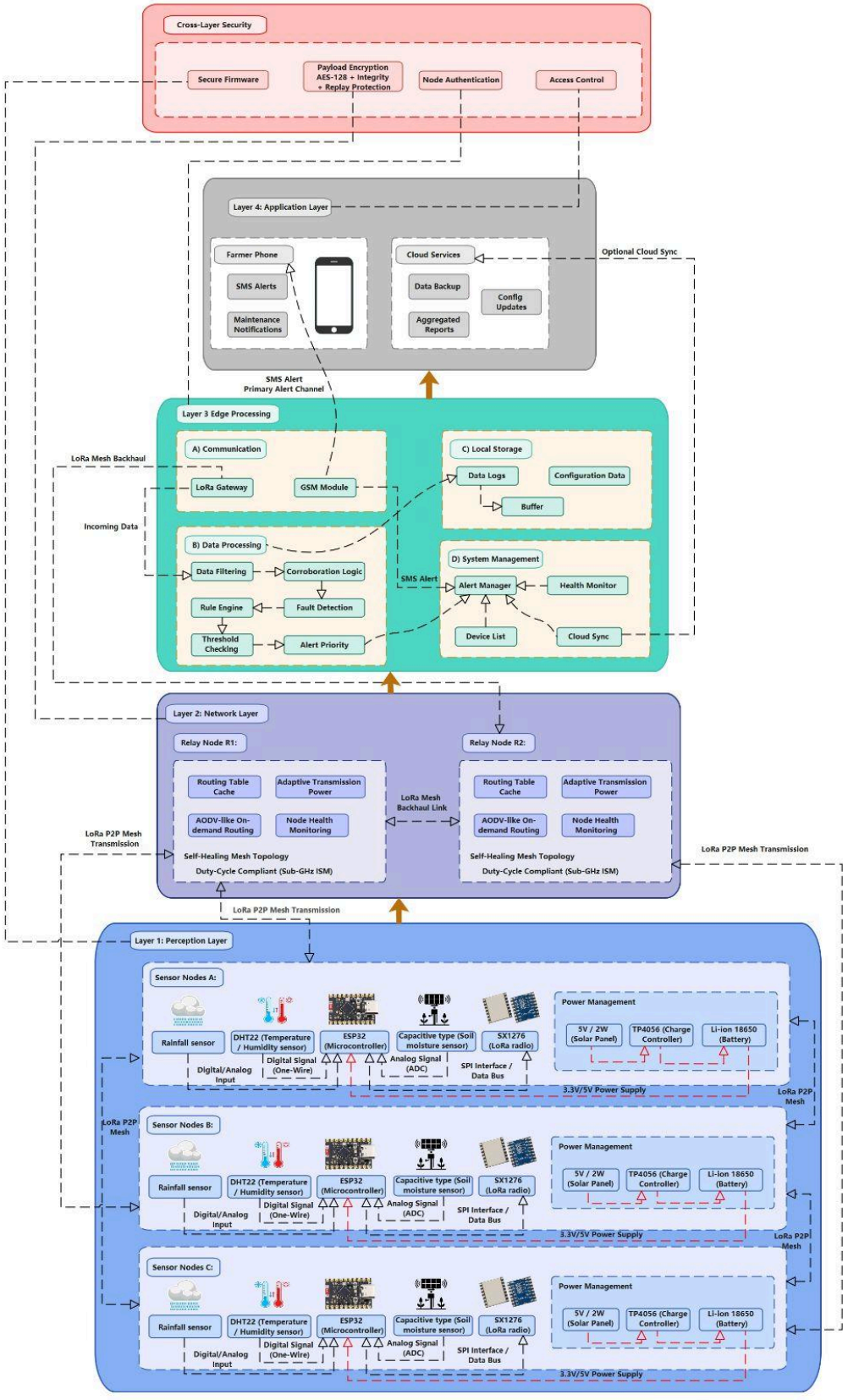
Table of Contents

Table of Contents **2**
Marking Rubric **3**
Detailed Report of the Assigned Tasks **4**
Part 1: System Architecture and Technology Stack **5**
1.1 Detailed System Architecture Diagram 6
1.2 Hardware Component Selection and Justification 7
1.3 Software Platform Selection and Justification 11
1.3.3 Optional Cloud Platform 12
1.4 Communication Protocol Selection and Justification 13
1.5 Data Storage and Management 16
1.5.3 Database Schema 17
Part 2: Algorithm Design and Implementation Details **20**
2.1 Environmental Control Algorithms 20
2.2 Data Processing and Analysis 26
2.3 Mobile App Functionality 28
2.4 Security Considerations 30
References **32**
Presentation Slides The Link **36**

Part 1: System Architecture and Technology Stack



1.1 Detailed System Architecture Diagram

The ROOT architecture follows a four-layer IoT design that enables reliable agricultural monitoring in areas with limited connectivity and power infrastructure. Each layer serves a distinct function, and a cross-layer security architecture (detailed in Section 2.4) provides data confidentiality, integrity, and access control throughout the system. Hardware and protocol specifics are addressed in Sections 1.2 and 1.4 respectively.

Layer 1 — Perception Layer

The Perception Layer comprises distributed sensor nodes deployed across farm plots. Each node measures soil moisture, temperature, humidity, and rainfall using low-power sensors connected to an ESP32 microcontroller [1]. Nodes perform minimal local processing and transmit readings at configurable intervals via LoRa radio. Solar panels and $LiFePO_4$ batteries enable autonomous operation with 72-hour backup without sunlight.

Layer 2 — Network Layer

The Network Layer provides communication between sensor nodes and the village gateway using LoRa physical-layer (PHY) modulation in a peer-to-peer mesh topology [2]. Unlike LoRaWAN's star-of-stars architecture, ROOT implements multi-hop mesh routing based on AODV (Ad-hoc On-demand Distance Vector) [3], enabling self-healing path discovery when nodes fail. Relay nodes extend coverage across large agricultural areas (5–15 km per hop). Payload security is implemented at the application level using AES-128-GCM encryption with message authentication and anti-replay nonces, rather than relying on LoRaWAN's built-in network security stack [4].

Layer 3 — Edge Processing Layer

The village gateway is the central intelligence of the ROOT system. Incoming LoRa packets enter a sequential processing pipeline: (1) data validation and filtering remove erroneous readings (Section 2.2); (2) collective corroboration cross-checks readings against neighbouring nodes (Section 2.1); (3) threshold evaluation applies crop-specific rules to detect drought, frost, and heat stress conditions; (4) alert generation prioritises and formats SMS notifications dispatched via an attached GSM modem. Local storage uses an encrypted SQLite database (Section 1.5). Cloud synchronisation is optional, when internet connectivity

exists, the gateway uploads aggregated summaries and downloads configuration updates via HTTPS [5].

Layer 4 — Application Layer

The Application Layer provides the user-facing interface. SMS messages delivered via GSM constitute the primary alert channel, functioning on any mobile phone without internet or smartphone requirements [6]. An optional mobile application (Section 2.3) offers cooperative administrators advanced monitoring, configuration, and visualisation capabilities. Optional cloud services provide data backup, regional reporting, and remote configuration but are never required for core system operation.

Cross-Layer Security

Security is distributed across all layers rather than concentrated at a single point. Device identity verification occurs at provisioning; payload encryption protects data in transit; node authentication is enforced at the gateway; and role-based access control restricts configuration changes at the application layer. Full details are provided in Section 2.4.

1.2 Hardware Component Selection and Justification

The ROOT hardware platform is designed for low-cost, long-term outdoor deployment in rural agricultural environments without grid electricity. All component selections prioritise low power consumption, compatibility with LoRa P2P mesh communication, durability in field conditions, and availability in developing-region supply chains [7].

1.2.1 Sensor Node Hardware

Table 1. Sensor Node Hardware Specifications

Component	Model	Key Specifications	Typical Power	Est. Cost (USD)	Datasheet / Source
Microcontroller	ESP32-WROOM-32	Dual-core 240 MHz, ADC/SPI/UART, 520 KB SRAM	80 mA active; <10 µA deep sleep	$4–6	[8]

LoRa Transceiver	Semtech SX1276	868/915 MHz, SF7–SF12, SPI interface	120 mA TX; 10 mA RX; 1 µA sleep	$7–10	[9]
Soil Moisture Sensor	Capacitive v1.2	Analogue output, corrosion-resistant (no exposed electrodes)	<5 mA	$2–3	[10]
Temp / Humidity Sensor	DHT22 (AM2302)	Temp ±0.5°C (−40–80°C); RH ±2–5%	2.5 mA measuring	$3–5	[11]
Rainfall Sensor	Tipping Bucket (generic)	0.2 mm/tip resolution; passive reed switch	Passive (0 mA)	$8–12	[12]
Solar Panel	6V / 2W Mini PV	Polycrystalline; ~330 mA at peak	—	$5–8	Vendor spec
Battery	18650 $LiFePO_4$ 3.2V	1500 mAh; >2000 cycle life; thermally stable	—	$4–6	[13]
Charge Controller	TP4056 (with DW01 protection)	Linear Li-ion charger; over-discharge / over-current protection	<1 mA standby	$1–2	[14]

Total estimated cost per sensor node: ~$36 USD.

Key selection rationale:

- ESP32 was chosen over Arduino Uno (no native deep sleep; limited RAM) and STM32 (steeper tooling curve; less community support) because it offers ultra-low deep sleep current (<10 µA), sufficient GPIO/ADC for all sensors, SPI for the LoRa transceiver, mature SDK support via ESP-IDF [15], and a large open-source community. Integrated Wi-Fi/Bluetooth are unused in ROOT but add no cost penalty.

- SX1276 provides LoRa chirp spread spectrum modulation at sub-GHz ISM bands, achieving typical rural line-of-sight range of 2–10 km (up to 15 km under ideal conditions depending on spreading factor, antenna gain, and terrain) [9]. It was chosen over the newer SX1262 due to wider availability, lower cost, and sufficient performance for ROOT's telemetry data rates.
- DHT22 was selected over the BME280 (higher cost $8; I²C bus complexity) and SHT31 ($12) because its ±0.5°C / ±2–5% RH accuracy meets agricultural monitoring requirements at the lowest per-node cost [11]. All outdoor temperature/humidity sensors are mounted inside a Stevenson-style radiation shield to prevent direct solar radiation and rain exposure from affecting readings.
- $LiFePO_4$ battery chemistry was selected over standard Li-ion (NCR18650B) for superior thermal stability (−20°C to 60°C), longer cycle life (>2000 vs ~500 cycles), and reduced fire risk [13].
- TP4056 was chosen for prototype simplicity and cost ($1–2). A production design would upgrade to an MPPT-capable charger (e.g., CN3065) to improve solar harvesting efficiency [16].

1.2.2 Gateway Hardware

Table 3. Gateway Hardware

Component	Model	Function	Est. Cost (USD)
Edge computer	Raspberry Pi 4 Model B (4 GB)	Local data processing, algorithm execution, database host	$60
LoRa transceiver	SX1276-based HAT (single-channel)	P2P mesh packet reception from sensor nodes	$15
GSM module	SIM800L	SMS alert transmission via AT commands; no internet required	$6
Storage	MicroSD card (32 GB)	Local database, logs, firmware images (Section 1.5)	$8

Gateway total			$89

The LoRa HAT is a single-channel transceiver for P2P mesh reception, not a multi-channel LoRaWAN concentrator. The Raspberry Pi runs all edge algorithms (Section 2.1), local storage (Section 1.5), and SMS dispatch.

1.2.4 Power Budget Analysis

Energy autonomy is critical for nodes deployed in areas without grid electricity. The following power budget assumes the default 15-minute sampling interval with a 2-second LoRa transmission per cycle.

Table 4. Power Budget per Sampling Cycle (15 minutes = 900 seconds)

Phase	Duration (s)	Current (mA)	Charge (mAs)
Deep sleep (ESP32 + SX1276 sleep)	893	0.01	8.93
Wake + sensor read	3	90	270
LoRa TX (SF7, 14 dBm)	2	120	240
LoRa RX window	2	10	20
Total per cycle	900	—	539 mAs

Average current = 539 / 900 ≈ 0.60 mA

Runtime without solar (1500 mAh $LiFePO_4$): 1500 / 0.60 ≈ 2500 hours ≈ 104 days

This comfortably exceeds the 72-hour design target. With solar recharging (~330 mA peak for several hours daily), nodes operate indefinitely under normal conditions [16].

1.2.5 Sensor Placement Strategy

Sensor	Placement	Rationale
Soil moisture	15–20 cm depth, within crop root zone	Captures moisture availability at root level [17]

Temperature / Humidity	1.5 m above ground, inside radiation shield	Standard meteorological height; shield prevents solar/rain interference
Rain gauge	1 m above ground, clear of obstructions	Unobstructed rainfall capture; level mounting for accuracy
Relay nodes	Elevated positions every 500–1000 m (terrain-dependent)	Maximises LoRa line-of-sight between sensor clusters and gateway
Gateway	Central location with GSM coverage	Minimises average hop count; ensures SMS delivery capability

1.3 Software Platform Selection and Justification

ROOT's software stack is divided between constrained sensor nodes and the village gateway. The design prioritises lightweight, proven technologies that support offline-first operation, with no component depending on continuous cloud connectivity [18].

1.3.1 Node Firmware

Sensor nodes run firmware developed using ESP-IDF (Espressif IoT Development Framework), Espressif's official C-based SDK for the ESP32 [19]. ESP-IDF was selected over the Arduino framework because it provides direct hardware abstraction layer (HAL) access, fine-grained power management (deep sleep wake sources, ULP co-processor control), FreeRTOS task scheduling, and mature OTA update support with dual-partition boot, all essential for battery-powered field nodes operating unattended for months.

1.3.2 Gateway Software Stack

The village gateway runs on a Raspberry Pi 4 Model B with the following software stack:

Table 5. Gateway Software Stack

Layer	Technology	Purpose
Operating system	Raspberry Pi OS Lite (Debian-based)	Lightweight headless Linux; no GUI overhead

Runtime	Python 3.11	Algorithm execution, LoRa packet handling, SMS dispatch
LoRa interface	pyLoRa / spidev	SPI communication with SX1276 HAT
Rules engine	Custom Python modules	Threshold evaluation, collective corroboration (Section 2.1)
Local database	SQLite 3 + SQLCipher	Encrypted time-series storage (Section 1.5)
SMS service	python-gsmmodem / AT commands	Serial interface to SIM800L for alert delivery
Sync service	requests (HTTPS)	Opportunistic upload to InfluxDB Cloud HTTP API (Section 1.5)
Service management	systemd	Auto-start, watchdog restart, logging
Encryption	OpenSSL (via Python cryptography library)	AES-128-GCM payload decryption; TLS for cloud sync

1.3.3 Optional Cloud Platform

Table 6. Cloud Platform Comparison

Platform	Advantages	Limitations	ROOT Suitability
InfluxDB Cloud (Selected)	Time-series native; simple HTTP write API with token auth; built-in dashboards; free tier (30-day retention)	Requires internet for sync; free tier storage limits	Best fit: lightweight integration, no broker dependency, purpose-built for sensor data [20]
AWS IoT Core + Timestream	Managed device auth (X.509 certs); scalable rule engine; broad ecosystem	Complex setup; multiple service dependencies; cost scales with message volume; overkill for optional sync	Over-engineered for ROOT's optional cloud role

Self-hosted MQTT + InfluxDB	Full control; no vendor lock-in	Requires server maintenance; reliability burden; hosting cost	Impractical for rural deployment support model

Decision: InfluxDB Cloud was selected because ROOT requires cloud only for optional backup and reporting, not real-time processing. The HTTP write API allows the gateway to push batched data directly without maintaining persistent MQTT connections or complex broker infrastructure, aligning with the intermittent-connectivity model (Section 1.5).

1.3.4 Mobile App Framework

The optional mobile application (Section 2.3) is built using Flutter [21], enabling a single codebase for Android and iOS deployment. Flutter was selected over React Native for superior performance on low-spec Android devices common in target regions, and over native development for cost efficiency. The app communicates with the gateway/cloud via RESTful HTTPS endpoints secured with JWT authentication.

1.4 Communication Protocol Selection and Justification

ROOT's communication requirements are defined by its operating environment: long range across dispersed farm plots, very low power consumption for solar-powered nodes, and zero dependence on external network infrastructure [22]. This section details the protocol selection for each communication segment and the custom protocol stack implemented over LoRa PHY.

1.4.1 Communication Technology Comparison

Table 7. Communication Technology Comparison

Technology	Range	Power	Infrastructure Required	Data Rate	ROOT Usage
Zigbee (802.15.4)	100–300 m	Low	Coordinator + mesh	250 kbps	✗ - Range insufficient for farms
NB-IoT	Several km	Medium	Cellular base station	250 kbps	✗ - Requires telecom infrastructure

LoRaWAN	2–15 km	Very low	Gateway + network server	0.3–50 kbps	✗ - Depends on network server infra
LoRa P2P mesh (Selected)	2–15 km per hop	Very low	No external infra; P2P mesh + edge gateway	0.3–50 kbps	✓ - Offline-first; self-healing mesh

Decision: LoRa PHY with a custom P2P mesh protocol was selected because it provides kilometre-scale range at microamp sleep currents without any external infrastructure - no cellular towers, no LoRaWAN network servers, no internet backhaul [2]. ROOT implements its own lightweight protocol stack on top of the raw LoRa modulation, as described below.

1.4.2 ROOT Protocol Stack

Table 8. ROOT Layered Protocol Stack

Layer	Implementation	Function
Physical	LoRa (Semtech SX1276, CSS modulation)	Long-range, low-power RF transmission at 868/915 MHz ISM band [9]
MAC	Custom duty-cycled MAC	Scheduled transmit windows with random backoff; compliant with regional duty-cycle regulations (e.g., 1% in EU 868 MHz)
Network	AODV-like on-demand routing [23]	Multi-hop mesh; route discovery only when needed; self-healing around failed nodes; relay nodes forward packets toward gateway
Security	AES-128-GCM	Payload encryption + authentication tag + monotonic counter for anti-replay (Section 2.4)
Application	Custom compact binary telemetry	Sensor readings, alerts, configuration, and health reports in CBOR-encoded payloadsll s

AODV-like routing: ROOT implements a simplified Ad-hoc On-demand Distance Vector protocol [23] where route discovery (RREQ/RREP) is initiated only when a node needs to transmit and has no cached route to the gateway. Route entries expire after a configurable timeout, and link failures trigger local repair or re-discovery. This on-demand approach

minimises control traffic overhead, critical for duty-cycle–constrained LoRa channels. Relay nodes are stateless forwarders that maintain small routing tables (typically <20 entries for a 50-node network).

1.4.3 Application Message Format

Sensor telemetry is encoded using CBOR (Concise Binary Object Representation) [24] to minimise payload size and LoRa airtime. The following fields constitute a standard telemetry message:

Table 9. Sensor Telemetry Message Format

Field	Size (bytes)	Description
msg_type	1	Message type: 0x01=telemetry, 0x02=alert, 0x03=join, 0x04=ack, 0x05=config
proto_ver	1	Protocol version (currently 0x01)
node_id	4	Unique node identifier
timestamp	4	UTC epoch seconds
soil_moisture	2	Soil moisture (% × 100, unsigned)
temperature	2	Temperature (°C × 100, signed)
humidity	2	Relative humidity (% × 100, unsigned)
rainfall	2	Rainfall since last report (mm × 100)
battery_pct	1	Battery level (0–100%)
flags	1	Bit flags: alert active, fault detected, low battery

counter	4	Monotonic anti-replay counter (per node)
hop_count	1	Incremented at each relay (mesh debugging)
auth_tag	4	Truncated AES-128-GCM authentication tag
Total	~29 bytes	

Overhead estimate: At SF7/125 kHz, a 29-byte payload results in ~50 ms airtime [9], well within the 1% duty-cycle budget at 15-minute transmission intervals.

MAC coverage: The GCM authentication tag is computed over the entire header + payload + counter, ensuring both integrity and authenticity. The monotonic counter prevents replay of captured packets (Section 2.4).

1.4.4 Gateway-to-Cloud Communication

When internet connectivity is available, the gateway synchronises data to InfluxDB Cloud using HTTPS (TLS 1.2+) with token-based authentication (Section 1.5). SMS alerts are dispatched via the SIM800L GSM modem using serial AT commands over the cellular voice/SMS network , this path requires no internet connectivity [25].

Segment	Protocol	Security	Dependency
Node → Gateway	LoRa P2P mesh	AES-128-GCM + counter	None (fully offline)
Gateway → Farmer	SMS via GSM (AT commands)	GSM network encryption	GSM signal only
Gateway → Cloud	HTTPS (REST API)	TLS 1.2+ with bearer token	Internet (optional)

1.5 Data Storage and Management

ROOT implements a hybrid storage architecture that prioritises local persistence over cloud dependency. The village gateway serves as the primary data store, maintaining all sensor readings, alerts, and configuration locally. Cloud synchronisation to InfluxDB Cloud is an

optional enhancement activated only when internet connectivity is available [20]. This design ensures that data logging, alert generation, and audit trails continue uninterrupted during extended offline periods.

Design principles:

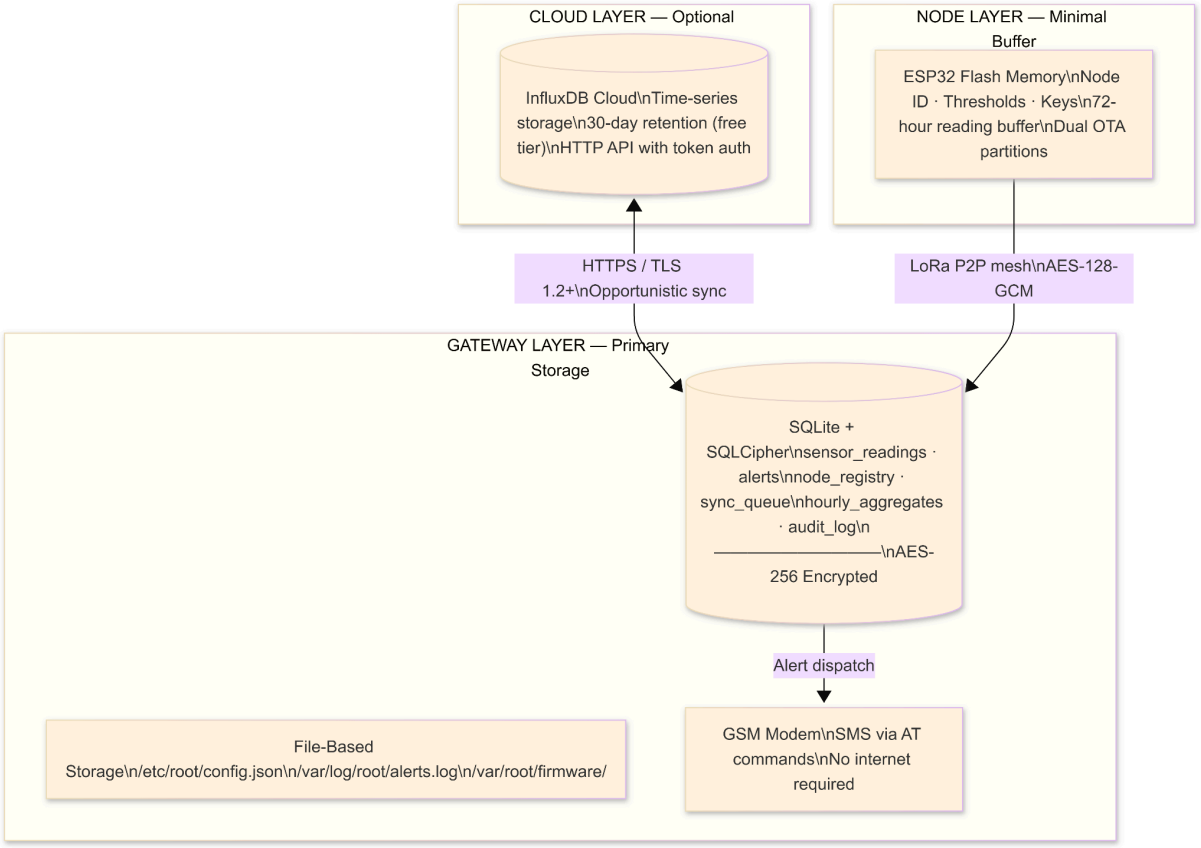
I. local-first persistence for uninterrupted operation;

II. time-series–optimised schema for efficient sensor queries;

III. encryption at rest and in transit;

IV. store-and-forward queuing for intermittent cloud sync.

1.5.1 Storage Architecture

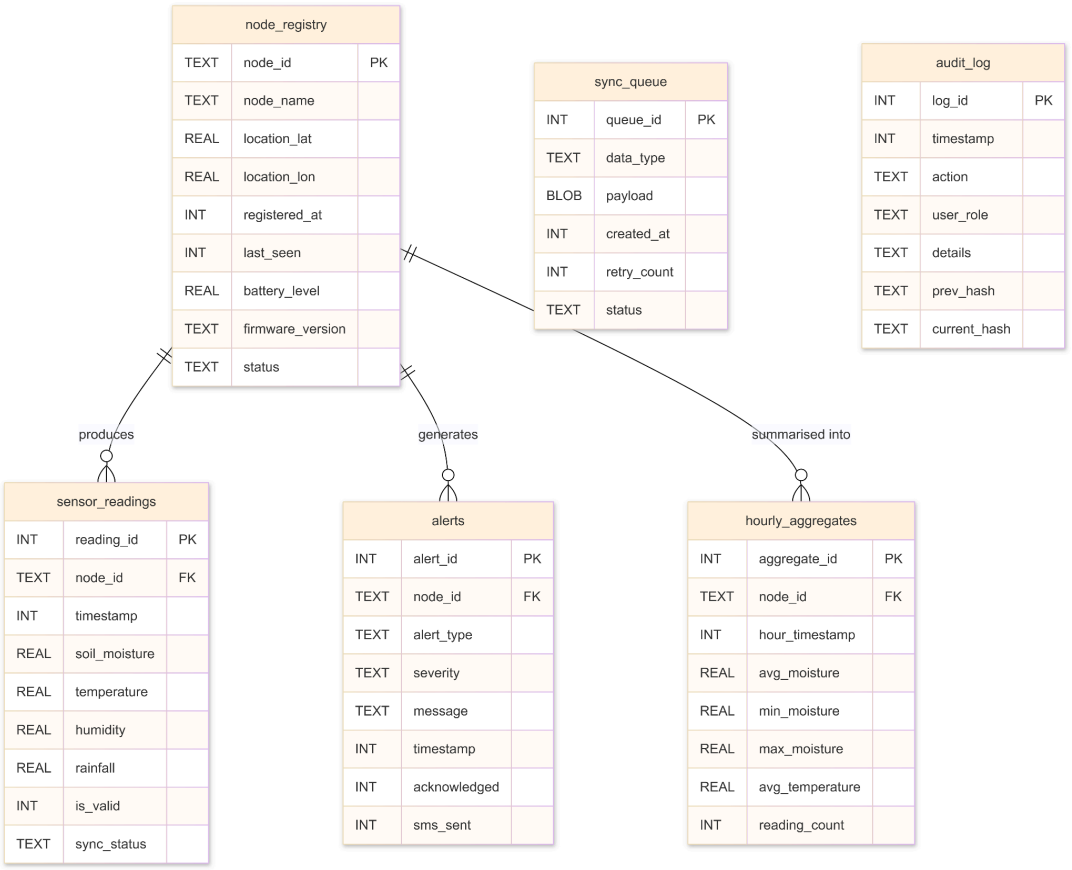


1.5.3 Database Schema

SQLite was selected for gateway storage due to its zero-configuration operation, single-file architecture, and minimal resource footprint, ideal for an embedded Raspberry Pi deployment

[26]. SQLCipher provides transparent AES-256 full-database encryption with keys derived via PBKDF2 using a unique per-installation salt [27].

Entity-Relationship Diagram:



1.5.4 Data Retention Policy

Data Type	Local (Gateway)	Cloud (InfluxDB)	Notes
Raw sensor readings	7 days	30 days	Aggregated before deletion
Hourly aggregates	90 days	1 year	Computed from raw data before purge
Daily aggregates	2 years	5 years	Long-term trend analysis
Alerts	1 year	5 years	Audit trail
Audit logs	Permanent	Permanent	Hash-chained; tamper-evident

Storage calculation (200-node deployment): Raw data per node ≈ 12 KB/day × 7 days = 84 KB. Aggregates per node ≈ 2 KB/day × 730 days = 1.5 MB. Total per node over 2 years ≈ 1.6 MB. Fleet total ≈ 320 MB, comfortably fits a 32 GB SD card with ample headroom for logs and firmware images.

1.5.5 Data Security Summary

Data Path	Encryption	Method
At rest (gateway DB)	AES-256	SQLCipher; PBKDF2-derived key with per-installation salt [27]
In transit (LoRa)	AES-128-GCM	Pre-shared keys; authentication tag + anti-replay nonce (Section 2.4)
In transit (cloud)	TLS 1.2+	Certificate-based authentication to InfluxDB Cloud [20]
SMS delivery	GSM network encryption	AT commands via serial; no internet path

Hash-chained audit logging (SHA-256) records all configuration changes, alert dispatches, and administrative actions. Tampering with any entry breaks the chain, enabling integrity verification (Section 2.4).

1.5.6 Synchronisation Strategy

The gateway implements store-and-forward synchronisation:

1. Queue: All outbound data (aggregated readings, alerts) is inserted into the sync_queue table with status pending.
2. Connectivity check: The gateway periodically issues an HTTPS HEAD request to the InfluxDB Cloud endpoint.
3. Batch upload: When connected, pending items are uploaded in batches of 100 records via InfluxDB's HTTP write API (token-authenticated) [20].
4. Retry logic: Failed items retry up to 3 times with exponential backoff (30 s, 120 s, 480 s). Items exceeding retries are logged and flagged for manual review.

5. Cleanup: Successfully synced items are marked complete; raw data is pruned per the retention policy.

Cloud sync uses InfluxDB Cloud's HTTP API directly, avoiding complex MQTT broker dependencies and simplifying the deployment stack.

1.5.7 Backup and Recovery

Backup Type	Frequency	Retention	Trigger
Full database snapshot	Daily (03:00)	7 rolling copies	Scheduled cron job
Configuration backup	On change	30 versions	Event-driven
Pre-update backup	Before OTA	Until successful boot	OTA trigger

Recovery priority: Local backup → Cloud backup (if synced) → Fresh initialisation with node re-registration.

Part 2: Algorithm Design and Implementation Details

2.1 Environmental Control Algorithms

ROOT implements five core algorithms executed entirely at the edge gateway, requiring no cloud connectivity for real-time decision-making [28]. Each algorithm processes locally filtered sensor data (Section 2.2) and generates actionable alerts delivered via SMS.

Algorithm	Purpose	Alert Type	Trigger Condition
Irrigation Trigger	Detect drought conditions	Drought Alert	Sustained low moisture across corroborated nodes
Frost Warning	Predict frost before crop damage	Frost Alert	Temperature decline rate and predictive regression
Heat Stress Detection	Identify prolonged excessive heat	Heat Alert	Cumulative degree-hours above stress threshold

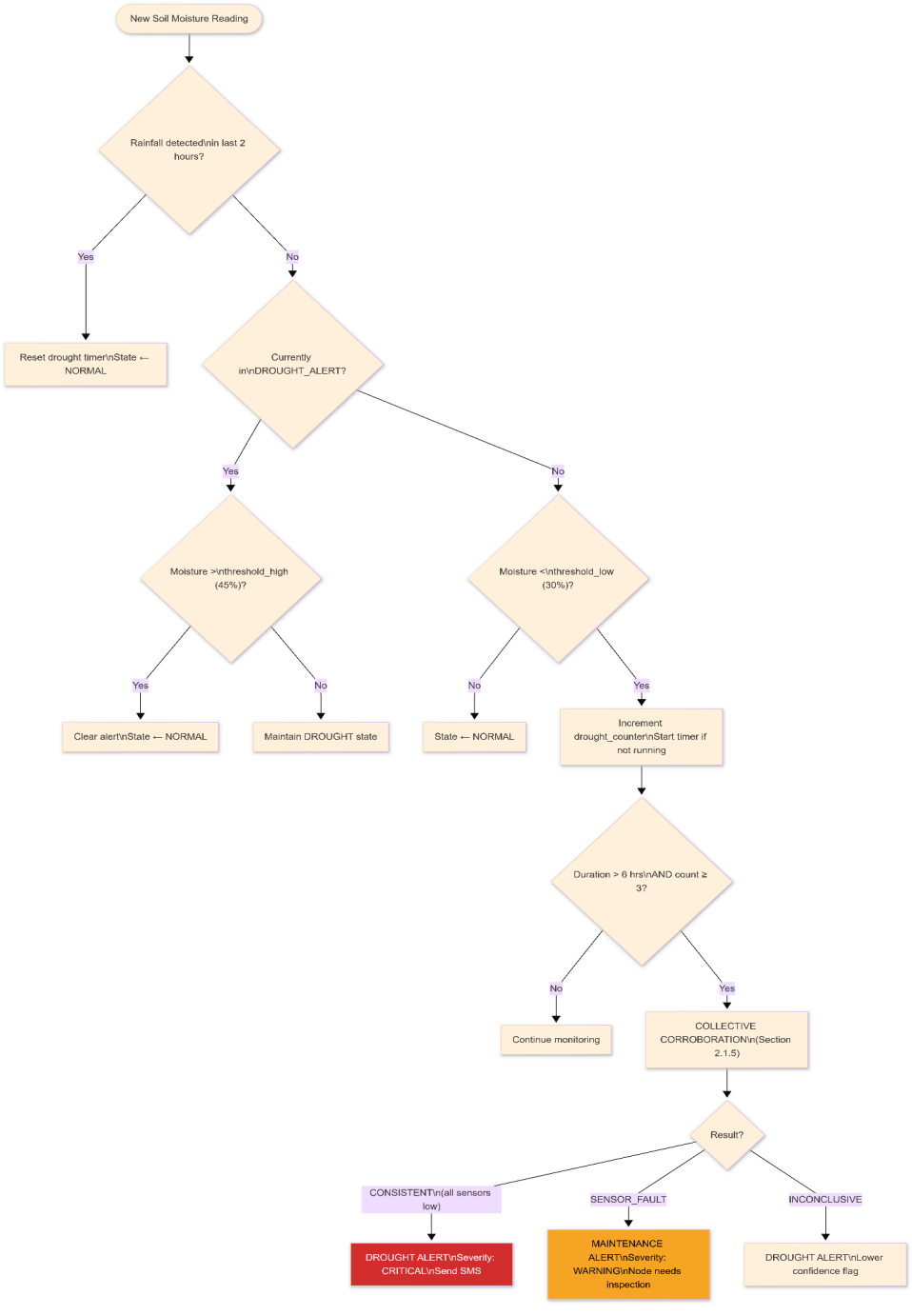
Collective Corroboration	Cross-validate sensor readings	Fault / Confirmed	Statistical deviation from neighbour consensus
Adaptive Sampling	Optimise node power consumption	Config Update	Battery level and environmental conditions

2.1.1 Irrigation Trigger Algorithm

This algorithm monitors soil moisture and generates drought alerts when moisture drops below crop-specific thresholds for a sustained period. Hysteresis (separate low/high thresholds) prevents rapid alert oscillation, and a rainfall lockout period suppresses false triggers after precipitation [17].

Parameters:

Parameter	Default	Description
threshold_low	30%	Below this triggers evaluation
threshold_high	45%	Above this clears drought state
sustained_duration	6 hours	Required duration below threshold
rainfall_lockout	2 hours	Suppress evaluation after detected rain



Key logic: When corroboration returns CONSISTENT and all readings (target + neighbours) are below threshold, this confirms a genuine drought event. SENSOR_FAULT is returned only when the target sensor deviates from neighbours that show normal conditions.

2.1.2 Frost Warning Algorithm

Frost causes instantaneous crop damage; early prediction is therefore critical. This algorithm uses least-squares linear regression on the most recent 2-hour temperature window (~8

readings at 15-minute intervals) to estimate time-to-frost and calculate a probability score [29].

Parameters:

Parameter	Default	Description
frost_threshold	2°C	Below this causes crop damage
prediction_horizon	3 hours	Maximum lookahead window
min_decline_rate	1.5°C/hr	Minimum rate to trigger evaluation
active_hours	18:00–06:00	Night hours when frost monitoring is active

2.1.3 Heat Stress Detection Algorithm

Unlike frost, heat damage results from prolonged exposure rather than an instantaneous threshold breach. This algorithm tracks cumulative heat using degree-hours above the crop stress threshold [17]. It uses actual elapsed time between readings (delta_hours) rather than assuming fixed intervals, ensuring correct accumulation even when adaptive sampling (Section 2.1.6) changes the reading frequency.

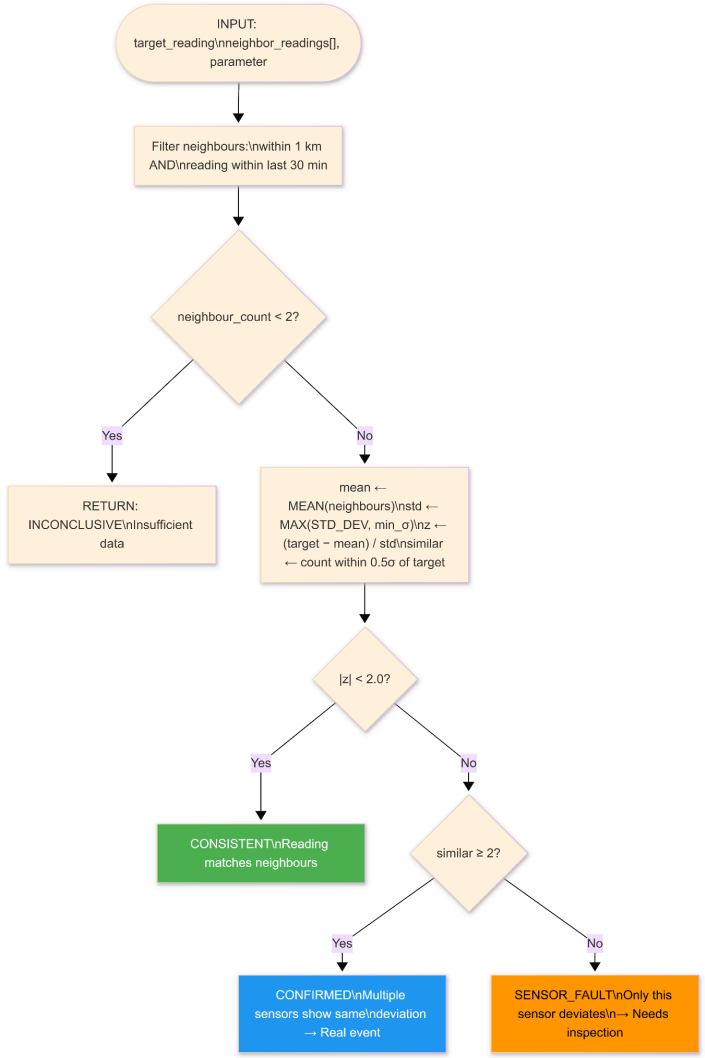
2.1.4 Collective Corroboration Algorithm

This is ROOT's key innovation, cross-validating a sensor reading against geographically proximate neighbours to distinguish real environmental events from sensor faults [23]. The algorithm uses the z-score statistic to quantify deviation, where $z = (x - \mu) / \sigma$; x is the target reading, μ the neighbour mean, and σ the neighbour standard deviation. A $|z| > 2.0$ indicates statistically significant deviation at ~95% confidence [30].

Parameter-specific minimum standard deviations (floor values to avoid division-by-zero when neighbours read identically):

Parameter	Min σ	Deviation Threshold
Soil moisture	2.0%	15%

Temperature	0.5°C	10%
Humidity	2.0% RH	15%



Pseudocode:

```
FUNCTION corroborate(target, neighbours[], parameter):
    valid ← [n for n in neighbours if n.distance < 1km AND n.age < 30min]
    IF len(valid) < 2: RETURN INCONCLUSIVE

    values ← [n.value for n in valid]
    mean   ← MEAN(values)
    std    ← MAX(STD_DEV(values), MIN_STD[parameter])
    z      ← (target.value - mean) / std
    similar ← COUNT(v in values WHERE |v - target.value| < 0.5 × std)

    IF |z| < 2.0:       RETURN CONSISTENT
    ELSE IF similar >= 2: RETURN CONFIRMED
    ELSE:                 RETURN SENSOR_FAULT
```

Example scenarios:

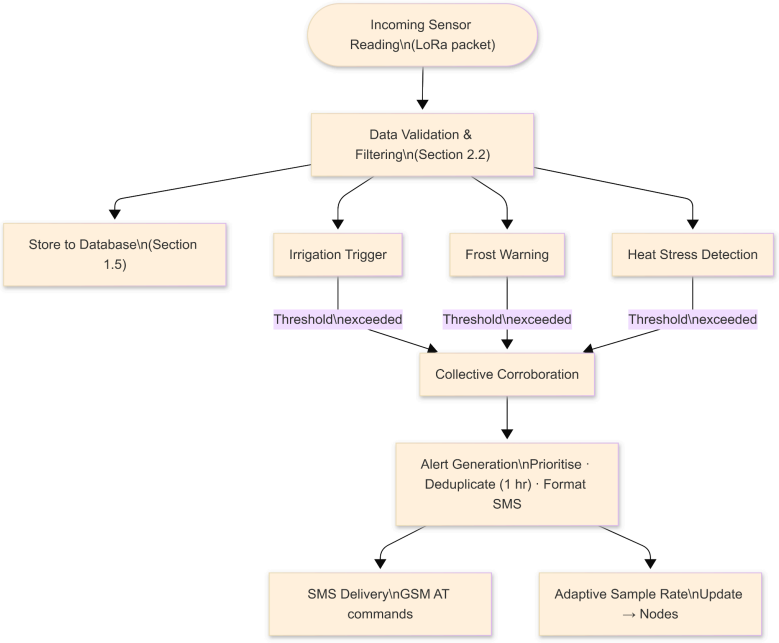
Scenario	Target	Neighbours	z-score	Similar	Result
Normal conditions	45%	43%, 46%, 44%	0.5	3	CONSISTENT
Real drought (all low)	22%	24%, 21%, 23%	−0.3	3	CONSISTENT → Drought alert
Faulty sensor	15%	45%, 43%, 47%	−3.2	0	SENSOR_FAULT
Localised dry patch	18%	20%, 19%, 45%	−1.8	2	CONFIRMED

2.1.5 Adaptive Sampling Rate Algorithm

This algorithm dynamically adjusts sensor reading frequency based on battery level and environmental conditions, extending node lifetime during stable periods while increasing resolution during developing events.

2.1.6 Algorithm Integration Pipeline

All algorithms execute within a unified processing pipeline at the gateway. Data validation (Section 2.2) feeds filtered readings into the three environmental algorithms in parallel. When any algorithm's threshold condition is met, collective corroboration is invoked before alert generation.



2.1.8 Performance Summary

Algorithm	Time Complexity	Execution Time	Memory
Irrigation Trigger	O(1)	< 1 ms	Minimal
Frost Warning	O(n), n = lookback readings	< 5 ms	~1 KB
Heat Stress Detection	O(1)	< 1 ms	Minimal
Collective Corroboration	O(m), m = neighbours	< 10 ms	~2 KB
Adaptive Sampling	O(1)	< 1 ms	Minimal

2.2 Data Processing and Analysis

ROOT's data processing pipeline is split between the edge gateway (primary) and optional cloud services (enhancement), consistent with the system's offline-first architecture. All real-time filtering, validation, and alert-triggering logic executes locally at the gateway; cloud processing is limited to long-term analytics and reporting when connectivity permits [28].

2.2.1 Data Filtering and Cleaning

Raw sensor readings are susceptible to environmental noise, transient electrical interference, and sensor drift [31]. The gateway applies three sequential filtering steps before data enters the decision algorithms (Section 2.1):

1. Range validation: Readings outside physically plausible bounds (e.g., soil moisture < 0% or > 100%, temperature < −20°C or > 60°C) are flagged invalid and excluded.
2. Spike detection: If a reading deviates by more than ±20% from the previous valid reading within one sampling interval, it is flagged as a transient spike and replaced with the last valid value.
3. Moving average smoothing: A 3-point simple moving average is applied to reduce high-frequency noise while preserving genuine trends.

```
FUNCTION filter_reading(new_val, prev_valid, history[]):
    IF new_val outside [RANGE_MIN, RANGE_MAX]:
        RETURN prev_valid, flag=INVALID
    IF |new_val - prev_valid| > 0.20 × prev_valid:
        RETURN prev_valid, flag=SPIKE
    history.append(new_val)
    smoothed = MEAN(history[last 3])
    RETURN smoothed, flag=VALID
```

All filtered data is stored locally (Section 1.5). Collective corroboration (Section 2.1.5) provides an additional cross-node validation layer beyond single-node filtering.

2.2.2 Data Aggregation (Edge + Cloud)

The gateway computes hourly aggregates (mean, minimum, maximum, reading count) per node and stores them in the hourly_aggregates table (Section 1.5). This reduces long-term storage requirements by 90% compared to retaining raw 15-minute readings [20].

When cloud connectivity is available, the gateway uploads daily aggregates to InfluxDB Cloud, enabling:

- Weekly and monthly trend comparisons across crop seasons
- Community-level statistics (anonymised mean moisture, temperature ranges)
- Water usage efficiency metrics over time

2.2.3 Data Visualisation (Cloud Dashboard + Mobile App)

The optional mobile application (Section 2.3) renders aggregated data through standard visualisation components:

- Line charts: Soil moisture and temperature over time, with alert-event markers overlaid
- Status indicators: Colour-coded current conditions per node (green/amber/red)
- Historical comparison: Current-season readings plotted against previous-season baselines

All visualisations consume pre-aggregated data, no raw data transfer to the app is required, minimising bandwidth use over intermittent connections.

2.3 Mobile App Functionality



The ROOT mobile application is an optional monitoring and management tool intended for cooperative administrators and extension officers. SMS remains the primary alert delivery mechanism for farmers, functioning on any basic mobile phone without internet (Section 1.1). The app supplements SMS by offering configuration, visualisation, and historical review capabilities over a data-rich interface.

2.3.1 Real-time Monitoring

The dashboard displays current field conditions received from the gateway: soil moisture, air temperature, humidity, and rainfall per node. Colour-coded status indicators (green/amber/red) enable rapid assessment. The active crop profile (e.g., "Maize - Plot 3") and system operational status are shown at the top. A scrollable activity log at the bottom records recent events such as irrigation triggers and low-moisture warnings.

2.3.2 Remote Control

Administrators can manually trigger optional actuator integrations (e.g., irrigation valves) during emergencies or maintenance. This feature is disabled by default and restricted to administrator-role accounts (see RBAC in Section 2.4). ROOT's core design is monitoring-and-alerting; actuation is an optional add-on.

2.3.3 Scheduling

The scheduling interface allows administrators to define automated irrigation windows (e.g., morning and evening) and set soil-moisture threshold triggers. Schedules are uploaded to the gateway during the next sync, where they execute locally regardless of app connectivity.

2.3.4 Notifications and Alerts

The in-app notification centre displays a log of all alerts generated by the gateway, colour-coded by priority (critical = red, warning = amber, informational = blue). While the app provides a convenient alert history, critical alerts are always dispatched via SMS first to ensure delivery even when the app is unreachable.

2.3.5 Personalisation

Administrators select crop profiles, maize, rice, cassava, mixed vegetables, which automatically configure recommended soil-moisture thresholds and irrigation schedules based on agronomic guidelines [17]. Profiles can be customised per plot, enabling a single ROOT deployment to serve diversified farming operations.

2.3.6 Technical Implementation

Component	Technology	Justification
Framework	Flutter [21]	Single codebase for Android + iOS; suitable for low-spec devices
API layer	RESTful HTTPS	Secured with JWT token authentication; stateless
Local cache	SQLite [26]	Offline access to last-synced data when connectivity drops
Notifications	Push (FCM) + SMS fallback	Push for app users; SMS for all farmers regardless

Sync model	Pull-based polling	App requests updates from gateway/cloud; avoids persistent connections

2.4 Security Considerations

ROOT operates in physically exposed outdoor environments using wireless long-range radio communication between distributed nodes and a central gateway. This exposure creates attack surfaces including physical tampering, wireless eavesdropping, replay attacks, rogue node injection, and unauthorised configuration access [33]. Because deployments use low-power constrained devices with intermittent connectivity, security mechanisms must be lightweight yet effective. ROOT addresses this through a cross-layer protection strategy where each architectural layer implements independent controls [34].

2.4.1 Per-Layer Security Controls

Layer	Threats	Controls	Residual Risk
Perception (Nodes)	Physical tampering, firmware modification, rogue node injection	Unique device ID assigned at provisioning; firmware integrity checks; tamper-evident enclosures	Physical access in remote fields remains possible
Network (LoRa Mesh)	Eavesdropping, replay attacks, packet injection	AES-128-GCM payload encryption; CMAC integrity tag; monotonic counter-based anti-replay nonce per node [35]	Metadata (timing, packet size) remains observable
Edge (Gateway)	Unauthorised data injection, configuration tampering	Challenge-response node authentication using pre-shared keys; encrypted local database (SQLCipher AES-256); hash-chained audit logs	Gateway itself is a high-value target
Application (SMS/App)	Unauthorised threshold changes, data exfiltration	Role-based access control (Table 1); server-side encryption for optional cloud data	SMS channel relies on GSM network security

Table 1: Role-Based Access Control

Role	View Data	Receive Alerts	Modify Thresholds	Manage Nodes	Firmware Update
Farmer	Own plot	✓ (SMS)	✗	✗	✗
Cooperative Admin	All plots	✓ (SMS + App)	✓	✓	✗
Technician	System diagnostics	✓ (App)	✗	✓	✓

2.4.2 Device Authentication

Before a node may submit data, it completes a challenge-response handshake with the gateway. The gateway sends a random nonce; the node computes an HMAC-SHA256 response using its pre-shared key. The gateway verifies the response against its device registry. Only registered node IDs are accepted [36]. This prevents rogue nodes from injecting false environmental data.

2.4.3 Key Management

Pre-shared keys are generated and installed during device provisioning. Keys are stored in the ESP32's secure flash partition. Periodic key rotation is supported through the gateway's configuration channel. Compromised keys can be revoked per-node without affecting the wider network [35].

2.4.4 Secure Firmware Updates

ROOT supports over-the-air (OTA) firmware updates staged through the gateway. Firmware images are digitally signed (Ed25519); the gateway verifies the signature before installation. Dual-partition boot on ESP32 enables rollback if the new firmware fails health checks. Due to LoRa duty-cycle and bandwidth constraints, full firmware images are delivered during scheduled maintenance windows rather than continuous OTA; incremental delta patches are used where possible [37]. Update frequency is rate-limited to prevent abuse.

2.4.5 Audit Logging

All security-relevant events, authentication attempts, configuration changes, alert dispatches, permission modifications, are recorded in a hash-chained audit log (SHA-256). Each entry includes the hash of the previous entry, creating a tamper-evident chain. If any entry is modified, the chain breaks, enabling integrity verification during maintenance inspections (Section 1.5).

References

[1] V. K. Quy et al., "IoT-Enabled Smart Agriculture: Architecture, Applications, and Challenges," Applied Sciences, vol. 12, no. 7, p. 3396, Mar. 2022, doi: 10.3390/app12073396.

[2] A. W. L. Wong, S. L. Goh, M. K. Hasan, and S. Fattah, "Multi-Hop and Mesh for LoRa Networks: Recent Advancements, Issues, and Recommended Applications," ACM Computing Surveys, vol. 56, no. 6, pp. 1–43, Dec. 2023, doi: 10.1145/3638241.

[3] S. R. Das, C. E. Perkins, and E. M. Belding-Royer, "Ad hoc On-Demand Distance Vector (AODV) Routing," IETF RFC 3561, Jul. 2003. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3561.html

[4] K.-L. Tsai, Y.-L. Huang, F.-Y. Leu, I. You, Y.-L. Huang, and C.-H. Tsai, "AES-128 Based Secure Low Power Communication for LoRaWAN IoT Environments," IEEE Access, vol. 6, pp. 45325–45334, 2018, doi: 10.1109/ACCESS.2018.2852563.

[5] P. Yu, F. Teng, W. Zhu, C. Shen, Z. Chen, and J. Song, "Cloud–edge–device collaborative computing in smart agriculture: architectures, applications, and future perspectives," Frontiers in Plant Science, vol. 16, Oct. 2025, doi: 10.3389/fpls.2025.1668545.

[6] M. Rahaman, C.-Y. Lin, P. Pappachan, B. B. Gupta, and C.-H. Hsu, "Privacy-Centric AI and IoT Solutions for Smart Rural Farm Monitoring and Control," Sensors, vol. 24, no. 13, p. 4157, Jan. 2024, doi: 10.3390/s24134157.

[7] E. Effah, G. Ghartey, J. K. Aidoo, and O. Thiare, "Hardware development and evaluation of multihop cluster-based agricultural IoT based on Bluetooth low-energy and LoRa communication technologies," Sensors, vol. 24, no. 18, p. 6113, 2024.

[8] Espressif Systems, "ESP32-WROOM-32 datasheet," 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[9] Semtech Corporation, "SX1276/77/78/79 datasheet," 2020. [Online]. Available: https://www.semtech.com/uploads/documents/sx1276.pdf

[10] E. López et al., "A low-power IoT device for measuring water table levels and soil moisture to ease increased crop yields," Sensors, vol. 22, no. 18, p. 6840, 2022, doi: 10.3390/s22186840.

[11] Aosong Electronics, "AM2302/DHT22 digital temperature and humidity sensor datasheet," 2023. [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[12] Y. Hu et al., "Tipping-bucket self-powered rain gauge based on triboelectric nanogenerators for rainfall measurement," Nano Energy, vol. 98, p. 107234, 2022, doi: 10.1016/j.nanoen.2022.107234.

[13] O. Lazarenko et al., "Determination of thermal physical characteristics of the Panasonic NCR18650b lithium-ion power supply element," Proc. Latvian Acad. Sci., Sect. B, vol. 78, no. 5–6, pp. 372–379, 2024, doi: 10.2478/prolas-2024-0047.

[14] NanJing Top Power, "TP4056 1A standalone linear Li-ion battery charger datasheet." [Online]. Available: https://datasheet.lcsc.com/lcsc/1811141815_TOPPOWER-Nanjing-Extension-Microelectronics-TP4056_C16581.pdf

[15] M. Babiuch, P. Foltynek, and P. Smutny, "Using the ESP32 microcontroller for data processing," in Proc. 20th Int. Carpathian Control Conf. (ICCC), 2019, doi: 10.1109/CarpathianCC.2019.8765944.

[16] A. Iqbal and T. Iqbal, "Low-cost and secure communication system for remote micro-grids using AES cryptography on ESP32 with LoRa module," in Proc. IEEE Electr. Power Energy Conf. (EPEC), 2018, doi: 10.1109/EPEC.2018.8598380.

[17] R. G. Allen, L. S. Pereira, D. Raes, and M. Smith, Crop Evapotranspiration: Guidelines for Computing Crop Water Requirements, FAO Irrigation and Drainage Paper 56. Rome: FAO, 1998.

[18] A. Pawar and S. B. Deosarkar, "IoT-based smart agriculture: An exhaustive study," Wireless Netw., 2023, doi: 10.1007/s11276-023-03346-0.

[19] Espressif Systems, "ESP-IDF programming guide," 2024. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

[20] InfluxData, "InfluxDB Cloud documentation," 2024. [Online]. Available: https://docs.influxdata.com/influxdb/cloud/

[21] Google, "Flutter documentation," 2024. [Online]. Available: https://docs.flutter.dev/

[22] Y.-T. Ting and K.-Y. Chan, "Optimising performances of LoRa based IoT enabled wireless sensor network for smart agriculture," J. Agric. Food Res., vol. 15, p. 100969, 2024, doi: 10.1016/j.jafr.2024.100969.

[23] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in Proc. 2nd IEEE Workshop Mobile Comput. Syst. Appl., 1999, pp. 90–100.

[24] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," IETF RFC 8949, Dec. 2020. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8949

[25] M. Saban et al., "A smart agricultural system based on PLC and a cloud computing web application using LoRa and LoRaWAN," Sensors, vol. 23, no. 5, p. 2725, 2023, doi: 10.3390/s23052725.

[26] SQLite, "SQLite documentation," 2024. [Online]. Available: https://www.sqlite.org/docs.html

[27] Zetetic, "SQLCipher: Full database encryption for SQLite," 2024. [Online]. Available: https://www.zetetic.net/sqlcipher/

[28] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016.

[29] R. L. Snyder and J. P. de Melo-Abreu, Frost Protection: Fundamentals, Practice and Economics. Rome: FAO, 2005.

[30] D. C. Montgomery and G. C. Runger, Applied Statistics and Probability for Engineers, 7th ed. Hoboken, NJ: Wiley, 2018.

[31] Z. Zhai, J. F. Martínez, V. Beltran, and N. L. Martínez, "Decision support systems for agriculture 4.0: Survey and challenges," Comput. Electron. Agric., vol. 170, p. 105256, 2020.

[32] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," Comput. Electron. Agric., vol. 147, pp. 70–90, 2018.

[33] A. Naseer et al., "A systematic literature review of the IoT in agriculture — Global adoption, innovations, security, and privacy challenges," IEEE Access, vol. 12, pp. 60986–61021, 2024.

[34] M. S. Abdalzaher, M. Krichen, M. Shaaban, and M. M. Fouda, "Quality-focused Internet of Things data management: A survey, perspectives, open issues, and challenges," IEEE Internet Things J., vol. 12, no. 5, pp. 4457–4480, 2025.

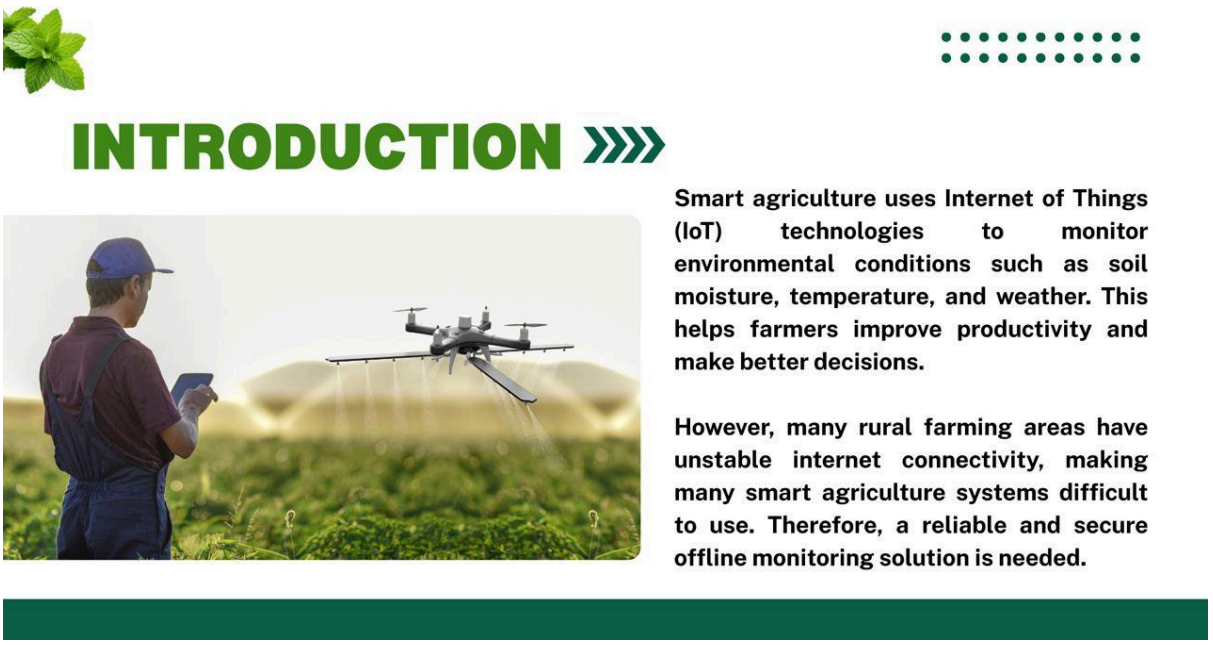
[35] G. K. Sanjay, "Encrypted P2P communication using LoRa waves," Authorea Preprints, 2022, doi: 10.22541/au.166781498.95498042/v1.

[36] A. N. Alsheavi et al., "IoT authentication protocols: Challenges and comparative analysis," ACM Comput. Surv., vol. 57, no. 5, pp. 1–43, 2025.

[37] B. Gunawan and R. P. Astutik, "Implementation of solar energy for an IoT-based agricultural irrigation and monitoring system," J. Eng. Technol. Appl. Sci., vol. 7, no. 3, pp. 188–203, 2025.

Presentation Slides [The Link]





PROBLEM STATEMENT (ROOT SYSTEM)



ROOT is a secure offline-first IoT monitoring system designed for rural agricultural environments. Sensor nodes collect environmental data and transmit it using LoRa communication.
The system processes data locally through an edge gateway and sends alerts to farmers when abnormal conditions are detected. Security mechanisms such as authentication and access control help protect system data and operations.

- Offline-First System
- Edge Data Processing
- Secure Communication

PROPOSED SOLUTION (ROOT SYSTEM)

ROOT is a secure offline-first IoT monitoring system designed for rural agricultural environments. The system deploys sensor nodes across farming fields to monitor environmental conditions and transmit data using LoRa communication.



Instead of relying on constant internet connectivity, ROOT processes data locally through an edge gateway and sends alerts to farmers when abnormal conditions are detected. The system also integrates security mechanisms such as authentication, access control, and secure communication to protect agricultural data and system integrity.

KEY SYSTEM FEATURES



- **Offline-First Operation:** The system continues operating even without internet connectivity.
- **LoRa Long-Range Communication:** Provides reliable low-power communication across large agricultural areas.
- **Edge Data Processing:** Environmental data is processed locally to enable faster response and reduce cloud dependency.
- **SMS Alert System:** Farmers receive alerts directly on their mobile phones without needing smartphones or internet.

COMMUNICATION TECHNOLOGY (LORA)

- Long-Range Wireless Network
- Low Power IoT Communication
- Secure Data Transmission

ROOT uses LoRa communication technology to connect distributed sensor nodes across agricultural fields. LoRa provides long-range wireless communication with very low power consumption, making it suitable for rural environments where connectivity infrastructure is limited.

In addition, communication between devices can be protected using encryption and authentication mechanisms to prevent unauthorized access or data interception.



ALERT SYSTEM (SMS-FIRST) »»



The ROOT system sends alerts to farmers using SMS messages to ensure that notifications reach any mobile phone without requiring internet access. This design improves accessibility for farmers working in rural areas.

When environmental conditions such as drought risk, frost, or abnormal soil moisture are detected, the system generates real-time warnings. An optional mobile application can also be used for extended monitoring when internet connectivity becomes available.



SYSTEM ARCHITECTURE & SECURITY «««

The ROOT system architecture consists of sensor nodes deployed in agricultural fields, a LoRa communication network, and a village gateway responsible for processing environmental data.

To ensure secure system operation, several security mechanisms are integrated. These include role-based access control (RBAC) to manage user permissions, secure device authentication through handshake protocols, encrypted communication, and over-the-air (OTA) firmware updates to maintain system integrity and security.

SECURITY TESTING





The security of the proposed system can be evaluated using several testing methods. Static Application Security Testing (SAST) can be used to analyse source code for vulnerabilities, while Dynamic Application Security Testing (DAST) evaluates the system during runtime.

Additional verification methods include penetration testing to simulate real-world attacks, vulnerability scanning to identify weaknesses, and protocol fuzzing to test communication robustness.

PRESENTATION CONCLUSION

This project demonstrates how a secure offline-first IoT architecture can improve agricultural monitoring in rural environments. By combining LoRa communication, edge data processing, and SMS-based alerts, the ROOT system enables farmers to receive critical information even without reliable internet connectivity.

In addition, integrating security mechanisms such as RBAC, secure device authentication, and encrypted communication helps protect the system from potential cyber threats. Future improvements may include stronger encryption methods, expanded cloud integration, and advanced data analytics for predictive agriculture.

THANK YOU