REAL-TIME ENVIRONMENT MONITORING FOR SUSTAINABLE AGRICULTURAL PRACTICES

Julia Chan, Max Cohn, Bharat Kathi, Pablo Sandoval Rivas,
Kriteen Shrestha, and Yogananda Isukapalli
University of California, Santa Barbara

Faculty Advisor:
Dr. Yogananda Isukapalli
University of California, Santa Barbara

I ABSTRACT

An intelligent agricultural telemetry system that provides users with data-driven real-time insights and promotes sustainable farming practices is presented. This system is scalable and customizable, and leverages a distributed network of sensor modules to collect important environmental data that are communicated wirelessly. Designed for deployment in remote areas without internet access, the low-power sensor modules transmit data via long-range LoRa to a centralized base station, which does require internet connectivity. By tracking the location of each module and analyzing environmental data such as soil moisture and nutrient levels, the system enables robust environmental monitoring. The collected data is transmitted to the cloud, where it can be analyzed and used to provide artificial intelligence-aided suggestions for effective farming. This product utilizes efficiently powered embedded systems, paired with long-range LoRa wireless communication and cloud-based data infrastructure to monitor diverse agricultural environments. The presented product can be easily extended to the ground sensor or aggregation of airborne sensors scenario.

II INTRODUCTION

As sustainable resource management becomes increasingly important, environmental monitoring has become a critical focus for farmers and other crop owners. Collecting accurate real-time data can help stakeholders make informed decisions about irrigation, resource allocation, and attention to their crops. However, implementing such a system on larger scales presents challenges in balancing power efficiency, long-range communication, data integrity, cost, and system scalability.

To address these challenges, we have designed and implemented a cost-effective, scalable, end-to-end Internet of Things (IoT) system with efficiently powered sensor modules that can securely transmit environmental data over long distances using long-range (LoRa) wireless communication. These sensor nodes measure environmental parameters such as temperature, humidity, soil moisture, irradiance, and nitrogen, phosphorus, and potassium levels, and transmit the collected data to a centralized base station. In the user's hands, the base station serves as a communication bridge by forwarding the data to a cloud-based server over a LTE connection, using an MQTT protocol. The data is then ingested, processed, and stored in a secure back-end system, where various machine learning algorithms analyze the data to provide actionable insights. Based on their farm conditions, the algorithms will take real-time soil data to curate a personalized watering scheduler for the given crop. Such insights, along with real-time sensor readings, are presented through a web-based application. The front-end displays intuitive visualizations, including time-series graphs of sensor data, maps showing the locations of each sensor module relative to the base station, a soil moisture density heatmap view, and real-time notifications alerting users when certain parameters are out of range. This allows users to monitor the field conditions of various parts of their field, track soil quality trends over time, and make data-driven decisions regarding irrigation, crop selection, and environmental management.

The system is designed with scalability in mind, allowing it to support a growing number of sensor modules without significant changes to the core infrastructure. With each sensor module equipped with only essential environmental monitoring components, each node has minimal cost and communicates independently over the LoRa network using minimal power, allowing easy deployment in large or remote agricultural areas. This distributed design ensures that the system can be expanded to monitor larger fields or even multiple farms with minimal reconfiguration.

III SENSOR MODULE DESIGN

SYSTEM COMPONENTS

After thorough research, we selected sensors that are cost-effective, high value, and capable of measuring key environmental data essential for building a smart agricultural system. Listed below are each component of our sensor module and their features.

Microcontroller

This system uses the STMicroelectronics STM32L476RGTx microcontroller for its optimal balance of performance, power efficiency, and peripheral support, making it well suited for our low-power, sensor-heavy agricultural system. It features a wide range of built-in peripherals, including multiple USART, I2C, SPI, and ADC channels, enabling seamless integration with various sensors and communication modules. Additionally, we utilized the STM32Cube ecosystem, which provides configuration tools and HAL libraries to simplify peripheral setup and firmware development.

External Sensors

1) Soil Moisture (Vegetronix VH400):
The Vegetronix VH400 Soil Moisture Sensor from Vegetronix, Inc. is known for its high precision, fast response time, and minimal power consumption, which are important qualities for long-term reliable deployment in remote agricultural environments. Protected with a waterproof case, the printed circuit board (PCB) of the VH400 uses capacitive sensing technology, which provides more

stable readings and significantly reduces the risk of corrosion over time, unlike resistive sensors. The stake of the sensor uses non-corroding material and is insensitive to salinity, allowing for more accurate sensor readings. This module is equipped with an onboard analog-to-digital converter (ADC), allowing us to directly interface it with our microcontroller. To convert the sensor's analog voltage output into volumetric water content (VWC), we use the VH400's manufacturer-provided piecewise linear approximation, which models the relationship using four voltage ranges and their respective linear equations. This method enables accurate, real-time estimation of soil moisture without requiring complex calibration, making the sensor both precise and highly practical for scalable smart agriculture applications.

2) Nitrogen, Phosphorus, Potassium (MODBUS-RTU NPK Soil Sensor): The MODBUS-RTU NPK Soil Sensor was selected for its reliability in measuring critical soil nutrient levels, namely Nitrogen (N), Phosphorus (P), and Potassium (K), which are essential nutrients that influence soil health and support the specific nutrient requirements of different plant species. Because the microcontroller lacks native RS485 support, to interface with this sensor, we use an RS485 to UART converter.

3) Temperature and Humidity (Sunfounder DHT-11): The Sunfounder DHT-11 Temperature and Humidity sensor was selected for its simplicity, low cost, and its ability to measure temperature and humidity in a wide range of settings. It provides consistent readings because of its water resistant construction and large heat range. This sensor interfaces with the STM32 microcontroller via an ADC digital line, making data communication very simple and practical.

4) Global Positioning System (GPS) (Adafruit Ultimate GPS): The Adafruit Ultimate GPS Module PA1616D, which uses the MTK3333 chipset, is a very common module that offers high sensitivity and is equipped with a low power mode. The position of each module is tracked to enable context-aware analysis, such as microclimate mapping for irrigation optimization.

5) Ultraviolet (UV) and Irradiance (Si115x Proximity and Ambient Light Sensor): The Si115x Proximity and Ambient Light Sensor from Silicon Labs was chosen for its accurate measurement of ultraviolet radiation. This metric is important for monitoring crop health, as ample sunlight exposure is vital for proper photosynthetic activity and yield. This sensor communicates using the I2C protocol, allowing for seamless integration with the microcontroller.

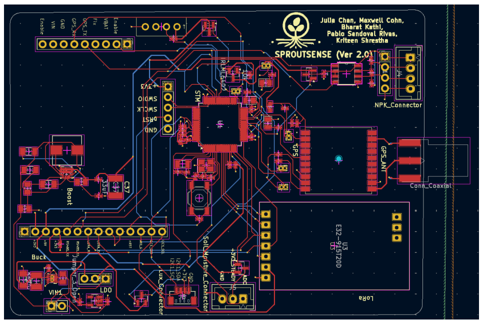
6) LoRa (E22-900T22D): The LoRa E22-900T22D module from Ebyte Technology Co., Ltd. was selected for its ability to support long-range, low-power wireless communication, which is an essential requirement for our distributed agricultural sensor network. This module can transmit data over several kilometers in open environments, making it ideal for wide-area monitoring. The module supports various spreading factors and adjustable transmission power, giving us flexibility in balancing range, power consumption, and data rate, as described in the LoRa module's user manual [4]. Its low standby current and sleep modes are especially advantageous in our low-power design, ensuring energy efficiency for solar-powered sensor nodes deployed in remote locations.

PRINTED CIRCUIT BOARD (PCB) DESIGN

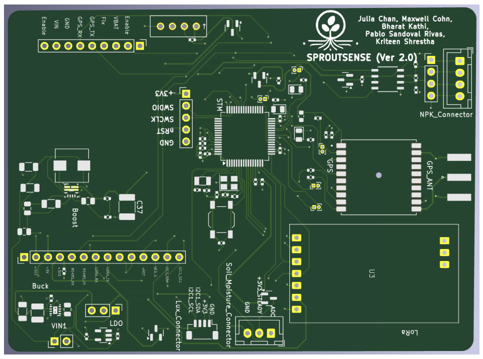
The PCB for our sensor module measures approximately 71 mm by 98 mm and manages internal connections across the sensor network. The temperature and humidity sensor, as well as the RS485 Converter, are integrated directly into the board. JST XH and SH connectors are embedded to provide easy access to external sensors such as the irradiance, soil moisture, and NPK modules. Additionally, through-hole pads are included to allow the LoRa module to be soldered via headers. The GPS chip is also integrated, along with an onboard antenna for improved signal range.

This 4-layer PCB design was chosen to accommodate a power plane and to optimize signal integrity by reducing electromagnetic interference. The multilayer structure includes a dedicated ground plane and a 3.3V power plane, along with two routing layers that provide greater flexibility and cleaner trace layout, which is particularly important given the number of components and the space needed for precise GPS trace routing. Since most components operate at 3.3V, the dedicated power plane ensures stable voltage delivery across the board. Particular attention was also paid to the routing of the GPS antenna trace, as even small impedance mismatches can significantly degrade signal quality. To ensure proper impedance matching, an impedance calculator was used to determine the optimal trace width of 0.3493 mm between the GPS antenna and the chip.

The PCB is meticulously designed with versatility in mind, allowing each component to be used in multiple configurations. Several challenges occurred while integrating the GPS module, so in the final PCB design, two options for GPS connectivity was implemented. First, through-hole pads were added to allow the Adafruit Ultimate GPS Module PA1616D breakout board [2] to be soldered directly onto the PCB. Alternatively, for a more polished and streamlined solution, the standalone GPS chip used in the Adafruit module was also directly embedded onto the board and wired it to the microcontroller. To switch between the breakout board and the onboard GPS chip, configurable jumpers were included, enabling easy rerouting of connections depending on the desired configuration. Additional components such as switch transistors, debugging pin headers, and more were added to support power management, code flashing, and improved access to the STM32 microcontroller. Visual representations of the sensor module, including PCB layout and 3D rendering, as shown in Figure 1, are available on the SproutSense GitHub repository [3].



(a) Layout



(b) 3D View

Figure 1: Layout, 3D View, and Schematic of the Sensor Module PCB

Housing Component

The main PCB, as well as cable management, will be enclosed in a 3D printed box, with screws to secure the PCB, solar chip, and battery pack inside. This housing component is a CAD designed on Autodesk Fusion 360, and is made waterproof by using PETG filament. It contains access holes for the antenna, as well as a stake component to guide external sensor cables into the soil.

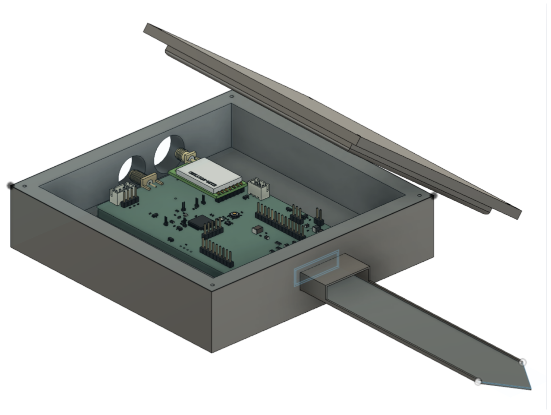


Figure 2: Housing Component CAD

IV POWER MANAGEMENT SYSTEM

Initial System Design

The initial design used the Sunny Buddy MPPT solar charger connected to a 3.7V 850mAh lithium battery and a Voltaic P126 solar panel. This configuration was intended to support autonomous operation, charging during the day and supplying power at night or in cloudy conditions. Power from the battery was routed into a custom-designed Power Management Unit (PMU), which was necessary due to the system's requirement for three distinct voltage levels (as described in the power block diagram in Figure 2):

- 3.3V rail, via a buck converter, for the microcontroller, GPS module, and irradiance sensor.
- 3.3V steady rail for the Vegetronix VH400 soil moisture sensor power.
- 5.0V rail for higher-power components such as the RS485 converter, temperature/humidity sensor, and the LoRa transceiver.

Since the battery output varies between 3.3V and 4.2V during normal operation, dedicated regulation was required to generate different voltages for different system components. To meet these demands, the PMU was designed using TI Power Designer, and consisted of:

- `TPS62172QDSGRQ1`, a buck converter configured to output 3.3V
- `TPS61236PRWLR`, a boost converter configured to output 5.0V
- `XC6220B331MR`, a 3.3V low-dropout (LDO) regulator

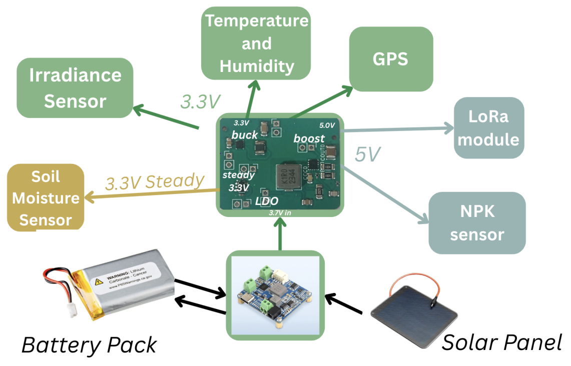


Figure 3: Power block diagram

Observed Limitations and System Upgrades

This setup performed well electrically, but through field testing, it was clear that the system could not operate continuously. With a fixed sampling frequency of once per minute, power consumption outpaced energy harvesting. The system typically operated from 8:00 AM to 3:00 AM before shutting down due to depleted battery reserves. To address these limitations, the system was revised with several improvements:

1. Replaced the 6V 2W Voltaic P126 solar panel with a 6W 12V Andoer solar panel which generated 800mAh–1000mAh more per day.
2. Modified the firmware scheduler to allow user-configurable sampling intervals of 15, 30, or 60 minutes, significantly reducing energy consumption.
3. Integrated transistors to enable firmware-based power control of each sensor via GPIO pins.
4. Upgraded to a 2000mAh lithium battery, more than doubling the available energy storage.

These changes allowed the sensor module to maintain reliable 24-hour operation and adapt more effectively to variable environmental conditions.

Theoretical Power Analysis

To verify the feasibility of overnight operation using only stored solar energy, we conducted a theoretical power analysis based on datasheet values and expected duty cycles. Each full sensor sampling event — including soil moisture, temperature and humidity, irradiance, GPS positioning, and LoRa transmission — draws approximately 250 milliamp-seconds (mAs), or 0.069 milliamp-hours (mAh). At a 60-minute sampling interval, the system runs eight such cycles per night, totaling approximately 0.55 mAh of energy usage from active sensing and data transmission.

In addition to these burst events, the system draws a continuous baseline current from components that are always on. The RS485 converter draws about 250 µA, while the STM32 microcontroller in its deep sleep mode draws around 100 µA, resulting in a total idle current of approximately 350 µA. Over an eight-hour night, this continuous draw amounts to an additional 2.8 mAh.

Theoretically, this results in a total overnight power consumption of just 3.35 mAh. Given a 2000 mAh battery, this represents less than 0.2% of total capacity, although this estimate is idealized and likely undermines actual field consumption.

Field Results and Final Adjustments

With the implemented hardware and firmware changes, the system achieved full energy autonomy under typical conditions. During field testing, the upgraded configuration successfully maintained continuous operation throughout the night. In good sunlight conditions, the solar panel was able to supply enough energy during the day to operate the module in real time and simultaneously charge the 2000 mAh battery with sufficient reserve to power the system throughout the entire night without interruption.

These results confirmed that the combination of reduced sampling frequency, improved battery capacity, and robust power management was sufficient to meet the system's energy demands over a standard day-night cycle. The only remaining scenario was when the system did not receive sunlight for an extended period of multiple days. In such a situation, the battery would eventually deplete and the system would shut off. However, once adequate sunlight returns and the MPPT module detects that the voltage and current thresholds are again met, the system will automatically power back on and resume normal operation.

Base Station Power Supply

To provide the base station with the proper power requirements, we had to ensure that the base station module received 5V of power and was connected to our MQQT broker. To supply power, we used the boost converter on the PMU which was powered by the MPPT.

V SYSTEM IMPLEMENTATION

BLOCK DIAGRAM

The architecture for the telemetry-based system is organized into three main categories: the sensor module, the base station, and the server, as described in Figure 3. Sensor modules are distributed in the field and are composed of the main PCB with external sensors. They are powered sustainably through a solar powered battery pack, and communicate wirelessly via LoRa to a nearby base station, which uses another microcontroller, the ESP32 Module, also powered by sunlight. The base station acts as a communication bridge, receiving packets via LoRa and forwarding them to a remote server. The server handles data ingestion and visualization using an MQTT broker to receive the packets, which are then processed and stored in a database. The REST API serves this data to a web-based front-end for real-time monitoring and analysis. This low-power design enables reliable long-range environmental data collection.

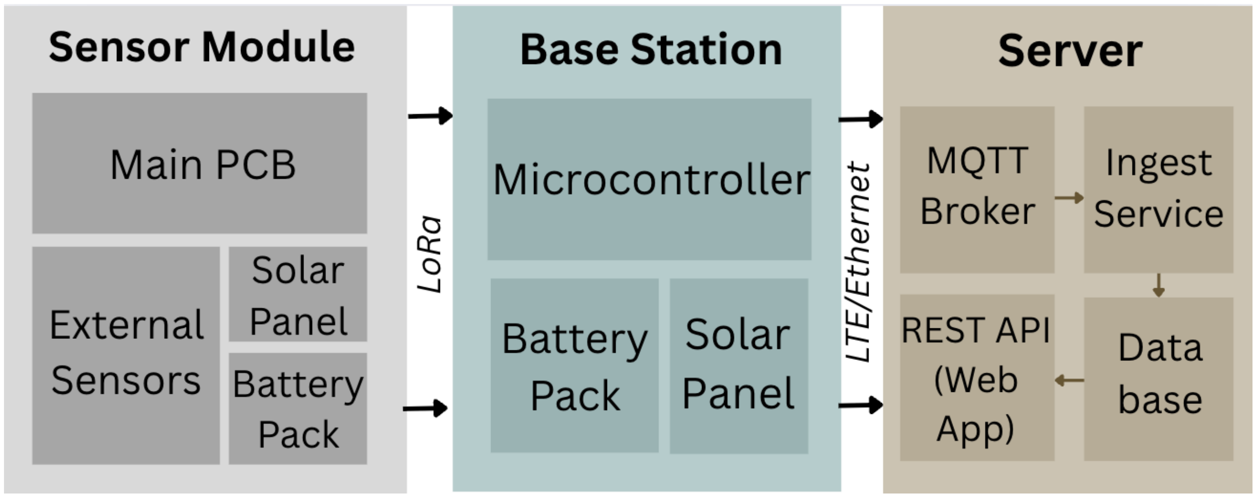


Figure 4: Block diagram

Scheduling and Data Transmission

As sensor modules send their environmental data, the goal is for each module to operate as long as possible. Hence, an appropriate data collection and transmission schedule was created. The frequency of reading such data was adjusted to fit the volatility of the data. As environmental data are collected through timer-based interrupts, a bit masking scheme is established to determine what data is sent to the base station. Each byte in the byte array corresponds to a certain sensor and is updated based on a certain frequency determined by the volatility of the sensor data. Once the byte array is filled with data, it is sent to the base station. The CPU is set to an interrupt-driven sleep to conserve power by remaining idle until incoming data triggers another interrupt. While the CPU is asleep, power to all peripherals is cut. This is accomplished by mapping transistors to GPIO pins, which control the flow of power to each peripheral. These updated byte arrays are compiled into four types of messages, containing a specific sensor reading and a unique identifier for the sensor module and its local uptime in milliseconds. Each message is transmitted through LoRa to nearby base stations, as described in Figure 4.

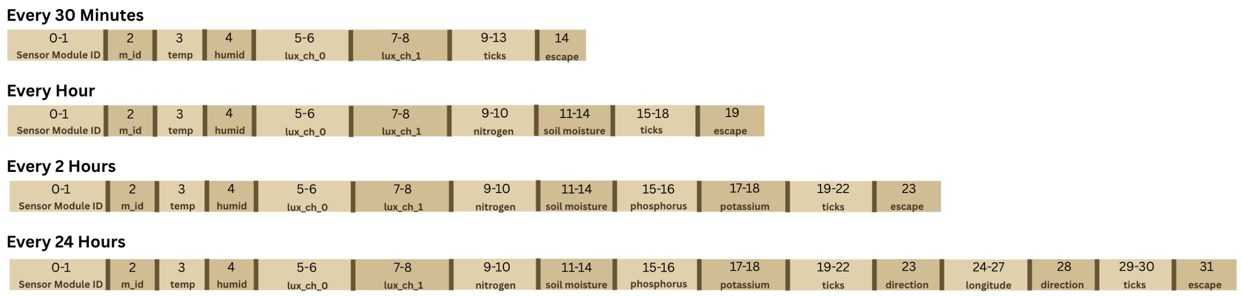

Every 30 Minutes
0-1 Sensor Module ID | 2 m_id | 3 temp | 4 humid | 5-6 lux_ch_0 | 7-8 lux_ch_1 | 9-13 ticks | 14 escape
Every Hour
0-1 Sensor Module ID | 2 m_id | 3 temp | 4 humid | 5-6 lux_ch_0 | 7-8 lux_ch_1 | 9-10 nitrogen | 11-14 soil moisture | 15-18 ticks | 19 escape
Every 2 Hours
0-1 Sensor Module ID | 2 m_id | 3 temp | 4 humid | 5-6 lux_ch_0 | 7-8 lux_ch_1 | 9-10 nitrogen | 11-14 soil moisture | 15-16 phosphorus | 17-18 potassium | 19-22 ticks | 23 escape
Every 24 Hours
0-1 Sensor Module ID | 2 m_id | 3 temp | 4 humid | 5-6 lux_ch_0 | 7-8 lux_ch_1 | 9-10 nitrogen | 11-14 soil moisture | 15-16 phosphorus | 17-18 potassium | 19-22 ticks | 23 direction | 24-27 longitude | 28 direction | 29-30 ticks | 31 escape

Figure 5: Scheduling diagram

LoRa: Sensors to Base Station

To achieve long-range, low-power communication between sensor modules and base stations, we selected **LoRa (Long Range)** as the wireless transmission protocol. LoRa operates in the sub-GHz ISM band and uses chirp spread spectrum modulation, which allows for robust communication over several kilometers, including noisy or obstructed environments because of its ability to demodulate below the noise floor.

We chose LoRa over alternatives such as Wi-Fi or Zigbee due to its superior range, resilience in rural and agricultural settings, and its suitability for transmitting small packets of sensor data with infrequent updates. The low bandwidth of LoRa is well suited to our application, which prioritizes energy efficiency and long operational lifetimes over high data throughput.

The network follows a star topology, where multiple sensor modules independently transmit data to nearby base stations. This decentralized structure allows any sensor to communicate with any base station within range, enabling dynamic routing in the event of a base station failure. If a primary base station becomes unreachable, modules can automatically reroute their transmissions to alternate stations, ensuring fault tolerance and robust data delivery without requiring manual reconfiguration.

To minimize packet collisions in our network of dozens to hundreds of sensor modules, we implemented a custom Listen Before Talk (LBT) strategy. Rather than using the built-in channel activity detection of LoRa modules, which can be unreliable or even missing across different hardware, we opted for a passive approach that adapts over time.

Each sensor module continuously estimates the current network congestion by maintaining an Exponentially Weighted Moving Average (EWMA) of the observed `bytes per second (BPS)` transmitted through the LoRa channel. This is done passively by monitoring the airtime of messages from other modules without transmitting. Before attempting to send its own data, a module compares the current BPS with a dynamic threshold. If the estimated BPS is less than 0.7 (indicating a quiet network), the module proceeds to transmit, but if the BPS exceeds the threshold, the module backs off for a randomized delay period before checking again.

This passive, decentralized scheduling system avoids synchronized transmission peaks and adapts automatically to varying network densities. It eliminates the need for centralized coordination or frequency hopping, yet ensures that messages are sent with high reliability and minimal collisions.

Our LBT implementation is lightweight and energy-efficient, designed specifically for ultra-low-power microcontrollers with limited computational resources. Since the system is only activated when it is time for a module to send a message, we can maintain the normal sleep schedule, extending battery life while maintaining consistent data delivery to base stations.

SOFTWARE PIPELINE

To access the environmental data collected by the sensor modules, a lightweight, scalable, and reliable ingestion system was designed. It consists of three main components: the MQTT communication layer, the ingestion service, and the database/API backend.

MQTT Broker

Each base station manages communication with multiple sensor modules via LoRa. Upon receiving a message, the base station decodes and parses the data before forwarding it to a remote server using the Message Queuing Telemetry Transport (MQTT) Protocol. As an MQTT client, the base station publishes messages to the topic `ingest/{id}`, where `id` corresponds to its unique identifier.

To ensure reliable communication even over intermittent LTE connections, the system uses NanoMQ, a lightweight open-source MQTT broker. Its low resource footprint and support for persistent queues make it ideal for constrained environments, ensuring that messages are reliably stored and forwarded to subscribers without data loss.

Ingestion Service

The ingestion service is a lightweight, modular Python application responsible for processing real-time messages from the MQTT broker and storing them in a structured format. Upon receiving a message from a base station, the service deserializes the data payload and extracts individual sensor readings (e.g., temperature, humidity, soil moisture), along with metadata such as timestamps, sensor module ID, and base station ID. Once processed, these data are written into a PostgreSQL database with appropriate indexing to enable efficient retrieval and filtering. The metadata associated with each data point such as which sensor module and base station it came from can be used to track how the network evolves over time, as more modules are added.

In addition to real-time ingestion, the service exposes a comprehensive REST API using the Flask framework. This API provides a set of endpoints that allow clients to query specific sensor values (e.g., soil temperature, humidity), filter by sensor module ID or base station ID, or request data for a given time range.

To handle sensor data that may arrive at slightly different times (due to network latency or transmission jitter), the ingestion service implements a time series synchronization mechanism. This process matches readings across sensors based on their timestamps, with a default matching tolerance of 50 milliseconds. Clients can customize the behavior using several alignment strategies: forward fill, backward fill, or interpolation. Forward fill uses the most recent earlier timestamp to fill missing values, backward fill uses the next closest later timestamp, and interpolation estimates missing values through linear interpolation.

This synchronization enables the system to deliver consistent, aligned datasets that are ideal for plotting multi-sensor trends or feeding machine learning models.

Database

PostgreSQL was chosen as the core database system due to its robustness, extensibility, and ecosystem maturity. It provides ACID-compliant transactions, powerful indexing, and flexible querying through standard SQL, making it ideal for managing structured time-series data from sensor modules across multiple deployments.

To efficiently handle both high-frequency, real-time data and large volumes of historical telemetry, the SproutSense backend adopts a hybrid rowstore/columnstore architecture enabled by the Mooncake extension for PostgreSQL. Mooncake allows Postgres to natively query Apache Iceberg tables stored in object storage (e.g., Amazon S3) using standard SQL.

Under the hood, Mooncake embeds the DuckDB analytical engine, enabling zero-copy, columnar access to remote datasets without requiring data duplication or migration into PostgreSQL. This architecture seamlessly blends a hot path, where recent sensor data is written into PostgreSQL rowstore tables for low-latency ingestion and queries, and a cold path, where older or archived data is stored as Iceberg-formatted columnstore tables on S3.

Using Mooncake, complex queries such as joining real-time sensor values with long-term historical patterns, can be executed transparently across both storage tiers. This unlocks OLAP-style analytics at scale, while keeping operational costs low by leveraging inexpensive object storage for historical data.

Schema Overview:

- **base_station**: Stores metadata for each deployed base station, including geographic coordinates and last known activity.
  - `id`, `name`, `latitude`, `longitude`, `last_ping`, `created_at`
- **sensor_module**: Tracks all registered sensor modules, with location and heartbeat timestamps for monitoring health and connectivity.
  - `id`, `name`, `latitude`, `longitude`, `last_ping`, `created_at`
- **sensor**: The core data table where environmental measurements are stored. Each row includes the sensor type, raw value, source IDs, and millisecond-level precision.
  - `id`, `bsid`, `smid`, `name`, `value`, `millis`, `created_at`
- **bps**: Captures the raw byte-per-second throughput of each base station, useful for diagnosing network performance or identifying communication bottlenecks.
  - `id`, `bsid`, `value`, `created_at`

This schema is optimized for time-bounded and geospatial queries, with indexed fields such as `created_at`, `bsid`, and `smid` to support performant filtering. As data volumes grow, older rows in the `sensor` table are automatically exported to Iceberg tables and removed from PostgreSQL, ensuring long-term scalability while maintaining fast access to recent data.

Deployment

The SproutSense software stack (which includes the MQTT broker, ingestion service, and PostgreSQL database) are deployed on a lightweight yet scalable AWS EC2 instance using the `t4g.micro` instance type. This ARM-based instance offers a cost-efficient foundation for continuous operation, optimized for both burstable compute and low idle overhead.

All services run in a containerized environment for consistency and portability. The base stations transmit sensor data to this EC2 instance over LTE networks using MQTT, which is then processed and stored in real-time by the ingestion service and PostgreSQL.

To support scalable data retention and high-performance analytics, the deployment is integrated with Amazon S3 for long-term object storage. Older telemetry data is periodically exported from PostgreSQL rowstore tables and written to Apache Iceberg tables on S3 as mentioned in the previous section.

This architecture empowers SproutSense to maintain real-time responsiveness for recent sensor data and scale cost-effectively by offloading historical data to durable cloud storage. It also enables deep analytics directly from S3 using SQL, with no duplication or latency penalties and seamlessly integrate future services (e.g., dashboards, anomaly detectors, ML pipelines) without architectural changes.

The use of AWS as a deployment backbone ensures high availability, with built-in options for horizontal scaling, automatic backups, and eventual migration of specific components to managed services (e.g., Amazon RDS, Lambda, or IoT Core) as system demands evolve. At the same time our containerized deployment enables our software stack to be easily deployed on any other cloud or on-premises for super remote or high security environments.

Web Application

To provide users with intuitive access to sensor data, a web-based application is developed using React.js. The main components discussed are shown in Figures 6–9 and the web application code can be found on GitHub [3].

Data Visualization

Each environmental parameter, such as soil moisture, temperature, humidity, and light intensity, is accompanied by an interactive graph on the dashboard. Users can toggle between time intervals that include one day, one week, one month, and one year to visualize how sensor values change over time. This flexibility enables users to easily identify trends, seasonal patterns, and anomalies, supporting more informed decisions about plant care and environmental management.

Map View

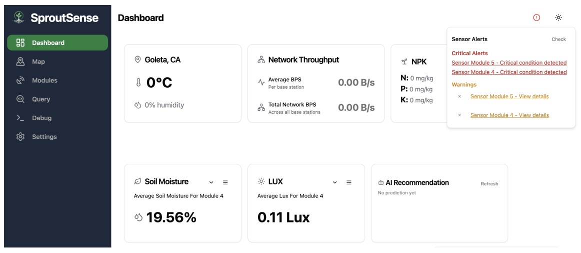
The dashboard includes a map interface that displays the geographic location of each sensor module. Users can click on individual modules to view detailed, module-specific information, including real-time sensor readings and active alerts. This spatial representation helps users to intuitively identify where issues are occurring. Additionally, users can toggle between a soil moisture density map view, enabling a quick visual assessment of moisture distribution throughout the monitored area.

Alert System

To enhance responsiveness to anomalies, an alert system was implemented within the dashboard. This system continuously monitors the incoming sensor data and compares them with plant-specific threshold values defined for parameters such as temperature, humidity, soil moisture, and nutrient levels. When values fall outside the acceptable range, alerts are generated and visually presented to the user. Critical alerts, such as dangerously low humidity or temperature extremes, are distinctly emphasized and cannot be dismissed, while standard warnings can be snoozed or muted. Users can select the duration of snooze from pre-configured options or specify a custom future date and time to resume notifications, ensuring flexibility and control over alert management.

Predictive Water Scheduler

As sensor data was integrated into the database, a predictive watering scheduling model was developed in parallel to extract meaningful insights and provide actionable recommendations from the collected data. This irrigation decision model was trained using a publicly available dataset originally collected for cotton crop irrigation management [7], which included environmental parameters such as soil moisture, temperature, and air humidity, along with corresponding optimal irrigation decisions. For this specific application, to provide more informed irrigation advice, weather forecast data was also adapted into the dataset. Using the OpenWeatherMap API to obtain weather and precipitation data, decision rules that consider both current soil status and anticipated weather conditions were implemented. Instead of simply determining whether crops should be watered, the dataset was also adapted to recommend irrigation intensity, i.e. whether to apply more or less water, based on the current soil conditions and predicted weather patterns. To refine the model, the binary classification was expanded into three categories: **Heavy Watering**, **Normal Watering**, and **Light Watering**, determined by considering additional parameters. It is important to note that this model serves as a proof-of-concept and demonstrates the feasibility of our approach. Due to time and capacity limitations, this model was trained only on one specific dataset on cotton crops, limiting the generalization of this model to other crops.



Dashboard

Goleta, CA
0°C
0% humidity

Network Throughput
Average BPS — Per base station — 0.00 B/s
Total Network BPS — Across all base stations — 0.00 B/s

NPK
N: 0 mg/kg
P: 0 mg/kg
K: 0 mg/kg

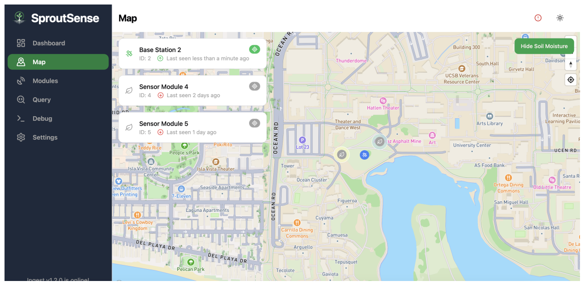
Sensor Alerts — Check
Critical Alerts
Sensor Module 5 - Critical condition detected
Sensor Module 4 - Critical condition detected
Warnings
Sensor Module 5 - View details
Sensor Module 4 - View details

Soil Moisture
Average Soil Moisture For Module 4
19.56%

LUX
Average Lux For Module 4
0.11 Lux

AI Recommendation — Refresh
No prediction yet

Figure 6: Dashboard

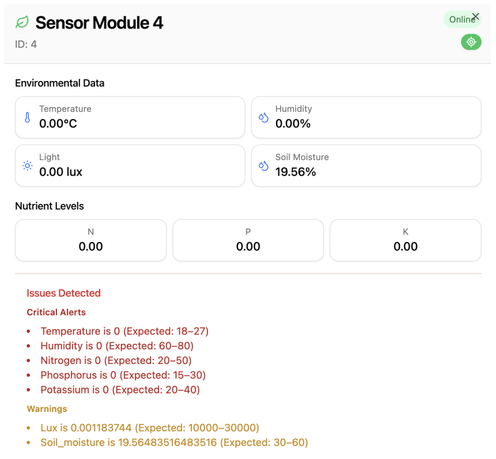


Base Station 2
ID: 2 Last seen less than a minute ago

Sensor Module 4
ID: 4 Last seen 2 days ago

Sensor Module 5
ID: 5 Last seen 1 day ago

Figure 7: Map



Sensor Module 4 — Online
ID: 4

Environmental Data

Temperature
0.00°C

Humidity
0.00%

Light
0.00 lux

Soil Moisture
19.56%

Nutrient Levels

N	P	K
0.00	0.00	0.00

Issues Detected

Critical Alerts

- Temperature is 0 (Expected: 18–27)
- Humidity is 0 (Expected: 60–80)
- Nitrogen is 0 (Expected: 20–50)
- Phosphorus is 0 (Expected: 15–30)
- Potassium is 0 (Expected: 20–40)

Warnings

- Lux is 0.001183744 (Expected: 10000–30000)
- Soil_moisture is 19.56483516483516 (Expected: 30–60)

Figure 8: Module Card

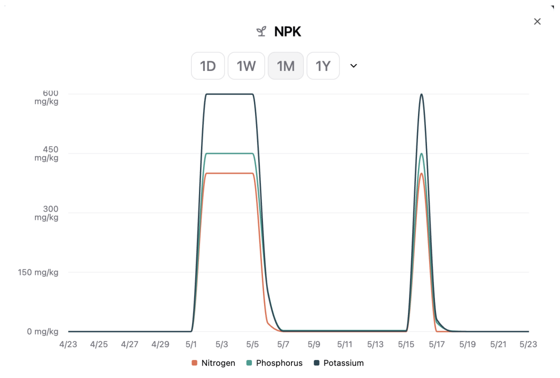


Figure 9: Graph views

VI CONCLUSION

This project presents a scalable, low-power environmental monitoring system for sustainable agriculture. By custom PCBs integrating sensor modules with LoRa communication, GPS, and a cloud-based data pipeline, the system provides real-time insights into soil and environmental conditions. The web-based dashboard offers intuitive visualizations, proactive notifications, and predictive tools powered by machine learning that help users optimize irrigation. The system combines embedded sensors, real-time data processing, and machine learning to support sustainable agriculture by providing users with data-riven insights and suggestions. The solution presented can be easily extended to the ground sensor or airborne sensor aggregation.

Through this work, it is demonstrated that it is possible to build a complete, end-to-end product that integrates embedded sensor modules, real-time data infrastructure, and machine learning to support sustainable agriculture. By using low-cost components and open-source tools, SproutSense shows that advanced environmental monitoring and decision support can be both accessible and effective.

ACKNOWLEDGMENTS

We would like to thank the International Foundation for Telemetering (IFT) for their support. We would also like to express our sincere gratitude to the University of California, Santa Barbara, for providing us with the opportunity to work on a project that allowed us the creative freedom to explore and integrate various disciplines. We would also like to thank Professor Isukapalli, Torin Schlunk, Hunter Larson, and Cameron Barrett for their continuous support, guidance, and expertise throughout the development of this project.

REFERENCES

[1] A., E. G. and Bala, G. J. (2024). IoT and ML-Based Automatic Irrigation System for Smart Agriculture System. *Agronomy Journal*. Accessed: 2025-03-02.

[2] Adafruit Industries (2022). *Adafruit Ultimate GPS: Overview*. Available at: `https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps.pdf`, Accessed: 2024-10-01.

[3] Chan, J., Cohn, M., Kathi, B., Rivas, P. S., and Shrestha, K. (2025). Sproutsense: Real-time environmental monitoring for sustainable agriculture. `https://github.com/BK1031/SproutSense`.

[4] EByte Technology (2019). *E32-915T20D UART Wireless Module User Manual*. Available at: `https://www.rcscomponents.kiev.ua/datasheets/e32-915t20d_usermanual_en_v1_6.pdf`, Accessed: 2025-10-01.

[5] García, L., Parra, L., Jimenez, J. M., Lloret, J., and Lorenz, P. (2020). IoT-based smart irrigation systems: An overview on the recent trends on sensors and IoT systems for irrigation in precision agriculture. Accessed: 2025-03-02.

[6] Kasera, R. K. and Acharjee, T. (2024). A Comprehensive IoT Edge Based Smart Irrigation System for Tomato Cultivation. *Internet of Things*, 28:101356. Accessed: 2025-03-02.

[7] Kaur, A., Bhatt, D. P., and Raja, L. (2023). Soil Moisture, Air Temperature, Humidity, and Motor On/Off Monitoring Data. Accessed: 2025-03-02.

[8] STMicroelectronics (2023). *STM32L476JE Ultra-Low-Power Arm® Cortex®-M4 MCU with FPU, 1MB Flash, 128KB RAM, USB OTG, LCD, Op-Amp, Comparators*. Available at: `https://www.st.com/resource/en/datasheet/stm32l476je.pdf`, Accessed: 2025-10-01.