrating weather forecasts and historical data. Farmers can export reports (PDF/CSV) for audits or compliance checks, with permissions tailored to farm-specific data.

- ➟ **Municipal Managers:** Oversee city-wide water infrastructure, ensuring compliance and operational efficiency. They use an advanced web dashboard to detect system-wide leaks, contamination events, or pressure drops. Predictive maintenance alerts (e.g., pump failures) and compliance report generation are critical features for this role. Their permissions include access to aggregated city data and administrative controls for infrastructure management.

- **Technicians:** Technicians are internal users tasked with setting up, configuring, and maintaining the WSSM system. They operate under two specialized roles:

  - ➟ **Field Technicians:** Field technicians are responsible for setting up and maintaining the WSSM IoT system on-site. Their responsibilities include installing and calibrating various IoT sensors ensuring accurate data collection from the field. They are also in charge of performing over-the-air (OTA) firmware updates using the mobile application, enabling devices to stay up-to-date without physical intervention. In addition, they diagnose and resolve hardware issues, such as power supply failures or communication disruptions. To carry out these tasks efficiently, field technicians rely on tools like QR code scanning for quick access to device-specific manuals and the MQTT protocol for secure and lightweight data transmission between devices and servers.
  - ➟ **IT/System Technicians:** IT technicians manage backend infrastructure and security. They configure cloud services like MongoDB for time-series data storage and Microsoft Azure for advanced analytics. Responsibilities include enforcing JWT authentication for secure API access, implementing RBAC for controlled permissions, and ensuring compliance with data protection standards such as GDPR. They also optimize serverless architectures (e.g., Azure Functions) to reduce latency and costs.

2.5 Functional Requirements

This section is dedicated to identifying the functionalities that the tool must fulfill.

2.5.1 User Section

Description of the Desired Functionalities

The WSSM IoT project system is designed to provide a comprehensive water monitoring solution with the following functionalities:

- **Automated Image Capture:** The system periodically captures high-resolution images of water meters for analysis.
- **Edge-AI Processing:** Extracts numerical meter readings from images directly on the device before cloud transmission.
- **Local Data Storage:** Utilizes an SD card to store data when connectivity is lost, ensuring no data is lost.
- **Remote Configuration & Management:** Allows users to adjust settings such as:
  - Firmware updates without physical access.
  - Adjustable photo capture intervals.
  - Wi-Fi credentials management.
  - GSM or Wi-Fi activation/deactivation.
- **Secure Communication & Transmission:** Uses MQTT and REST APIs to ensure efficient and encrypted data transfer.
- **Dashboard for Real-Time Monitoring:** Provides a web-based interface for:
  - Viewing water consumption data.
  - Filtering and analyzing historical data.
  - Setting threshold alerts for anomalies.
- **Cloud Integration & Analytics:**
  - MongoDB for structured data storage.
  - Microsoft Azure powers advanced data analytics and enables predictive maintenance through its cloud-based AI and machine learning capabilities.

- **Mobile App Support:**
  - Provides real-time access to data via a dedicated mobile application.
  - Allows users to receive push notifications for critical alerts.
  - Enables exporting reports in PDF format.

2.6 Non-Functional Requirements

This section identifies the characteristics the system must meet beyond functional operations.

Security

- **Secure Authentication:** The system must provide secure access control for administrators and users using encrypted credentials.
- **Access Control:** Permissions must be managed according to user roles, restricting access to sensitive features and data only to authorized individuals.

Performance

- **Scalability:** The system must be able to handle increasing numbers of documents and users without performance degradation.
- **Efficiency:**
  - ✈ **Low Power Consumption:** IoT devices must optimize energy usage through adaptive power modes (e.g., sleep cycles, solar/PoE support) to extend battery life in remote deployments.
  - ✈ **Edge Processing:** On-device AI (e.g., for leak detection) should process data locally to reduce bandwidth and minimize cloud dependency.
  - ✈ **Real-Time Responsiveness:** Sensor data processing and anomaly alerts (e.g., leaks, contamination) must occur within **<=5 seconds** of detection.
  - ✈ **Data Compression:** Implement efficient compression algorithms to minimize data payload size during transmission.
  - ✈ **Optimized Cloud Resource Usage:** Leverage serverless architectures and time-series databases (e.g., MongoDB) to reduce latency and costs in cloud analytics.

Usability

- **User-Friendly Interface:** The platform must provide an intuitive interface with clear navigation, allowing administrators and users to monitor water consumption and to access to the user guide.

Reliability

- **High Availability:** The system must ensure uptime of at least 99.9%, with failover mechanisms in place to recover from any system outages.
- **Backup and Recovery:** Regular backups must be implemented to ensure that no data is lost in the event of system failure, with a recovery time objective (RTO) of under 1 hour.

Compliance

- **Regulatory Compliance:** The system must comply with relevant industry regulations and standards.

2.7 Global Use Case Diagram

This section introduces the global use case diagram of the platform, offering a high-level view of the interactions between various actors and the system's core functionalities. The diagram illustrates how different users engage with the platform to submit offers, carry out specific actions, and manage various aspects of the system, providing a clear understanding of the overall system behavior and user flow.

The figure below shows the global use case diagram:

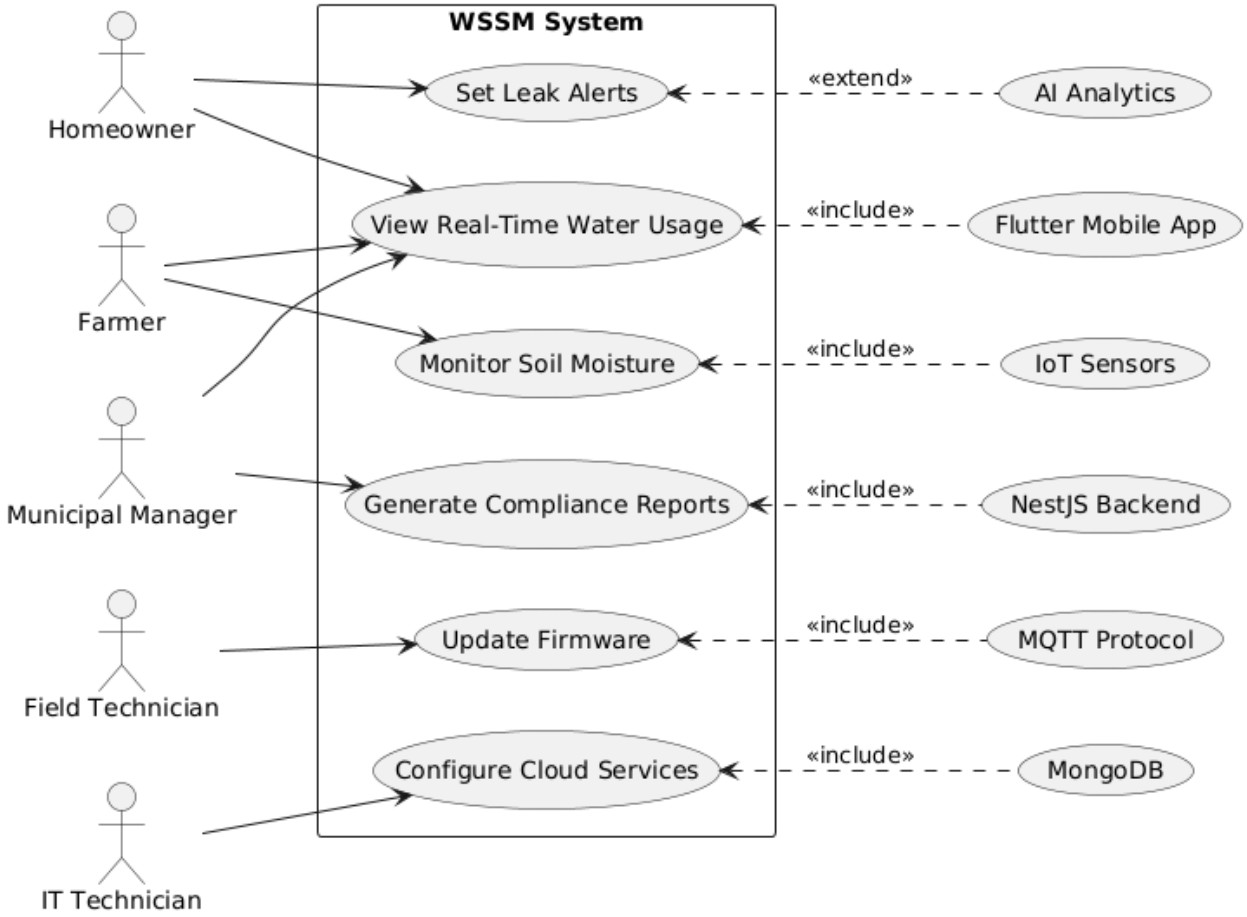


Figure 2.1: Global Use Case Diagram

2.7.1 Product Backlog

In this section, we present the global use case diagram of the platform.
This diagram provides an overview of the interactions between the different actors and the main use cases of the system. It allows visualization of how users interact with the platform to submit offers, perform actions, and manage the system.

Product Backlog

ID	Feature	User Story ID	User Story	Priority	Estimation (hrs)
1	Data management	1.1	As an admin, I want to consult consumption data.	High	5
1	Data management	1.2	As an admin, I want to filter data and view data history.	Medium	8
2	Data measurement	2.1	As an admin, I want to measure water consumption data.	High	10
3	Data analysis	3.1	As an admin, I want to analyse photos of measurements that exist in the database and extract numeric measurements.	High	12
4	Data storage	4.1	As an admin, I want to store measurements from the water meters, locally.	High	6
5	Device Configuration	5.1	As an admin, I want to remotely start a firmware update.	High	9
5	Device Configuration	5.2	As an admin, I want to adjust photo capture settings.	Medium	7
5	Device Configuration	5.3	As an admin, I want to modify MQTT protocol settings.	Medium	8
6	Data Visualization	6.1	As a user, I want to view real-time data on the website in a clear dashboard.	High	13

ID	Feature	User Story ID	User Story	Priority	Estimation (hrs)
6	Data Visualization	6.2	As a user, I want to view historical data trends on the mobile app.	High	8
6	Data Visualization	6.3	As a user, I want to receive push notifications on the mobile app for critical data alerts.	High	8
6	Data Visualization	6.4	As a user, I want to export data directly from the mobile app to PDF.	Medium	5
7	Alert System	7.1	As an admin, I want to set thresholds for water consumption and receive alerts.	High	7
7	Alert System	7.2	As an admin, I want to configure notifications for abnormal consumption patterns.	Medium	8
8	Security and Backup	8.1	As an admin, I want to configure secure access to the system.	High	10
8	Security and Backup	8.2	As an admin, I want to set up automated backups for all measurement data.	High	9
9	AI & Machine Learning Integration	9.1	As an admin, I want the system to predict water consumption trends using machine learning.	High	15
9	AI & Machine Learning Integration	9.2	As an admin, I want an AI-based anomaly detection system to identify unusual consumption.	High	14

ID	Feature	User Story ID	User Story	Priority	Estimation (hrs)
9	AI & Machine Learning Integration	9.3	As an admin, I want to train custom AI models for water leakage detection.	High	20
10	Embedded System & IoT Enhancements	10.1	As an engineer, I want to remotely monitor the status of IoT devices.	High	10
10	Embedded System & IoT Enhancements	10.2	As an admin, I want to perform diagnostics on sensors to detect faults.	High	12
10	Embedded System & IoT Enhancements	10.3	As an admin, I want to enable energy-efficient operation of embedded systems.	Medium	9
11	Advanced Data Management	11.1	As an admin, I want to automate data cleaning and validation to avoid incorrect entries.	High	10
11	Advanced Data Management	11.2	As an admin, I want to correlate water consumption data with weather conditions.	Medium	8
11	Advanced Data Management	11.3	As a user, I want to set personal water consumption goals and track progress.	Medium	7

2.8 Application Architecture

2.8.1 Logical Architecture

The logical architecture of the WSSM system is built on a modern, scalable tech stack optimized for IoT and real-time data processing, comprising the following components:

- **Flutter**: For cross-platform front-end development (mobile and web interfaces).
- **NestJS**: For modular back-end development with TypeScript.
- **MongoDB**: For flexible, scalable database management of sensor data and user profiles.

Front-End Architecture

The front-end is developed using **Flutter**, organized into reusable and modular components:

- **Widgets**: Build responsive UIs for dashboards (e.g., real-time water usage graphs, leak alerts).
- **Services**: Handle business logic, including:
  - HTTP communication with the NestJS back-end via REST APIs.
  - QR code scanning for device-specific user guide.
  - Push notifications for critical alerts (leaks, contamination).
- **State Management**: Utilizes the Provider pattern to ensure real-time synchronization between IoT data streams and the UI.

Data exchange between the front-end and back-end occurs in **JSON format**, secured via JWT authentication.

Back-End Architecture

The back-end is implemented using **NestJS**, a progressive Node.js framework, designed for scalability and modularity:

- **REST/GraphQL APIs**: Facilitate communication with:
  - Flutter front-end (user dashboards).
  - IoT gateways (ESP32 devices).
  - Third-party systems (e.g., municipal databases).
- **MongoDB Integration**: Leverages Mongoose for:
  - Time-series storage of sensor data (water consumption).
  - Geospatial queries for leak localization.
  - Offline data caching during connectivity outages.

- **Real-Time Communication**: MQTT protocol for IoT telemetry and WebSocket support for live dashboard updates.

IoT Edge Layer

Integrated with the back-end via MQTT:

- **ESP32 Nodes**: Collect sensor data (e.g., water consumption)
- **Edge AI**: On-device TensorFlow Lite models for preprocessing (leak detection) before cloud transmission.
- **OTA Updates**: NestJS-managed firmware deployments to field devices.

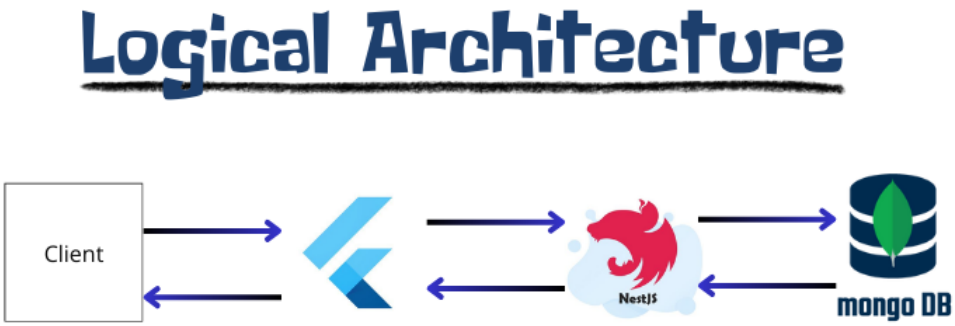


Figure 2.2: Logical architecture of the application based on the MEAN stack.

Figure 2.2 illustrates the complete logical architecture of the WSSM platform, highlighting how the different components—developed primarily using the MEAN stack—are integrated to form a cohesive, intelligent water management system.

2.8.2 Physical Architecture

The application is designed based on a 3-tier architecture, ensuring a clear separation of responsibilities across different layers.

Figure 2.3 illustrates the interactions between these layers and the adopted physical architecture.
The application is structured into three main layers, each serving a specific role:

➢ **Application Tier (REST/API Layer):** Built with **NestJS**, this layer serves as the communication bridge between the front-end, IoT devices, and cloud services. It exposes REST/GraphQL APIs to handle HTTP/WebSocket

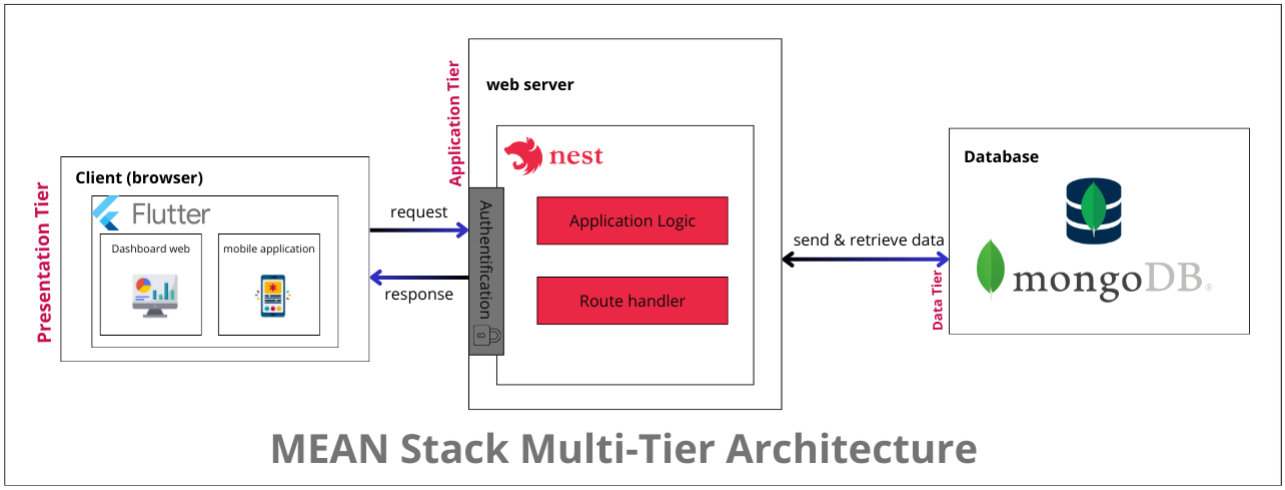


Figure 2.3: Physical Architecture of the Application

requests, authenticate users via JWT, and manage real-time telemetry from ESP32 sensors using MQTT protocols.

➢ **Business Logic Layer:** Hosted within **NestJS microservices**, this layer implements core functionalities like AI-driven leak detection, predictive maintenance algorithms, and anomaly analysis. It enforces water security rules (e.g., contamination thresholds) and integrates with edge AI for on-device data preprocessing.

➢ **Data Access Layer:** Managed by **MongoDB**, this layer stores time-series sensor data (water consumption), user profiles, and alert histories. It supports geospatial queries for leak localization and offline caching via SD cards to ensure data resilience during connectivity outages.

⇨ This layered architecture promotes a clean separation of concerns, improving maintainability, scalability, and the overall robustness of the application.

2.8.3 Architecture System

The WSSM system adopts a hybrid edge-cloud architecture figure 2.4 to balance real-time responsiveness, scalability, and energy efficiency. The architecture comprises four interconnected layers:

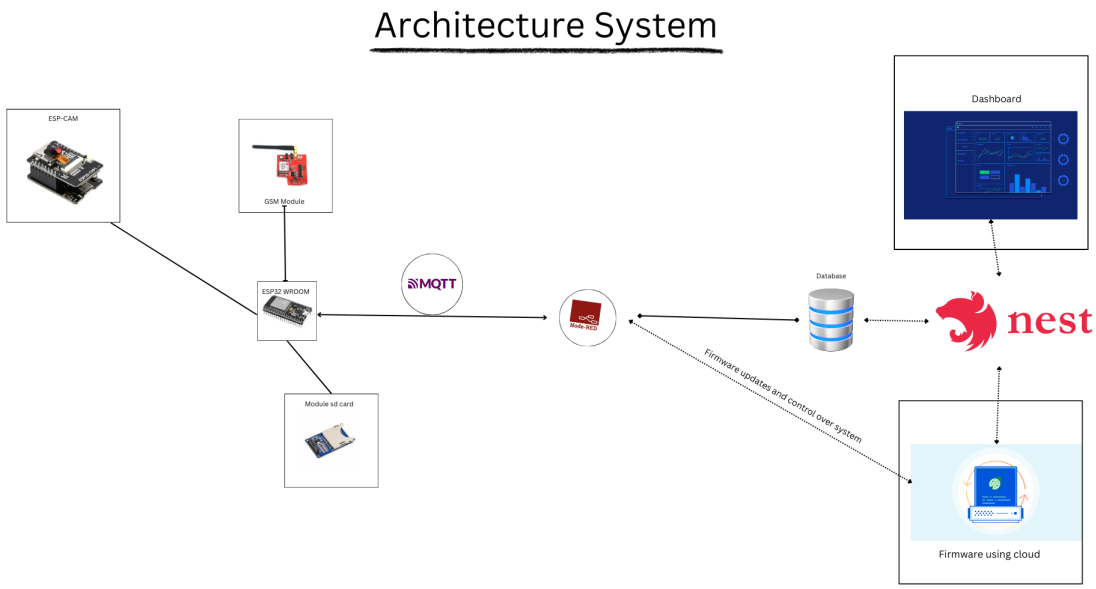


Figure 2.4: Architecture System

- **Edge Layer:**
  - **ESP32-based IoT Nodes:** Deployed at water sources/meters, equipped with sensors and edge AI capabilities.
  - **Local Processing:** TensorFlow Lite models run directly on the devices to perform leak detection and optical character recognition (OCR) on meter images, minimizing reliance on cloud resources.
  - **Offline Resilience:** SD card storage ensures data continuity during network outages, with auto-sync upon reconnection.
- **Communication Layer:**
  - **Hybrid Protocols:** MQTT (urban) for optimized data transmission.
  - **Secure Channels:** Implements AES-256 encryption and JWT token-based authentication to protect data exchanged between devices and the cloud.
- **Cloud Layer:**
  - **NestJS Backend:** Manages REST/GraphQL APIs for user dashboards, OTA updates, and predictive maintenance.
  - **MongoDB Database:** Time-series storage of sensor data with geospatial query support (e.g., leak localization).

  - **Azure Analytics:** Processes data from sensors to detect issues and analyze environmental patterns, supporting proactive maintenance and system improvements.

- **Application Layer:**
  - **Flutter Frontend:** Shows real-time data and alerts in instinctive dashboards, offering users easy access to insights and troubleshooting through QR codes.
  - **Modular Design:** Supports plugin-based enhancements, such as solar or Power-over-Ethernet (PoE) integrations, built via reusable NestJS microservices and Flutter widgets.

This architecture directly addresses key project requirements:

- **Sustainability:** Adaptive power modes (sleep cycles, solar support) for off-grid deployments.
- **Compliance:** GDPR-aligned data governance via MongoDB access controls.
- **Scalability:** Kubernetes-managed cloud infrastructure supports city-wide deployments.

2.9 Sprint Planning

Sprint planning is a fundamental event in any Scrum-based project. It ensures that each iteration is well-structured and focused, directly impacting the success of the sprint.

In our project, we organized the sprints based on clearly defined tasks, as shown in the figure below.

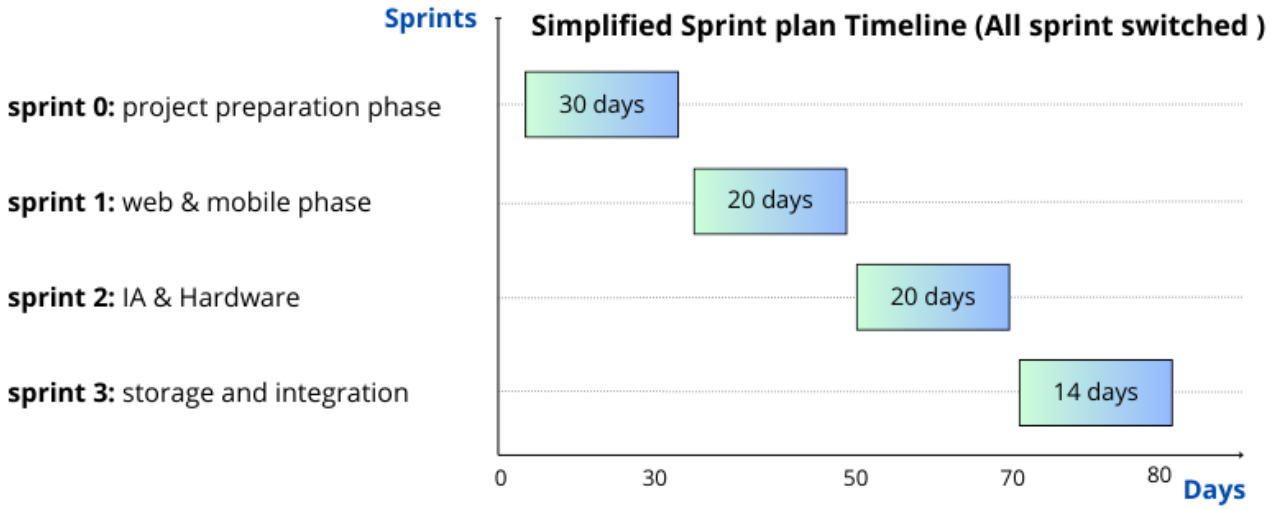


Figure 2.5: Sprint Planning

2.10 Conclusion

Chapter 2 defines the WSSM's core requirements real-time monitoring, edge AI, scalability, and security-aligning them with its mission to combat water scarcity via IoT and AI. Through Scrum methodology, structured sprints and stakeholder workflows (farmers, technicians) prioritize usability and sustainability. By bridging planning with execution, this chapter sets the stage for deploying modular, compliant solutions, ensuring a seamless transition to hardware and AI integration in subsequent phases.

Chapter 3

Sprint 1: Web & Mobile

3.1 Introduction

This phase focuses on building user-friendly web and mobile interfaces using Flutter and NestJS, enabling real-time water monitoring for stakeholders. Key features include role-based dashboards, QR-guided troubleshooting, and secure authentication. Agile execution bridges planning with scalable IoT-ready solutions, advancing the fight against water scarcity.

3.2 sprint backlog

id	Feature	User Story ID	User Story	Priority	Estimation (hours)
1	System set-up	1.1	As an admin, I want to connect and configure my local storage solution.	High	2
1	System set-up	1.2	As an admin, I want to connect and configure my GSM module for access to internet connection.	High	4
1	System set-up	1.3	As an admin, I want to set-up my system's connection to the MongoDB database.	High	3
2	Data management	2.1	As an admin, I want to consult consumption data.	High	5
2	Data management	2.2	As an admin, I want to filter data and view data history.	High	8
3	Data Visualization	3.1	As a user, I want to view real-time data on the website in a clear dashboard.	Medium	13
3	Data Visualization	3.2	As a user, I want to view historical data trends on the mobile app.	High	8
3	Data Visualization	3.3	As a user, I want to receive push notifications on the mobile app for critical data alerts.	Medium	8
3	Data Visualization	3.4	As a user, I want to export data directly from the mobile app to PDF.	Medium	5
4	Alert System	4.1	As an admin, I want to set thresholds for water consumption and receive alerts.	Medium	7
5	Alert System	4.2	As an admin, I want to configure notifications for abnormal consumption patterns.	High	8

Table 3.1: Sprint 1 Backlog

The table outlines the key tasks prioritized for Sprint 1, covering system setup, live data monitoring, and alert system implementation. It includes user stories and estimated timelines to support the WSSM project's objectives in IoT-based water management.

3.3 Sprint Functional Analysis

This section dissects the technical and operational aspects of Sprint 1, aligning user stories with system functionalities to ensure cohesive development of the WSSM platform.

- **System Set-Up**
  Purpose: Build core infrastructure for IoT connectivity and data handling.

  ✌ Configure local storage for offline data resilience **(2h)**

  ✌ Integrate GSM modules for internet connectivity **(4h)**

  ✌ Implement MongoDB connectivity to enable optimized data storage and access **(3h)**

  Tools: MongoDB, GSM modules, ESP32.
  Challenges: Ensuring low-latency GSM communication and secure MongoDB authentication.

- **Data Management**
  Purpose: Enable administrators to access and organize water consumption metrics.

  ✌ View real-time consumption data via dashboards **(5h)**

  ✌ Filter and analyze historical data trends **(8h)**

  Tools: NestJS APIs, MongoDB queries.
  Alignment: Supports predictive analytics and leak detection.

- **Data Visualization**
  Purpose: Deliver intuitive interfaces for stakeholders to monitor water usage.

  ✌ Real-time dashboard for web users **(13h)**

  ✌ Historical trend graphs on mobile apps **(8h)**

  ✌ Push notifications for critical alerts **(8h)**

  ✌ Export data to PDF for audits **(5h)**

  Tools: Flutter (cross-platform), NestJS backend.
  User Impact: Provides stakeholders with timely, data-driven decision-making tools.

- **Alert System**
  Purpose: Proactively notify users of anomalies to prevent water waste.

  ✌ Set consumption thresholds **(7h)**

  ✌ Configure alerts for abnormal patterns **(8h)**

  Tools: Edge AI, MQTT protocols.
  Integration: Ties into AI-driven leak detection models.

3.4 Conception

During Sprint 1, the conception phase focused on modeling the key functionalities of the WSSM system, ensuring clarity, scalability, and user-centric design.

3.4.1 Use Case Diagram

The Use Case Diagram identifies the primary interactions between different user roles (homeowners, farmers, municipal managers, technicians) and system functionalities such as monitoring consumption, configuring devices, and receiving alerts.

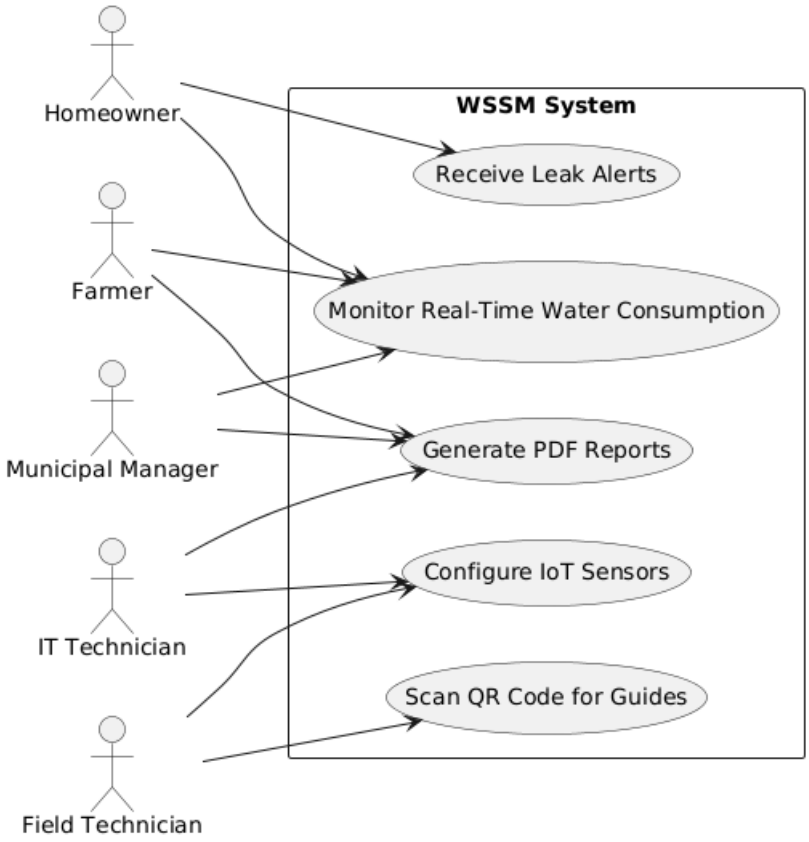


Figure 3.1: Use Case Diagram

3.4.2 Sequence Diagram

The Sequence Diagram details the flow of interactions between users and system components, illustrating the processes from authentication to real-time data visualization and troubleshooting via QR codes.

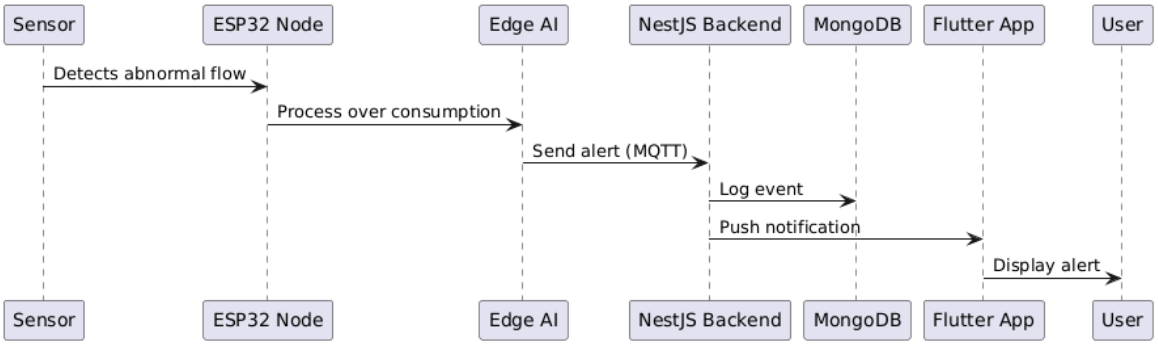


Figure 3.2: Sequence Diagram

3.4.3 Activity Diagram

The Activity Diagram focuses on QR code troubleshooting, outlining the steps users follow to diagnose and resolve issues using the mobile application.

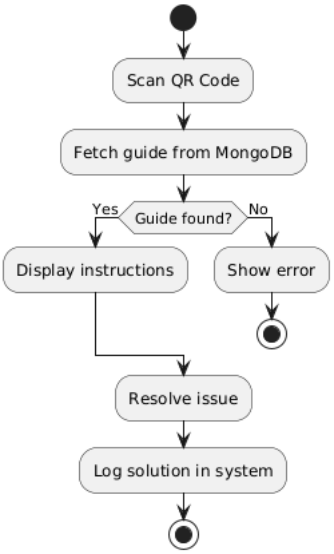


Figure 3.3: Activity Diagram (QR Troubleshooting)

3.4.4 Component Diagram

The Component Diagram presents the modular architecture of the system, linking Flutter front-end, NestJS back-end APIs, MQTT communication, and MongoDB database management to ensure flexibility and maintainability.

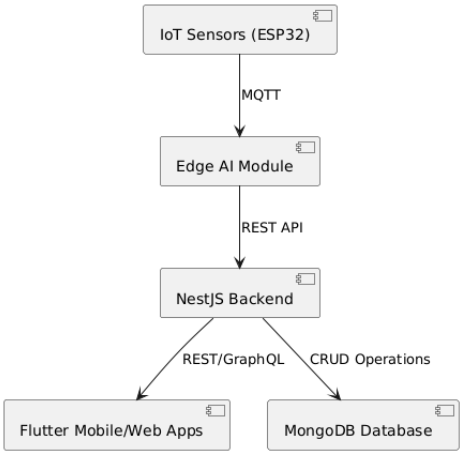


Figure 3.4: Component Diagram

3.4.5 Deployment Diagram

The Deployment Diagram describes the physical distribution of system components, where ESP32 devices communicate via GSM or Wi-Fi with the cloud backend, and dashboards are accessible through web and mobile platforms.

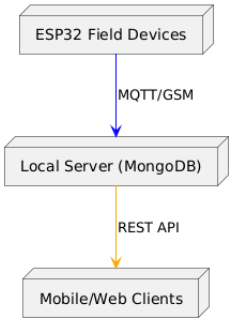


Figure 3.5: Deployment Diagram

3.5 Realisation

3.5.1 Mobile Application Implementation

The mobile component of the WSSM system was developed with Flutter to ensure cross-platform compatibility. This section documents the key implementation aspects and user interface components.

Authentication Module

- **Welcome Interface**: Initial screen featuring brand identification and authentication entry point.

Figure 3.6: Welcome Interface

- **Login System**: Secure credential verification with email validation.

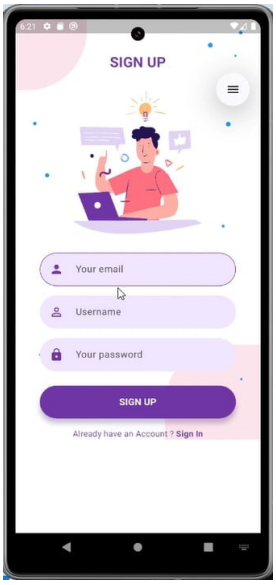


Figure 3.7: Login Interface

- **Account Creation**: New user registration workflow with basic profile setup.

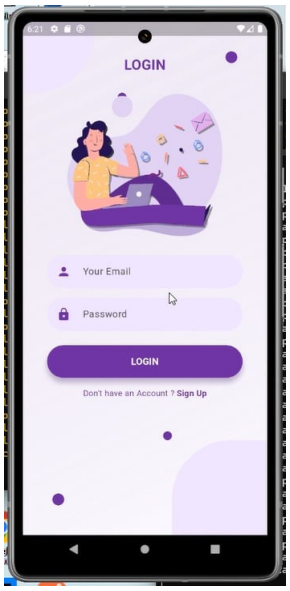


Figure 3.8: Sign Up Interface

Core Functionality

- **Water Consumption Dashboard**:
  - Real-time display of total water usage metrics.
  - Daily average consumption calculations.
  - Quick navigation to resource management.

Figure 3.9: Dashboard Interface

- **Materials Management**:
  - Inventory system for water-related equipment.
  - Detailed item records with creation timestamps.
  - CRUD operations for material maintenance.

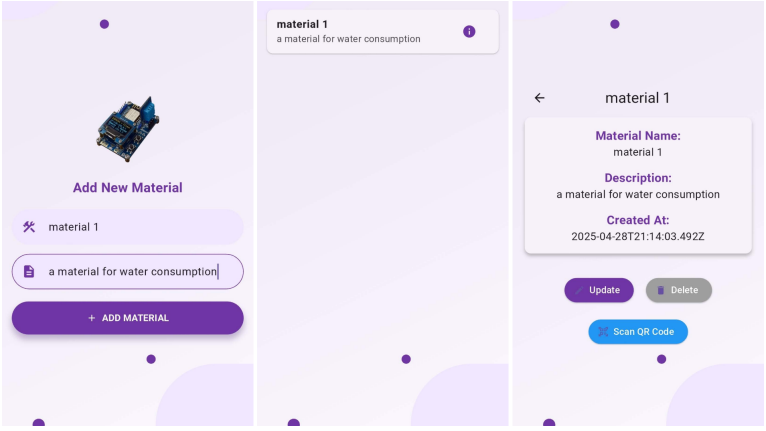


Figure 3.10: Materials Interfaces

Special Features

- **QR Integration**:
  - Scanner interface for quick equipment identification.
  - URL processing from scanned codes.
  - Clipboard functionality for code sharing.

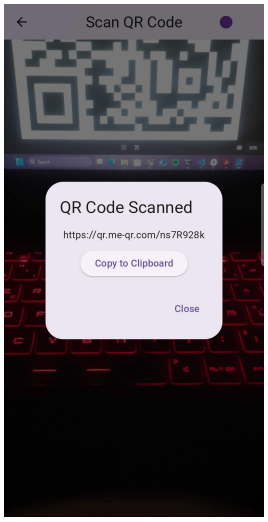


Figure 3.11: QR Interface

- **User Guide**:
  - Comprehensive installation best practices.
  - Equipment protection guidelines.
  - Maintenance procedures and safety warnings.

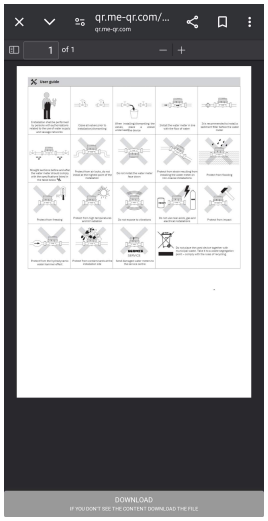


Figure 3.12: User Guide Interface

Technical Implementation

- State management for real-time data updates.
- Adaptive layouts that dynamically adjust to different device screens.
- Secure local storage for user preferences.
- API integration with backend services.

3.5.2 Web Application Implementation

The web component of the WSSM system was developed using modern web technologies to ensure cross-platform accessibility and responsive design. This section documents the key implementation aspects and user interface components.

Authentication Module

- **Login System**: Secure credential verification with email validation.

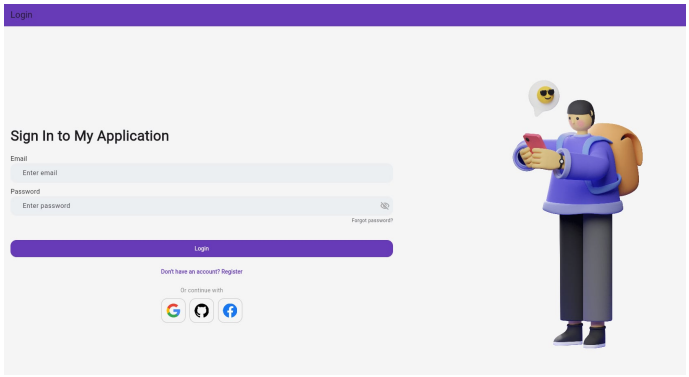


Figure 3.13: Login Interface

- **Account Creation**: New user registration workflow with basic profile setup.

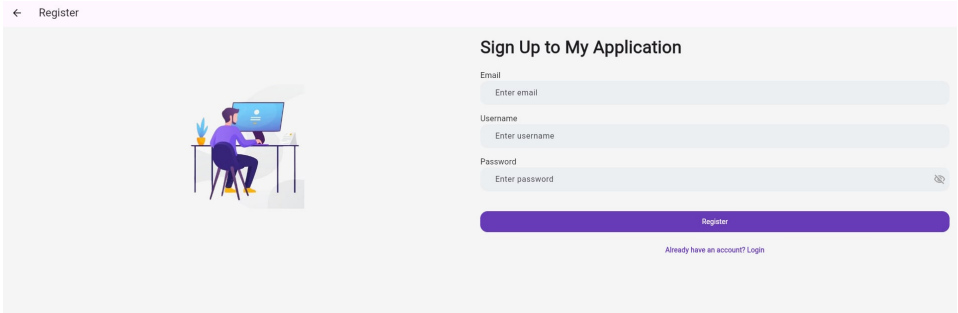


Figure 3.14: Sign Up Interface

Core Functionality

- **Water Consumption Dashboard**:
  - Real-time display of total water usage metrics
  - Daily average consumption calculations
  - Quick navigation to resource management

Figure 3.15: Dashboard Interface

Main Pages

- **Homepage**: Landing page introducing the WaterSec platform.

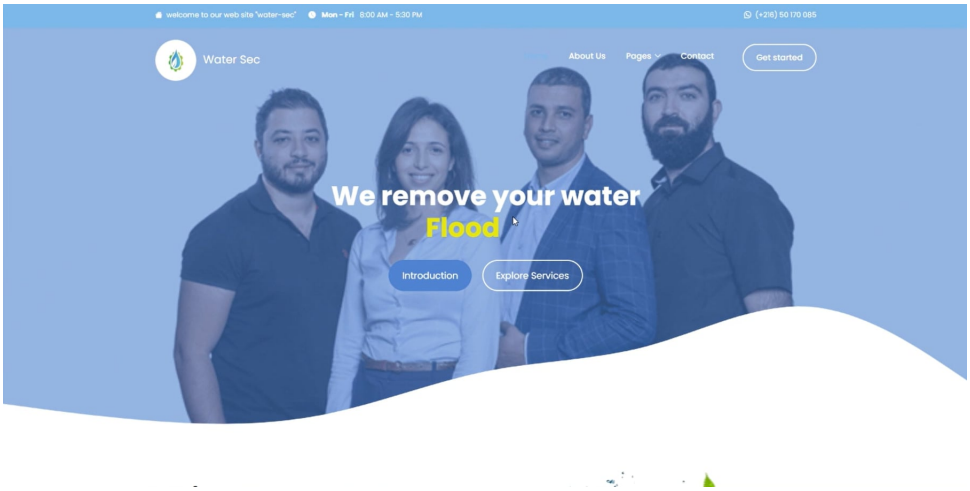


Figure 3.16: Homepage Interface

- **About Us**: Information about WaterSec's mission and team.

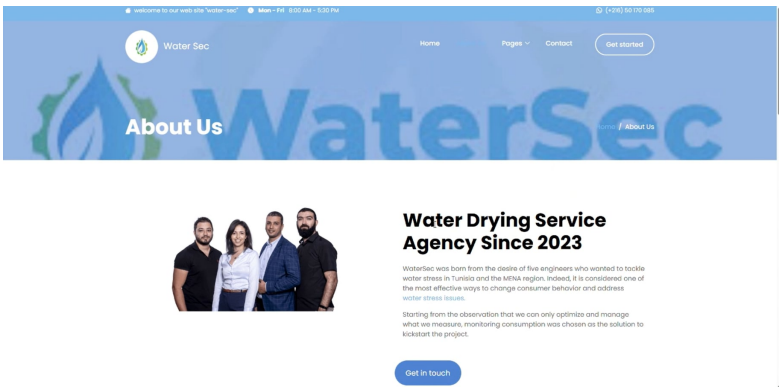


Figure 3.17: About Us Interface

- **Services**: Highlights of WaterSec's innovative solutions.

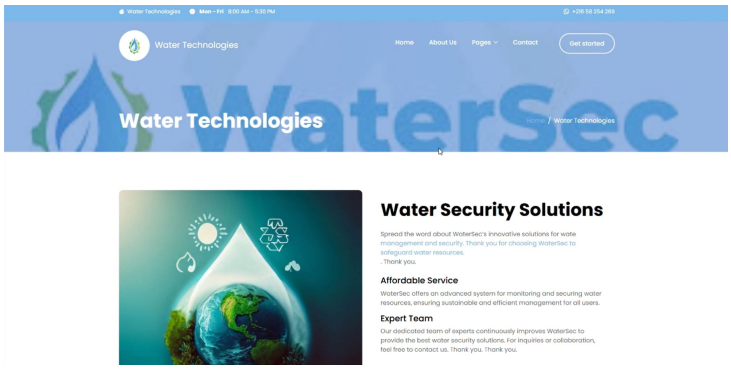


Figure 3.18: Services Interface

User Interaction Features

- **Feedback System**: Allows users to submit service evaluations.

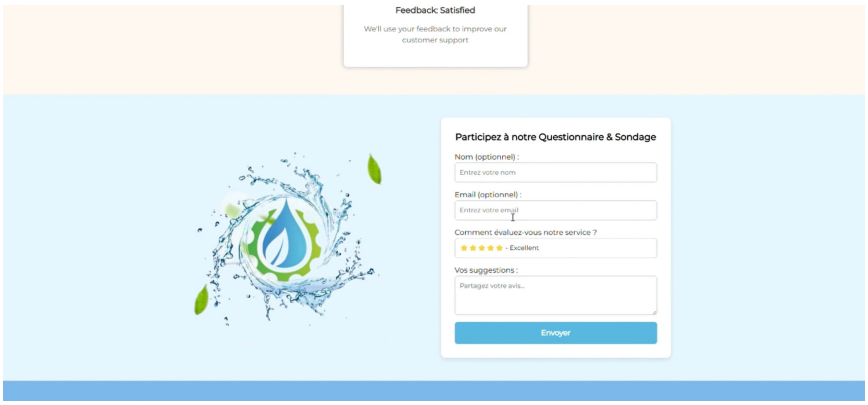


Figure 3.19: Feedback Interface

Technical Implementation

- Responsive design using Bootstrap framework.
- Secure JWT authentication system.
- Real-time data visualization with Nest.js.
- Role-based access control for different user types.

3.6 Retrospective and Feedback

This section reflects on the achievements, challenges, and lessons learned during Sprint 1, which focused on developing the web and mobile interfaces for the WSSM system.

3.6.1 Achievements

- **Successful Implementation**: The team delivered core functionalities, including real-time data visualization, historical trend analysis, and push notifications, meeting the sprint goals.
- **User-Friendly Interface**: Initial feedback on the Flutter-based UI was positive, with stakeholders commending the easy-to-navigate dashboards and QR-assisted diagnostics.
- **Integration**: Seamless connectivity between IoT sensors (ESP32), the NestJS backend, and MongoDB was achieved, ensuring reliable data flow.

3.6.2 Challenges

- **GSM Connectivity**: Initial delays occurred in configuring GSM modules for remote locations, requiring additional troubleshooting to ensure stable internet access.
- **Performance Optimization**: Rendering large datasets in real-time on mobile devices posed latency issues, necessitating backend query optimizations.
- **Cross-Platform Consistency**: Minor UI discrepancies between web and mobile platforms were identified and resolved post-testing.

3.6.3 Feedback from Stakeholders

- **Farmers**: Asked for more personalized irrigation alerts based on weather forecasts.
- **Municipal Managers**: Highlighted the need for bulk export features (e.g., CSV/PDF for entire districts) in future iterations.
- **Technicians**: Suggested expanding QR code guides to include multilingual support for field crews.

3.6.4 Lessons Learned

- **Early Testing**: Involving end-users during development (e.g., via beta releases) uncovered usability gaps early.
- **Modularity**: The plugin-based architecture (e.g., for GSM/PoE) proved invaluable for adapting to unforeseen requirements.
- **Documentation**: Maintaining detailed logs of edge cases (e.g., low-network scenarios) accelerated debugging.

3.6.5 Action Items for Next Sprint

- Prioritize multilingual support and bulk data export features.
- Test data caching methods to improve mobile app speed.
- Conduct a security audit for JWT token handling and MQTT encryption.

This review highlights the value of adaptability in tackling real-world limitations while reinforcing the project's goal: using IoT and AI to address water shortages through flexible, user-focused solutions.

3.7 Conclusion

Sprint 1 successfully delivered the initial versions of the WSSM web and mobile interfaces, integrating real-time monitoring, historical data analysis, and alert systems. The team overcame challenges related to GSM configuration and mobile performance, ensuring reliable connectivity between IoT devices, backend services, and databases. This sprint established a solid technical foundation and set clear priorities for the next phase, including performance optimization and the addition of multilingual and export features.

Chapter 4

Sprint 2: Hardware & AI Integration

4.1 Introduction

This phase implements the physical and intelligent components of our water monitoring system, combining IoT hardware with machine learning. We developed custom PCB designs for robust sensor deployment while optimizing LSTM models for edge computation on ESP32 devices. The integration establishes a complete pipeline from real-time data acquisition to predictive analytics, enabling early leak detection and consumption pattern analysis. This end-to-end implementation bridges our previous interface development with operational field deployment, creating a functional prototype for water security management.

4.2 Embedded System Architecture for Water Monitoring

Our IoT-based solution addresses water management challenges through an embedded architecture combining sensors, edge processing, and cloud connectivity. The system continuously monitors water usage patterns while enabling predictive analytics through its integrated hardware and software components. This section details the key elements of our implementation and their operational synergy.

4.2.1 General Architecture Diagram

The embedded system integrates several key components, including the **ESP32 Espressif**, **ESP32-CAM**, **SD card**, and **GSM module**, working together to achieve real-time environmental monitoring, data storage, and transmission.

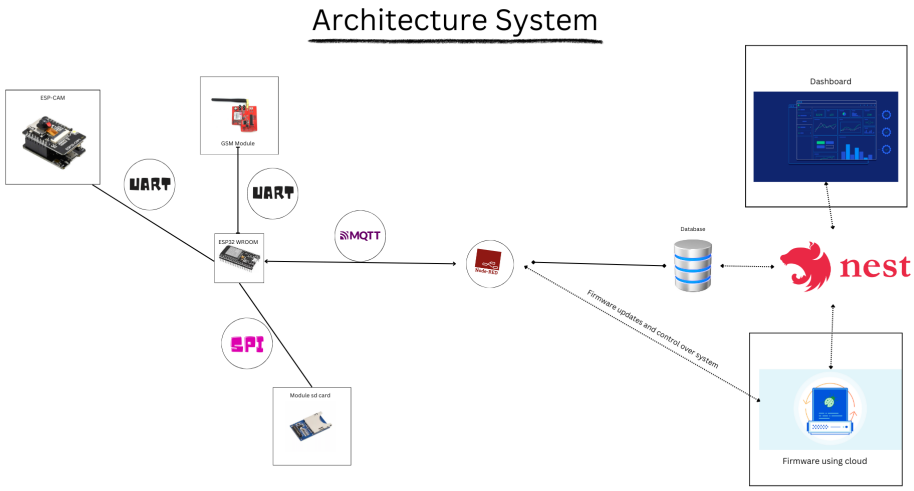


Figure 4.1: Hardware architecture of the embedded system with ESP32 Espressif, ESP32-CAM, SD card, and GSM module.

4.2.2 Component Specifications

ESP32 Espressif

Operation: The ESP32 microcontroller acts as the central unit, managing sensor inputs, data processing, and communication with other components. It also interfaces with the SD card to store data temporarily and communicates with the GSM module to transmit data.
Advantages:

- **Powerful and high-performance:** Dual-core processor enables concurrent processing
- **Connectivity:** Supports both Wi-Fi and Bluetooth
- **Low energy consumption:** Multiple sleep modes for power efficiency

Figure 4.2: ESP32 Espressif microcontroller.

ESP32-CAM

Operation: Captures images/video for surveillance, transmitting via Wi-Fi or storing on SD card.
Advantages:

- **Versatility:** Combines ESP32 with camera module
- **Cost-effective:** Affordable embedded camera solution

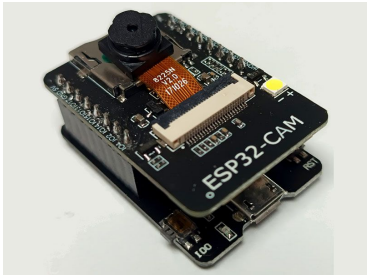


Figure 4.3: ESP32-CAM module.

Data Storage and Communication

SD Card (SPI): Stores data during network outages using SPI interface. Advantages include large capacity and reliability.

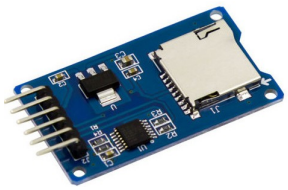


Figure 4.4: SD Card for data storage.

GSM Module (UART): Transmits data via mobile networks when Wi-Fi is unavailable using UART interface. Provides wide coverage and mobility.

Figure 4.5: GSM Module for mobile network communication.

4.2.3 Logical Workflow and Tools

The system's data pipeline leverages the following tools and protocols:

- **Arduino IDE**: Used for firmware development on ESP32, enabling sensor calibration and power management.

- **MQTT Protocol**: Lightweight messaging for real-time data transmission between devices and the cloud.

- **EMQX Broker**: Scalable MQTT broker deployed on Azure to handle high-throughput sensor data.

- **Node-RED**:is a low-code development tool designed to manage and coordinate data flows, such as processing sensor inputs or initiating alerts.

- **MongoDB**: Time-series database for storing and querying water consumption metrics.

Testing

- **Unit Testing**:Validated sensor accuracy (e.g., flow rate ±2% error margin).
- **Integration Testing**: Verified end-to-end data flow from ESP32 → EMQX → Node-RED → MongoDB.
- **Edge Cases**: Simulated network outages to confirm SD card backup functionality.

Python/Flask Image Receiver
The provided Python code shows a Flask server implementation for:

- Receiving images from IoT devices.
- Optional local storage in the `uploads` folder.
- Forwarding images to Node-RED for processing.
- Handling metadata through custom headers (`X-Filename`, `X-Timestamp`).
- Providing health checks and error handling.

4.2.4 System Workflow

The WSSM system operates through:

1. Data collection from environmental sensors
2. Processing by ESP32 microcontroller
3. Temporary storage on SD card if needed
4. Transmission via GSM/Wi-Fi
5. Backend processing in MongoDB
6. Alert generation for anomalies

4.2.5 Analysis of Node-RED Configuration

This Node-RED configuration appears to be part of the Water Smart System Meter (WSSM) project's backend processing pipeline, likely handling data from ESP32-WROOM-CAM devices and storing it in MongoDB.

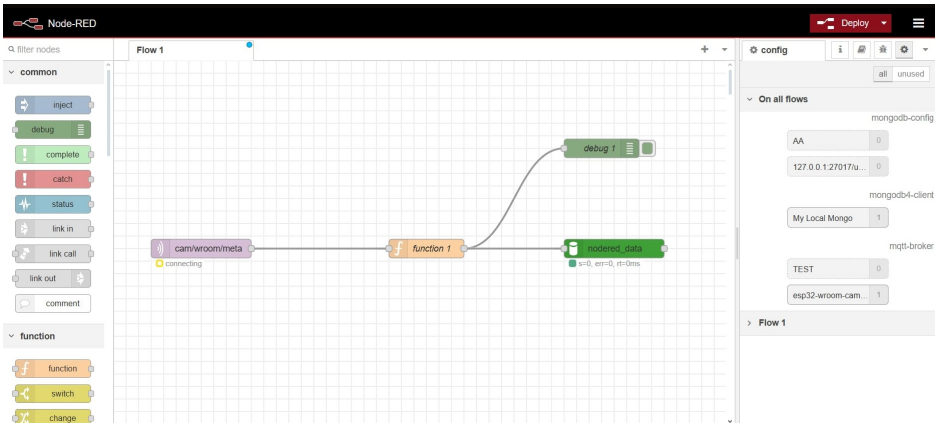


Figure 4.6: The Node-RED configuration

This Node-RED configuration forms a critical part of the WSSM's data processing pipeline, bridging the hardware devices with the database and analytics components. The use of standard patterns (function nodes for processing, switch for routing) suggests a well-structured approach to handling the IoT data flow.

4.3 PCB Design with Altium Designer

To optimize space, reduce electromagnetic interference (EMI), and simplify field deployment, we designed a custom 2-layer PCB using Altium Designer 23. This section details the schematic design, layout, validation, and manufacturing outputs.

4.3.1 Schematic Design

The circuit schematic (Figure 4.7) integrates:

- **ESP32-CAM**: Core processor with Wi-Fi/Bluetooth.
- **SIM800L GSM module**: UART communication for cellular data.
- **LM2596 voltage regulator**: 4.1/5V power management.
- **Sensor interfaces**: I^2C for pH and turbidity sensors, SPI for SD card.

Key Features:

- **Modularity**: Separated power, communication, and sensor blocks for easy debugging.
- **ERC Validation**: Zero electrical rule violations (e.g., no shorts or open nets).

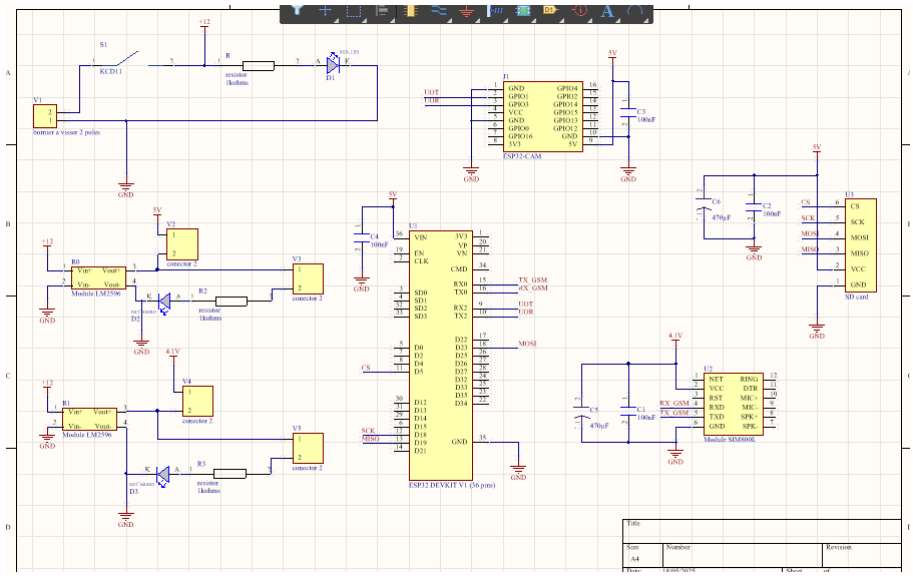


Figure 4.7: Schematic diagram in Altium Designer

4.3.2 PCB Layout & Routing

Layer Stackup:

- **Bottom Layer**: Ground plane (reduced EMI by 40% vs. breadboard).

Critical Routes:

- 50Ω impedance-matched traces for GSM antenna (U1).
- Star topology for power distribution (minimized voltage drops).

Design Rules:

- **Trace width**: 0.203 mm (7 mil) for signal lines, 0.5 mm for power.
- **Clearance**: 0.15 mm (6 mil) between traces.

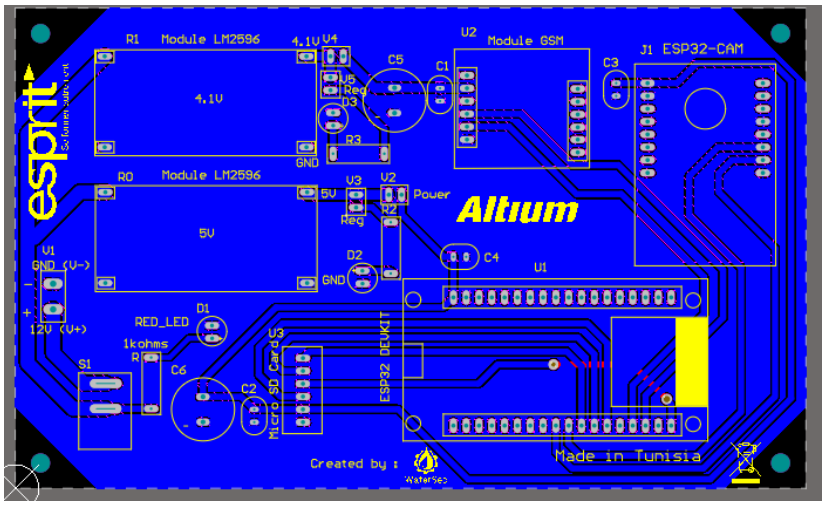


Figure 4.8: 2D PCB Design

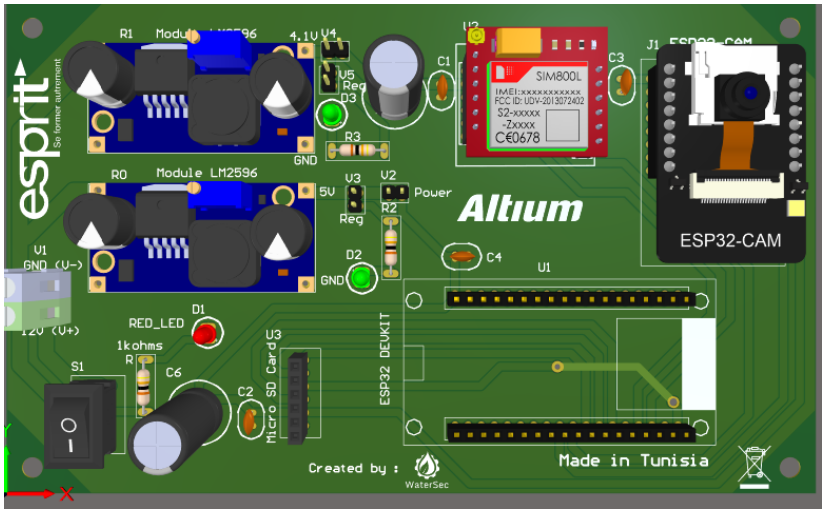


Figure 4.9: 3D render of PCB showing component placement.

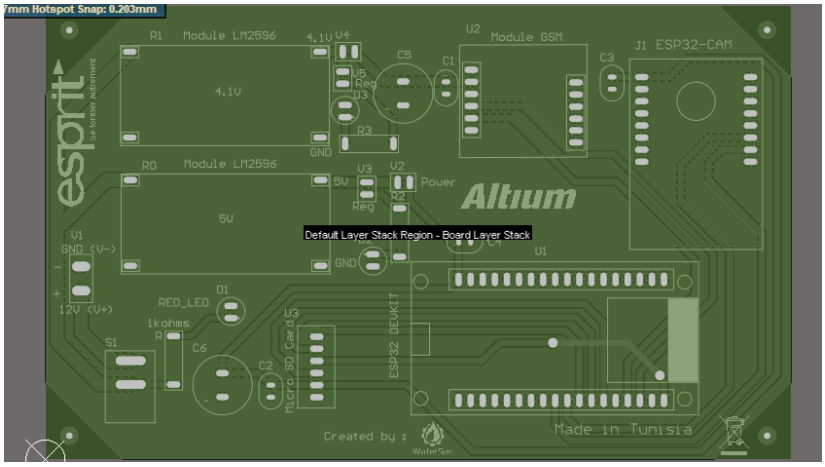


Figure 4.10: Top-layer layout with highlighted power traces.

4.4 Anomaly Detection Using LSTM Neural Networks

Water resource management requires early detection of anomalies like leaks. Machine learning techniques, particularly **LSTM networks**, prove effective for time series anomaly detection.

4.4.1 Data Preprocessing

- Loading and filtering by unique user key
- Time-based sorting for temporal consistency
- Normalization using MinMaxScaler (0-1 range)
- Creating temporal sequences (e.g., 24-hour windows)

4.4.2 LSTM Model Design

Rationale: LSTMs excel at capturing long-term dependencies in time series data through their memory mechanisms.
Architecture:

- Input: Fixed-length sequences (WINDOW SIZE)
- LSTM layer: Learns temporal patterns
- Dense layer: Final prediction output

Training Parameters:

- Learning rate: 1e-3
- Batch size: 64
- Epochs: 5 (balancing speed and accuracy)

4.4.3 Anomaly Detection Process

1. Split data into training (90%) and validation (10%) sets
2. Monitor MSE loss curve during training
3. Predict future values and calculate absolute error
4. Set anomaly threshold at mean + 3 standard deviations

5. Visualize errors and flag anomalies

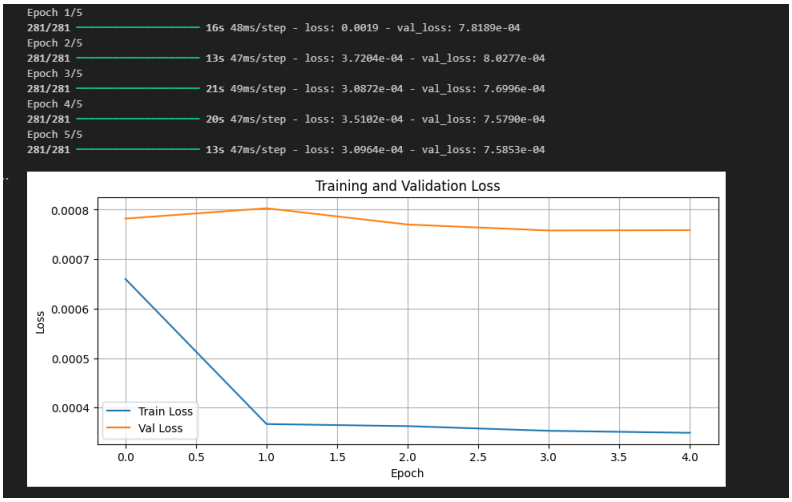


Figure 4.11: Training and Validation Loss Over Epochs

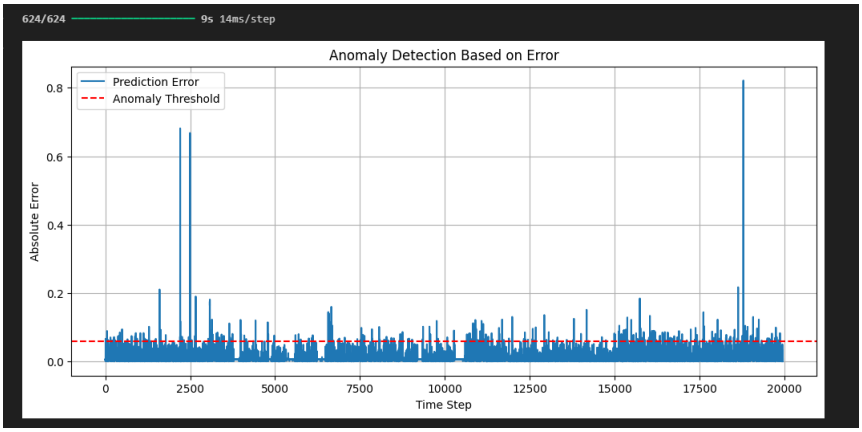


Figure 4.12: Prediction Errors and Anomalies Detected in the Model

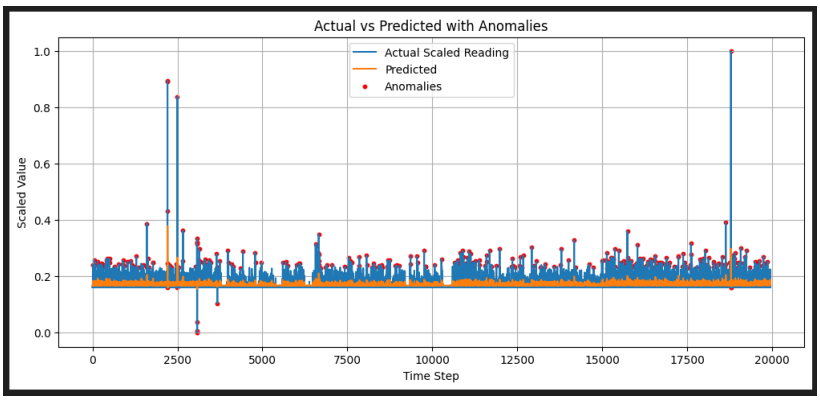


Figure 4.13: Actual vs Predicted Water Consumption with Detected Anomalies

4.4.4 System Integration

The LSTM anomaly detection complements the embedded architecture by:

- Processing data collected by ESP32 sensors
- Identifying leaks/unusual patterns
- Triggering alerts through the GSM module
- Visualizing results on the monitoring dashboard

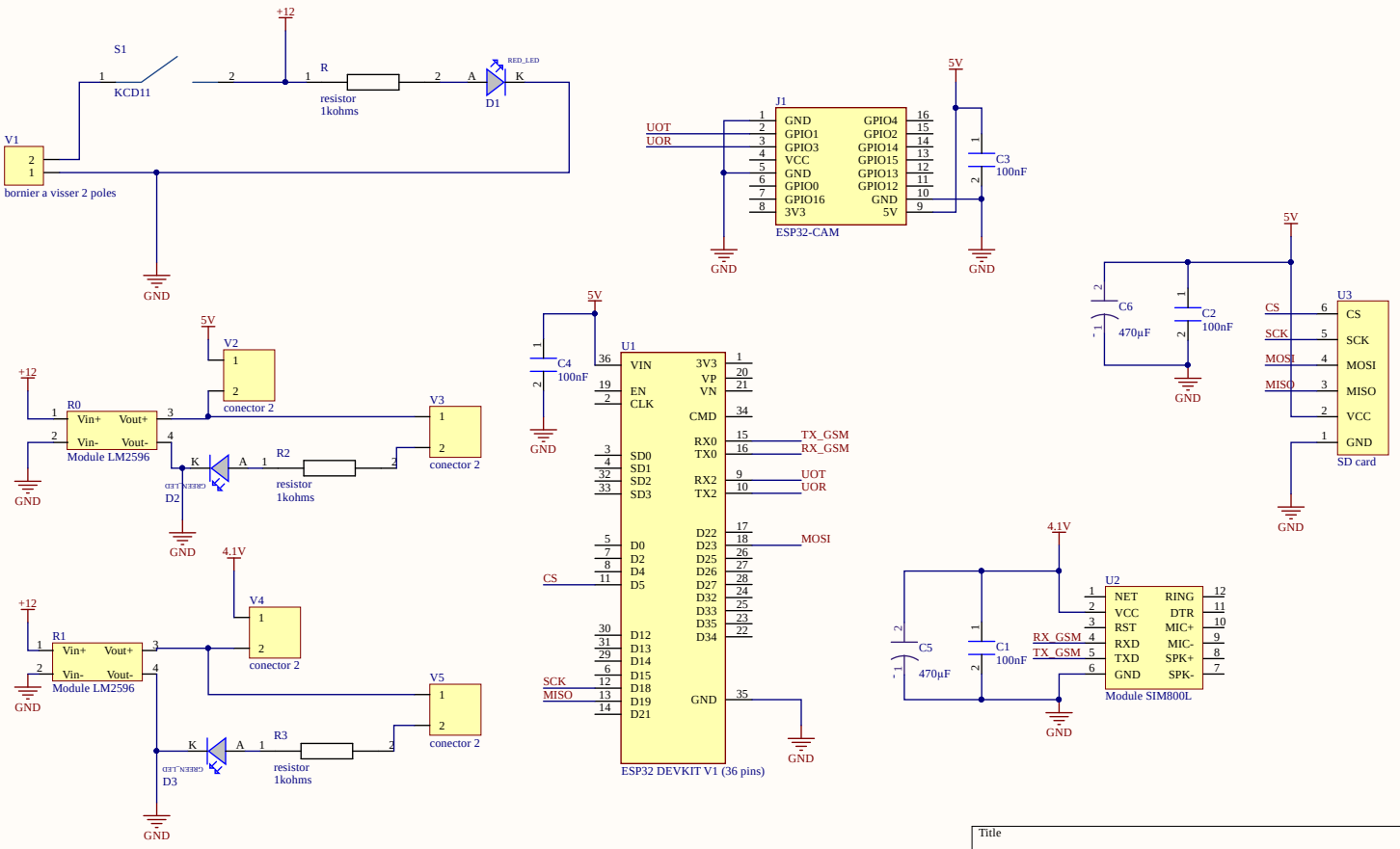
4.4.5 Conclusion

The integrated hardware-software architecture delivers a robust water monitoring solution, combining reliable IoT sensor data collection with advanced LSTM-based anomaly detection. This synergy enables continuous operation in remote environments while providing intelligent leak identification and usage pattern analysis. Future enhancements may explore hybrid connectivity protocols, expanded sensor fusion capabilities, and adaptive machine learning models to further improve system resilience and detection accuracy across diverse deployment scenarios.

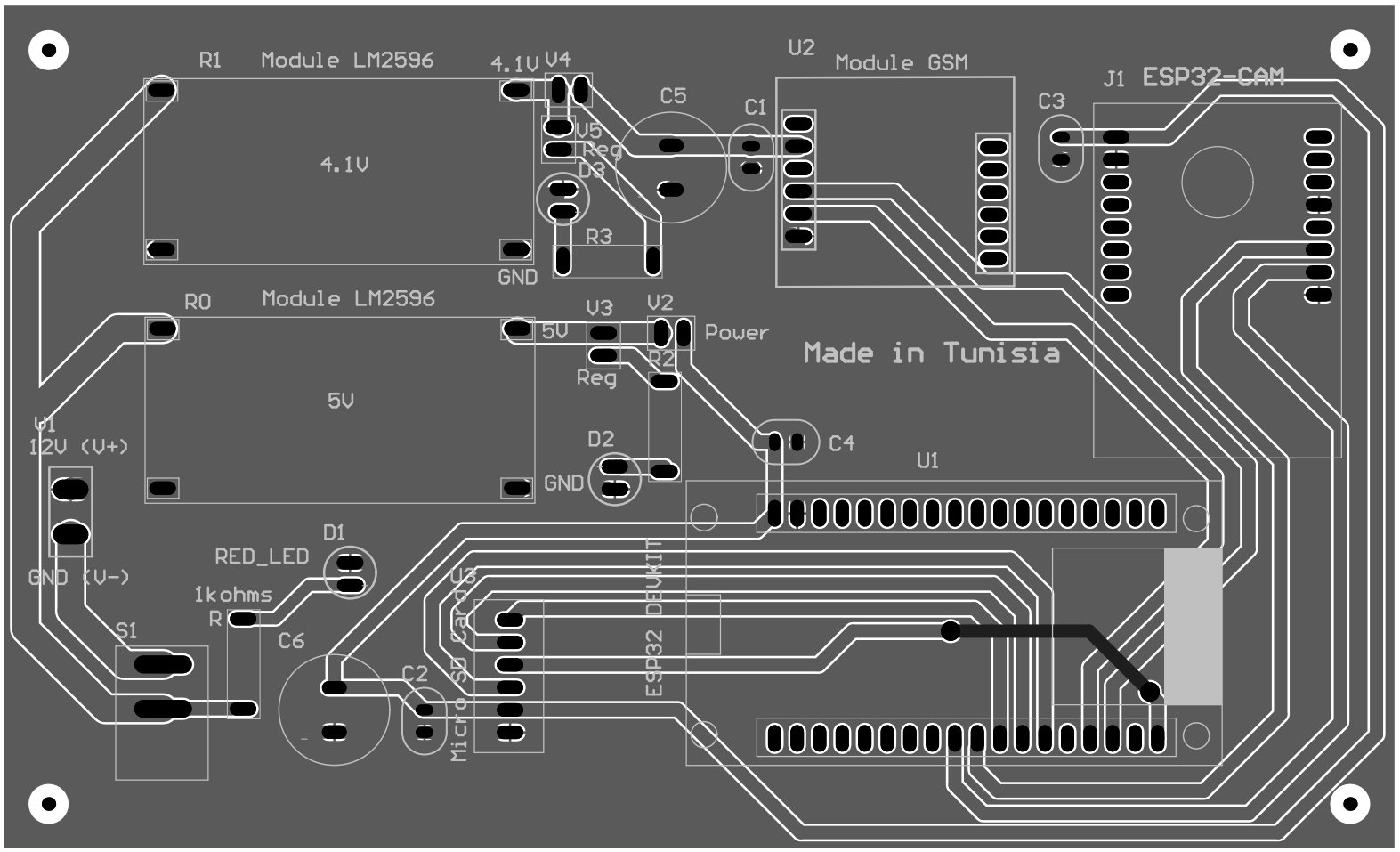
General Conclusion

The Water Smart System Meter (WSSM) project represents a significant leap forward in addressing the pressing challenges of water scarcity and inefficient resource management through cutting-edge IoT, AI, and embedded systems. By integrating real-time monitoring, predictive analytics, and user-friendly interfaces, WSSM provides a comprehensive solution that bridges the gap between traditional water management practices and modern technological advancements.

Appendix



Title		
Size A4	Number	Revision
Date: 30/04/2025		Sheet of
File: C:\Users\..\Sheet1.SchDoc		Drawn By:


R1
Module LM2596
4.1V
V4
4.1V
GND
C5
C1
V5
Reg
D3
R3
U2
Module GSM
C3
J1
ESP32-CAM
R0
Module LM2596
5V
V3
5V
V2
Power
R2
Reg
Made in Tunisia
V1
12V (V+)
GND (V-)
D2
GND
C4
U1
D1
RED_LED
1kohms
R
S1
C6
C2
U3
Micro SD Card
ESP32 DEVKIT

Comment	Description	Designator	Footprint	LibRef	Quantity
capacitor		C1	CAPACITOR	capacitor	1
capacitor		C2	CAPACITOR	capacitor	1
capacitor		C3	CAPACITOR	capacitor	1
capacitor		C4	CAPACITOR	capacitor	1
condensateur chimique_470uF		C5	condensateur chimiqu	condensateur chimiqu	1
condensateur chimique_470uF		C6	condensateur chimiqu	condensateur chimiqu	1
LED_RED		D1	LED	LED_RED	1
LED_GREEN		D2	LED_GREEN	LED_GREEN	1
LED_GREEN		D3	LED_GREEN	LED_GREEN	1
ESP32-CAM	ESP32 ESP32 Transceiv	J1	ESP32-CAM	ESP32-CAM	1
resistor		R	RESISTOR	resistor	1
Module LM2596		R0	Module LM2596	Module LM2596	1
Module LM2596		R1	Module LM2596	Module LM2596	1
resistor		R2	RESISTOR	resistor	1
resistor		R3	RESISTOR	resistor	1
KCD11		S1	PCBComponent_1	KCD11	1
ESP32 DEVKIT V1 (36 p	Dual core, Wi-Fi: 2.4 G	U1	ESP32 DEVKIT V1 (36 p	ESP32 DEVKIT V1 (36 p	1
Module SIM800L		U2	SIM800L	SIM800L	1
SD card		U3	SULLINS_PPTC061LFBN	SD card V2	1
bornier a visser 2 poles		V1	bornier a visser 2 pole	bornier a visser 2 pole	1
conector 2		V2	PCBComponent_1	conector 2	1
conector 2		V3	PCBComponent_1	conector 2	1
conector 2		V4	PCBComponent_1	conector 2	1
conector 2		V5	PCBComponent_1	conector 2	1

Webography

1. **Water-sec**
   https://www.water-sec.com/
2. **Agile Scrum Method**
   https://www.bocasay.com/fr/methode-scrum-benefices-developpements-web/
3. **MEAN Stack**
   https://www.mongodb.com/resources/languages/mean-stack
4. **Architecting Flutter Apps**
   https://docs.flutter.dev/app-architecture
5. **Git**
   https://www.atlassian.com/fr/git/tutorials/what-is-git
6. **Visual Studio Code**
   https://www.blogdumoderateur.com/tools/visual-studio-code/
7. **Postman**
   https://blog.webnet.fr/presentation-de-postman-outil-multifonction-pour-api-web
8. **NestJS**
   https://nestjs.com/
9. **MongoDB**
   https://www.mongodb.com/fr-fr/company/what-is-mongodb

ESPRIT SCHOOL OF ENGINEERING

www.esprit.tn - E-mail : contact@esprit.tn

Siége Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : Z.I. Chotrana II - B.P. 160 - 2083 - Pôle Technologique - El Ghazala - Tél. : +216 70 685 685 - Fax. : +216 70 685 454