CASE STUDY
ON
PlantWise: Intelligent Hydration & Environmental Management System

Submitted By

Naggender Singh (2024H1120216P)
Shiven Keshav (2024H1120268P)
Abhishek Bhosale (2024H1120217P)
Vaibhav Tripathi (2024H1030079P)

Submitted To

Dr. Meetha V Shenoy

PlantWise Case Study

User Requirements

1. Crop Selection & Threshold Loading
    - User selects a plant/crop from a predefined list; the system instantly applies that crop's built-in soil-moisture threshold.

2. Real-Time Monitoring
    - Dashboard displays live soil moisture, ambient light, and air temperature/humidity readings.

3. Mode & Pump Control
    - Toggle between Automatic and Manual modes.
    - In Manual mode, press Start or Stop to control the irrigation pump.

4. Automatic Irrigation
    - In Automatic mode, the pump runs whenever soil moisture dips below the selected crop's threshold and stops once it's back above the threshold.

5. Alerts & Notifications
    - Notify when watering starts, stops, or completes.
    - Alert on low-light or out-of-range temperature/humidity.
    - Acknowledge successful threshold updates.

6. Responsiveness & Reliability
    - All user actions and alerts appear within 1 second (soft real-time guarantee).
    - If connectivity is lost, STM32 continues auto-watering with the last threshold; on reconnection, data and alerts sync to the app.

Requirement Specification

Purpose

- To build a Smart Plant Monitoring & Irrigation system that continuously tracks soil moisture, ambient light, and air temperature/humidity, and automatically waters plants when needed.
- To enable real-time, remote monitoring and control of irrigation via a mobile (Android) app, with optional manual override.

Behavior

- The embedded STM32F407 node shall poll soil-moisture, light (LDR), and DHT11 sensors on a fixed schedule (500 ms for UI updates, hourly for irrigation decisions), convert readings to engineering units, and publish them via MQTT.
- In Automatic mode, the system shall compare moisture readings against a configurable threshold and turn the pump ON/OFF accordingly, sweeping the nozzle via a servo.
- In Manual mode, the user may start/stop watering on demand from the Android app; these commands are delivered over MQTT and immediately executed.

Data Requirements

Collected Data:

- Soil moisture (% or analog value)
- Ambient light intensity (lux)
- Air temperature (°C) and relative humidity (%)

Frequency:

- UI-level polling: every 500 ms
- Irrigation decision polling: every 1 hour (via SysTick counters)

Analysis:

- Compare current moisture vs. user-set threshold
- Generate events for “low moisture,” “watering done,” and “weather alert”

System Management Requirements

Communication:

- All inter-device messaging MUST use MQTT over Wi-Fi via an ESP-01 module on the STM32 and Mosquitto on the Raspberry Pi.

Reliability & Real-Time Constraints:

- Sensor reads and UI updates respond within 1 s.
- MQTT QoS 1 for critical commands (pump control, config update).

Security & Privacy:

- Only authenticated app instances may publish control or config topics.
- Broker enforces topic ACLs: app ↔ Pi, Pi ↔ STM32.

User Interface:

- Android app shall indicate connection status, current mode (Auto/Manual), and pump state.

Application Requirements (Android)

Subscriptions:

- plant/sensors/updates (live data)
- plant/sensors/status (periodic heartbeat)
- plant/config/ack (config acknowledgements)

Displays:

- Real-time soil moisture, light, temperature & humidity
- Pump ON/OFF indicator, current operating mode

Controls:

- Toggle Automatic ↔ Manual mode
- Manual Start / Stop watering buttons
- Crop-Selection Dropdown (e.g. “Tomato,” “Lettuce,” “Rose”) that applies preset moisture thresholds under the hood

Feedback:

- Toast or dialog on command success/failure
- Error screen on connectivity loss

System Specification

Once powered, the PlantWise system shall continuously monitor environmental parameters, evaluate automatic irrigation logic, and relay data and commands via MQTT. Sensor readings are sampled every 500 ms. Every 1 hour, the system logs the latest readings and pump state. When soil moisture falls below the selected crop's preset threshold (or on manual-water command), the pump and servo automatically activate for up to 60 s (or until moisture recovers), then shut off. All sensor data, system status, and configuration updates flow STM32 ↔ Raspberry Pi ↔ Android App over MQTT, with the Pi storing data to on-board storage and the App reflecting real-time values and accepting user commands. Pi is being used as a broker for communication between STM32 ↔ Raspberry Pi ↔ Android App over MQTT.

Detailed System Specification

Input Devices

- Soil-Moisture Sensor Module (Analog → STM32 ADC)
- LDR Light Sensor Module (Analog → STM32 ADC)
- DHT11 Temp/Humidity Module (Digital GPIO)

Actuators

- 1-ch Relay Module (5 V) driving DC Water Pump
- Servo Motor via L298N Driver (PWM)

Communication

- ESP-01 Wi-Fi Module (UART ↔ STM32) running MQTT
- Raspberry Pi 3 as MQTT Broker (Mosquitto)
- Android App MQTT Client (Eclipse Paho)

Computational Units

- STM32F407IG-DISC1: Cortex-M4 @168 MHz, 1 MB Flash, 192 KB RAM
- Raspberry Pi 3: ARM Cortex-A53 @1.2 GHz, 1 GB RAM, Raspbian
- Android Smartphone: Android 9+ (MQTT client + UI)

Storage

- Raspberry Pi Micro-SD (64 GB): SQLite logs of hourly readings & pump state
- STM32 retains only live sensor values & config in RAM

Software Platform

- STM32 Firmware: HAL drivers + MQTT client library
- Raspberry Pi: Mosquitto broker + Python scripts (data logger & auto-irrigation logic)
- Android App: Kotlin/Java with Paho MQTT, Jetpack Compose UI

Power Requirements

- 5 V regulated rail for Pi, pump & relay module
- 3.3 V for STM32 and sensors (via on-board regulator)

Peripherals & Timers

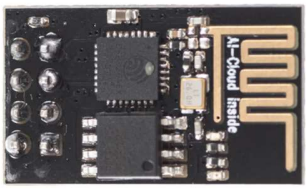
- SysTick @1 ms for 500 ms & 1 hr software counters
- ADC peripheral for soil & light sensors
- USART2 for ESP-01 MQTT
- PWM timer for servo sweep

Environmental

- Operating: 0 °C–50 °C, indoor/outdoor shelter
- Humidity: 10 %–90 % RH (non-condensing)

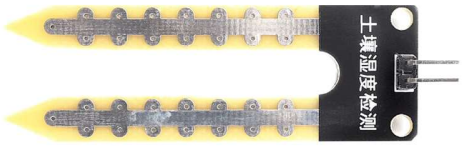
Input Devices & Sensor Details

ESP-01 Wi-Fi Module



- **Function:** Provides 802.11 b/g/n connectivity and MQTT link between STM32 and Broker (Raspberry Pi).
- **Interface Required:**
  - TTL-level UART (TX/RX) on STM32 USART2 @115 200 baud
  - 3.3 V VCC (≥300 mA) and GND
  - CH_PD/EN pin pulled HIGH
- **Requirements:**
  - Wait ≥10 ms after power-up before sending AT commands
  - Common ground with STM32

Soil-Moisture Sensor Module



- **Function:** Outputs an analog voltage proportional to soil water content (0–3.3 V).
- **Interface Required:**
  - STM32 12-bit ADC channel
  - Onboard resistor network (no external divider)
- **Requirements:**
  - Sampling ≤500 ms

Ambient-Light (LDR) Sensor Module

- **Function:** Outputs analog voltage proportional to ambient light.
- **Interface Required:**
  - STM32 12-bit ADC channel
  - Onboard pull-down resistor
- **Requirements:**
  - Sampling ≤500 ms

DHT11 Temperature & Humidity Module

- **Function:** Delivers digital temperature (0–50 °C) and humidity (20–90 % RH).
- **Interface Required:**
  - Single-wire GPIO protocol
  - ±1 ms timing accuracy via timer or delay loops
- **Requirements:**
  - Sampling ~1 Hz per spec

Partitioning & Decomposing the Design

Each module is highly cohesive and loosely coupled via well-defined MQTT topics.

1. Sensor Acquisition & Processing (STM32F407)

- Read soil-moisture (ADC), ambient light (LDR/I²C), DHT11 (temperature/humidity).
- Convert raw readings into engineering units.
- Package and publish JSON to plant/sensors.
- Set flags (s500_flag, h_flag) for timed tasks.

2. Irrigation Control & Local Logic (STM32F407)

- Subscribe to plant/control/app and plant/config topics.
- Apply crop-specific soil-moisture threshold.
- Evaluate moisture vs. threshold and manual flags.
- Drive relay and servo to water plants; handle time-outs and stop conditions.
- Publish watering start/stop/complete notifications to plant/notification.

3. Broker & Automation Engine (Raspberry Pi)

- Run Mosquitto MQTT broker.
- Subscribe to plant/sensors, plant/notification, plant/control/app, plant/config.
- Log all sensor readings & events into a lightweight database.
- On receiving config updates, forward to STM32 and acknowledge on plant/config/ack.
- In Automatic mode, evaluate latest soil data and publish auto-irrigation commands to plant/control/app.
- Respond to app's data-fetch requests by republishing the latest reading.

4. Mobile UI & Configuration (Android App)

- Subscribe to plant/sensors, plant/notification, plant/config/ack.
- Display real-time dashboard: soil moisture, light, humidity/temperature, pump status.
- Allow crop selection (loads built-in threshold) and send plant/config updates.
- Provide Manual mode controls (Start/Stop) via plant/control/app publishes.
- Show histories (pull from Pi), and surface alerts: low-light, out-of-range env, watering events.

Use Case Diagrams

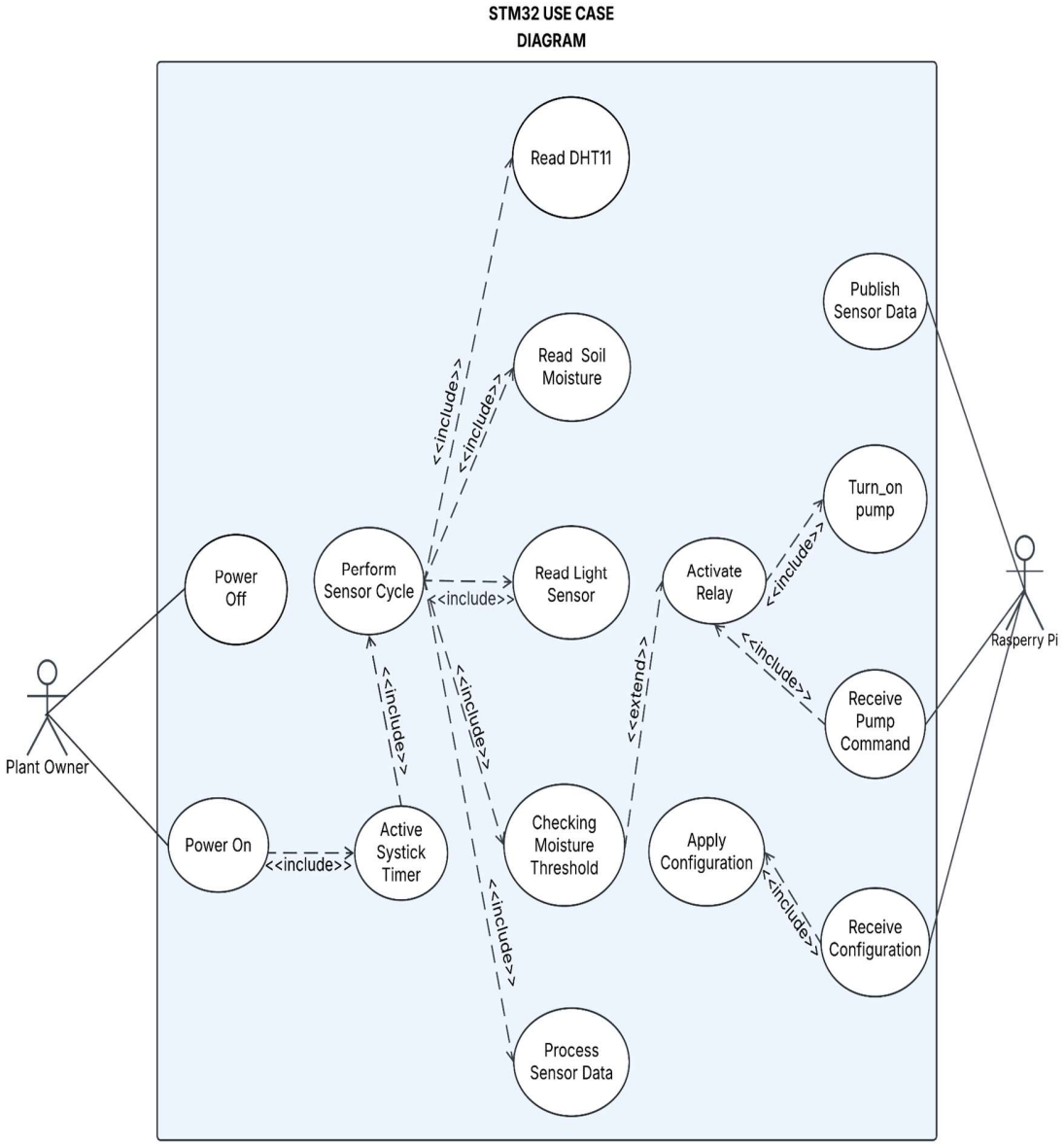


Fig:1

Every hour the STM32 runs a Sensor Cycle, reading DHT11, soil-moisture and light sensors, converting values, and checking the moisture threshold against the latest setting. If moisture is low or if a manual "start" command arrives via Receive Pump Command on plant/control/app it activates the relay to turn on the pump. When a new threshold or mode JSON lands on plant/config, the Apply Configuration use case parses and stores it immediately. After each cycle or configuration change, the STM32 publishes sensor data or config acknowledgements back to the broker (plant/sensors, plant/config/ack).

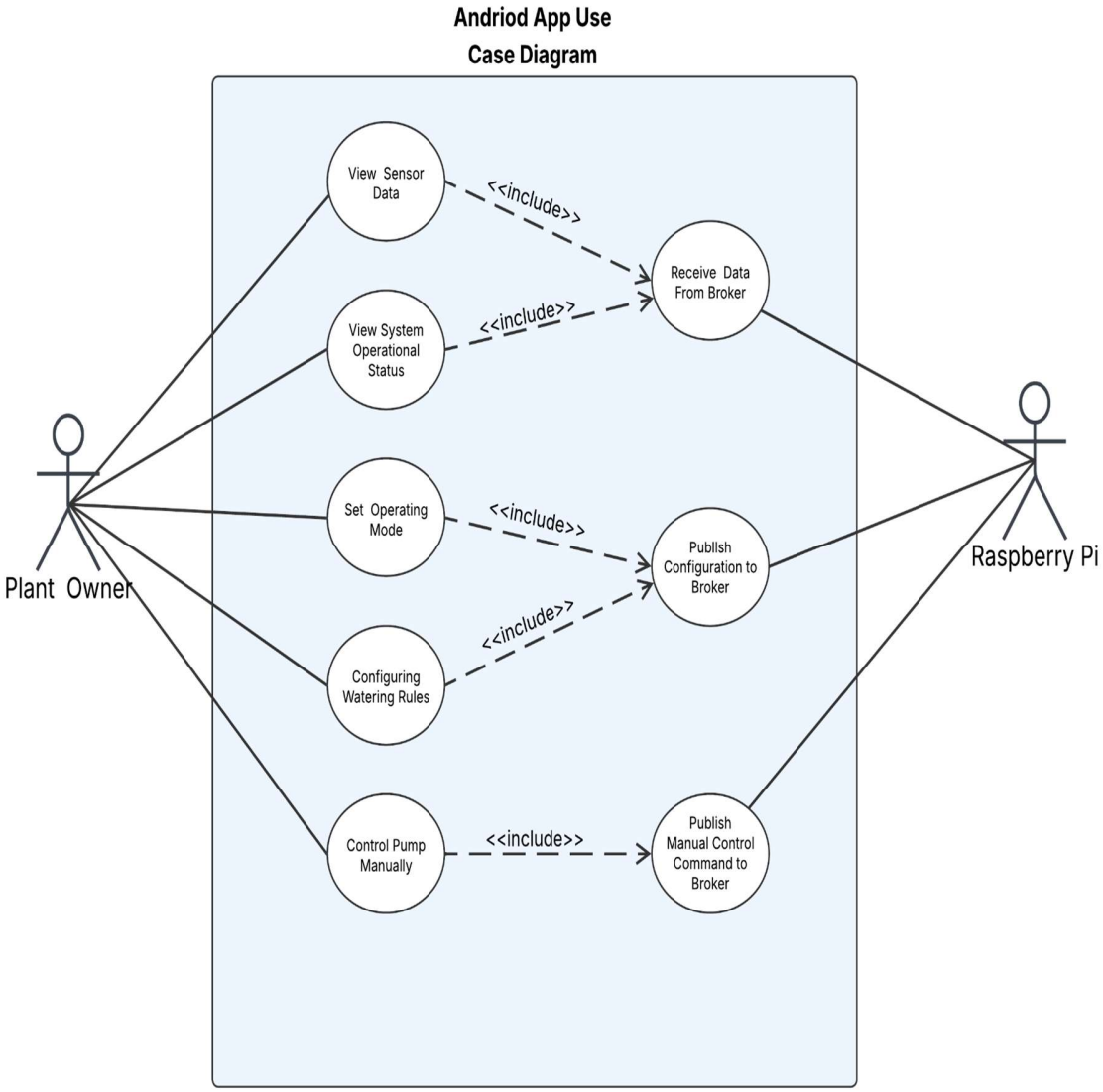


Fig:2

The plant owner can view sensor data (soil moisture, light, temperature, humidity) and see pump status by subscribing to MQTT topics both include pulling real-time JSON from the broker. In Set Operating Mode or Configure Watering Rules, the user adjusts automatic/manual modes and threshold values; each action publishes a configuration JSON to plant/config. When in manual mode, Control Pump Manually sends a simple {"cmd":"start"} or {"cmd":"stop"} to plant/control/app, triggering the same relay logic on the STM32.

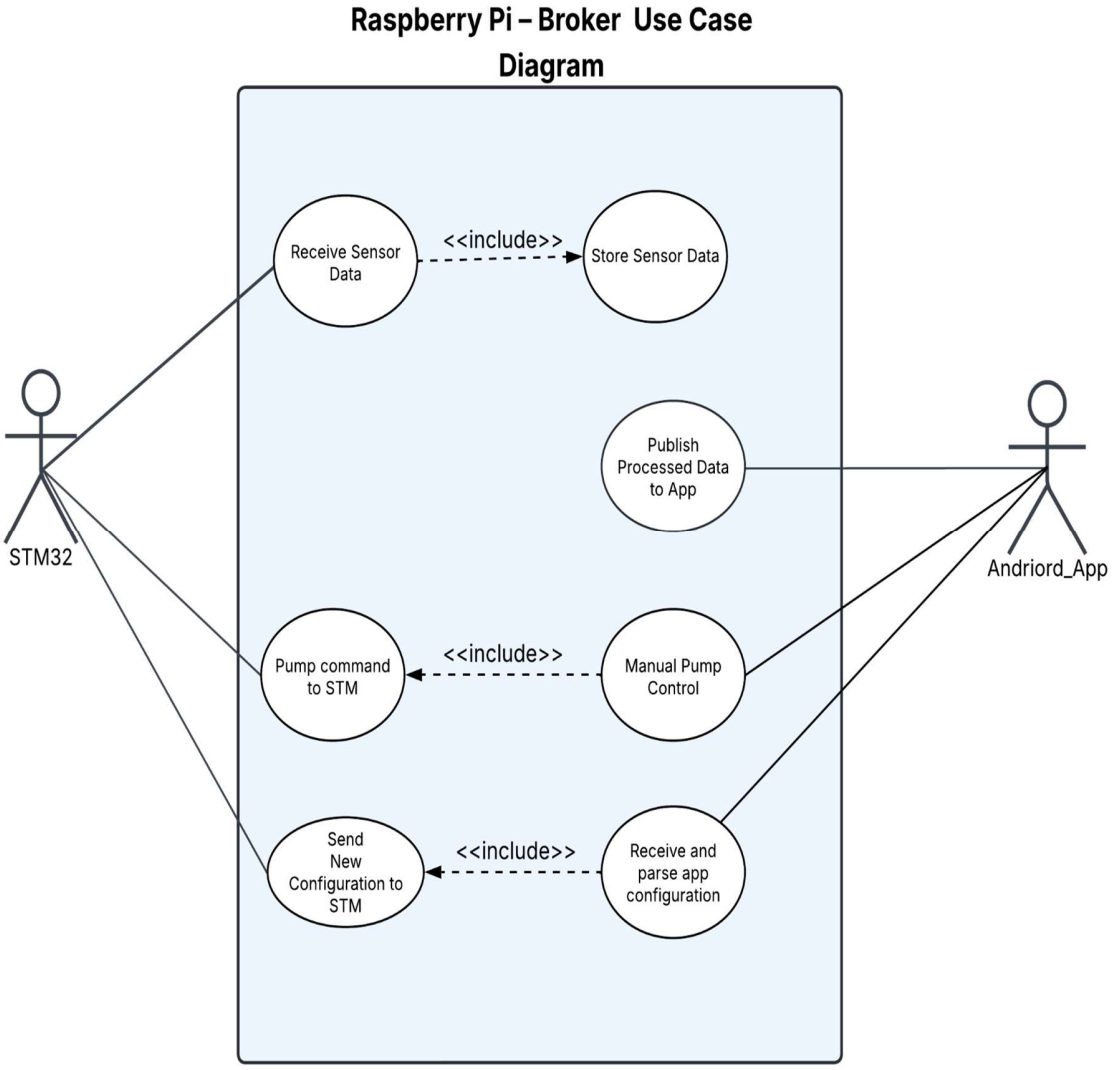


Fig:3

The Pi continuously receives sensor data from plant/sensors, stores it in its database, and optionally pushes processed updates to plant/sensors/updates for the app. When the app issues a manual-pump command or new configuration, the Pi's Receive App Command use case parses those messages and forwards them: pump commands to plant/control/app for the STM32, and threshold/mode changes to plant/config. This ensures the STM32, app, and Pi stay synchronized through MQTT.

Activity Diagrams

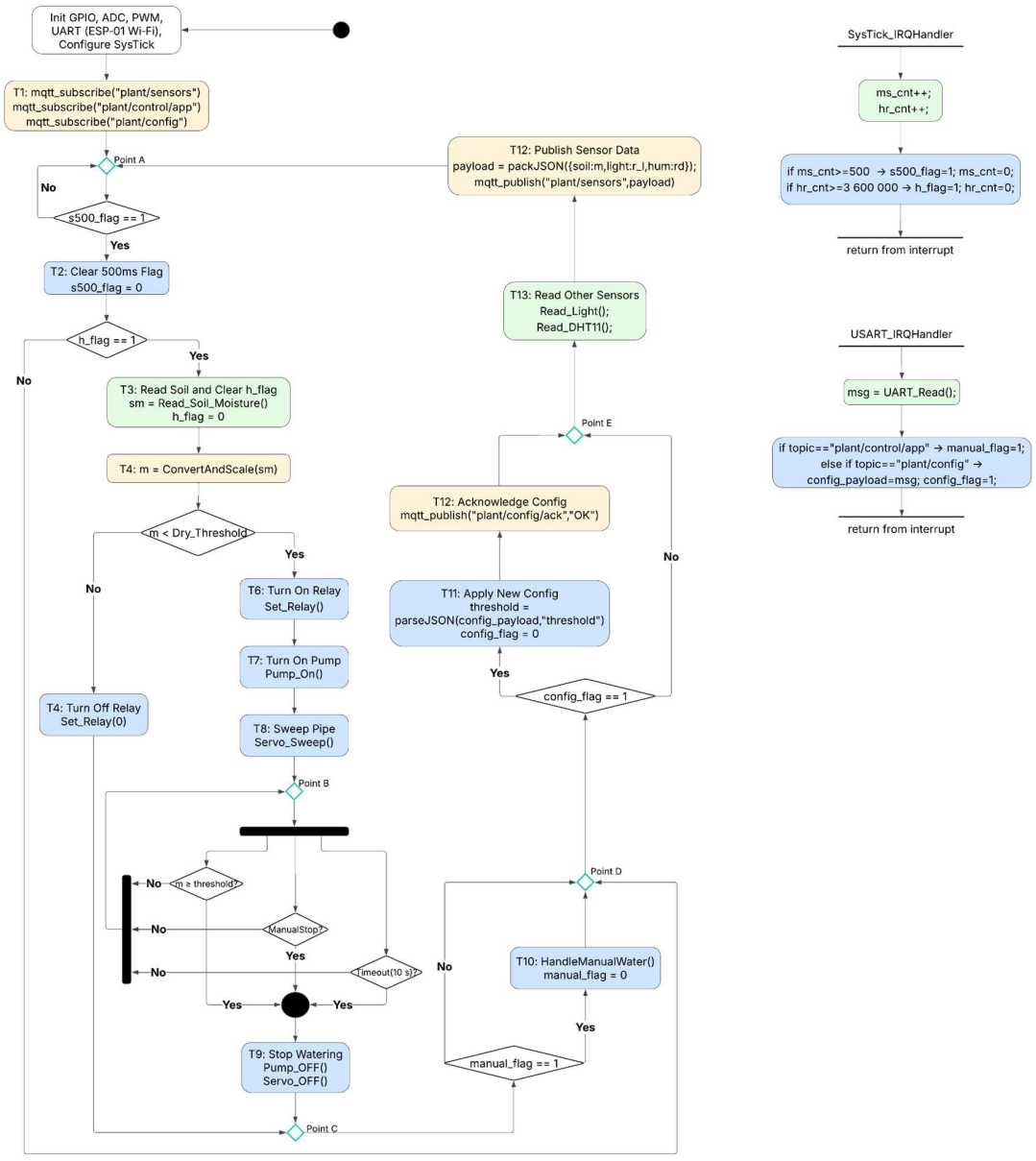


Fig:4 (For STM)

On power-up, the STM32F407 initializes GPIO/ADC/PWM/UART and sets up SysTick for 1 ms ticks, then subscribes to plant/sensors, plant/control/app, and plant/config. A single SysTick_IRQHandler drives two software counters that raise a 500 ms flag (s500_flag) and a 1 hour flag (h_flag). When h_flag fires, the MCU reads and scales soil moisture, clears h_flag, and if below threshold, turns on the pump relay and sweeps the servo until moisture recovers, a manual-stop signal arrives, or 60 s elapses. Independently every 500 ms it reads light and humidity, publishes all sensor JSON, and in its main loop checks for manual commands or config updates , applies new thresholds.

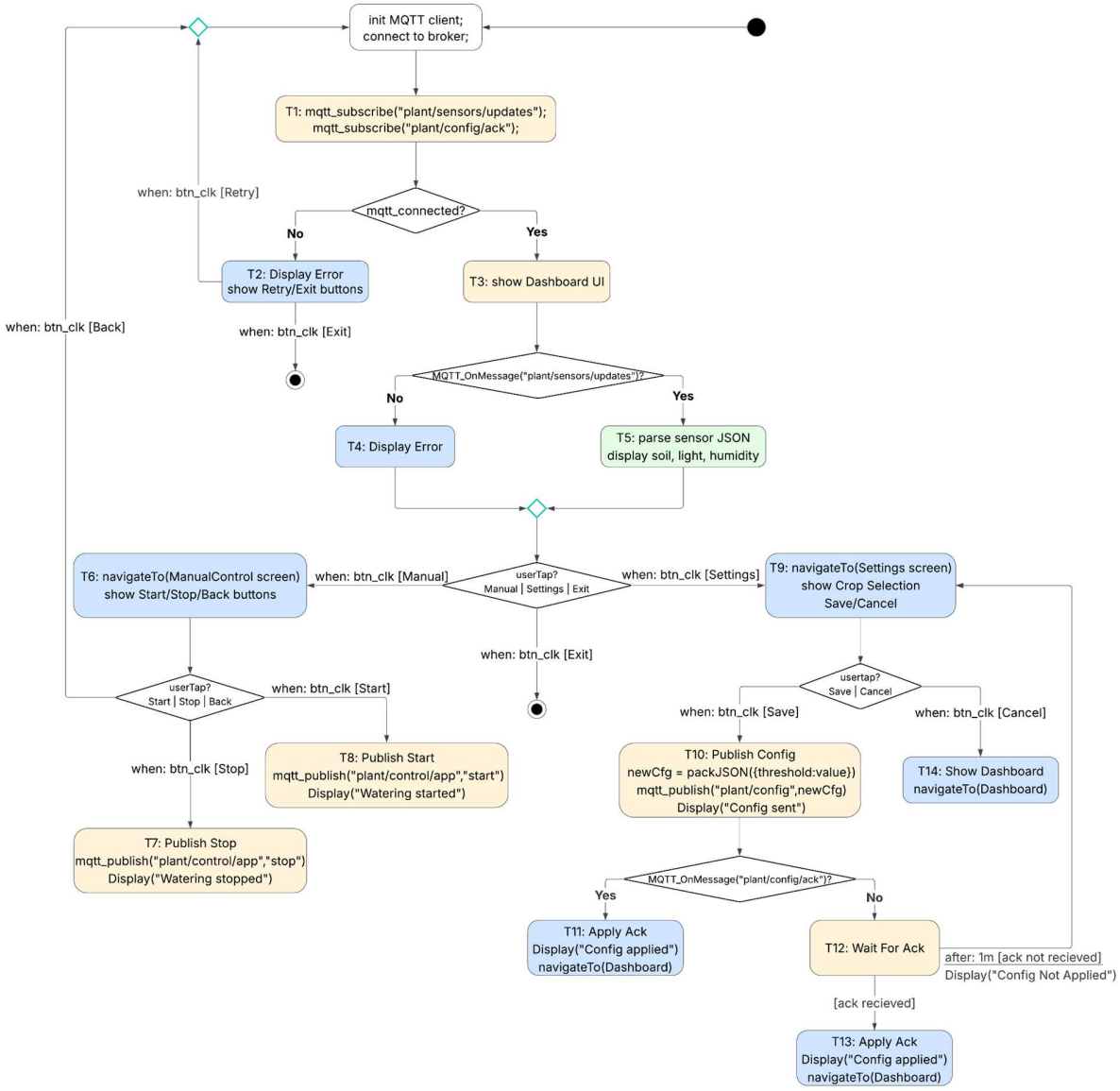


Fig:5 (For Android)

On launch the app initializes its MQTT client, connects, and subscribes to plant/sensors/updates and plant/config/ack; connection failures (<<signal>>) show a Retry/Exit dialog. Once connected it displays a Dashboard with a loading spinner, then waits (<<signal>>) for sensor messages; on success it parses and updates the UI, on parse-error or timeout (<<change>>/<<time>>) it toasts an error and re-awaits. The Dashboard offers Manual, Refresh, Settings, and Exit (<<signal>>) options. Manual opens start/stop controls that publish plant/control/app; Settings shows a threshold slider that publishes plant/config and awaits an ack (<<signal>>) before returning; Refresh simply reloads, and Exit closes the app.

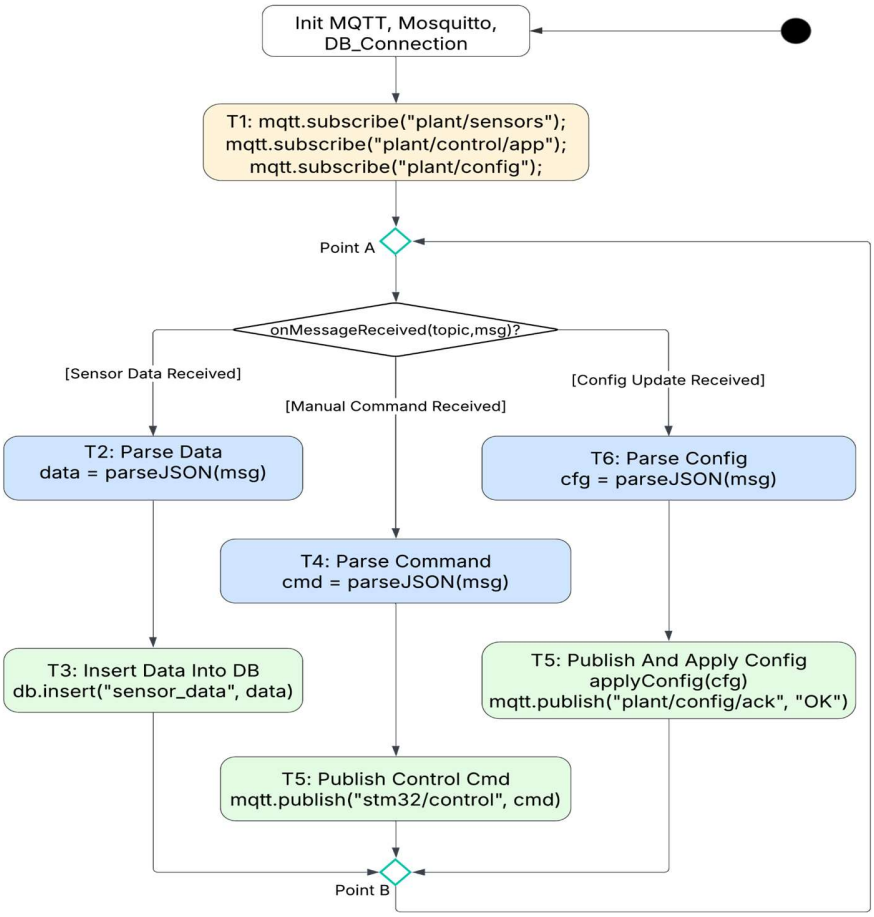


Fig:6 (For Raspberry Pi)

At startup, the Pi launches Mosquitto, opens its database, and subscribes to plant/sensors, plant/control/app, plant/config, and plant/data/request. A fork splits execution into (A) an event-driven branch that logs incoming sensor data, forwards manual-water and config messages (and returns config-acks), and answers data-requests by re-publishing the latest reading; and (B) a timed branch waking every 5 s to auto-water and publish a consolidated status on plant/sensors/status. Both loops run indefinitely, ensuring immediate response to events and regular enforcement of irrigation logic.

State Diagrams

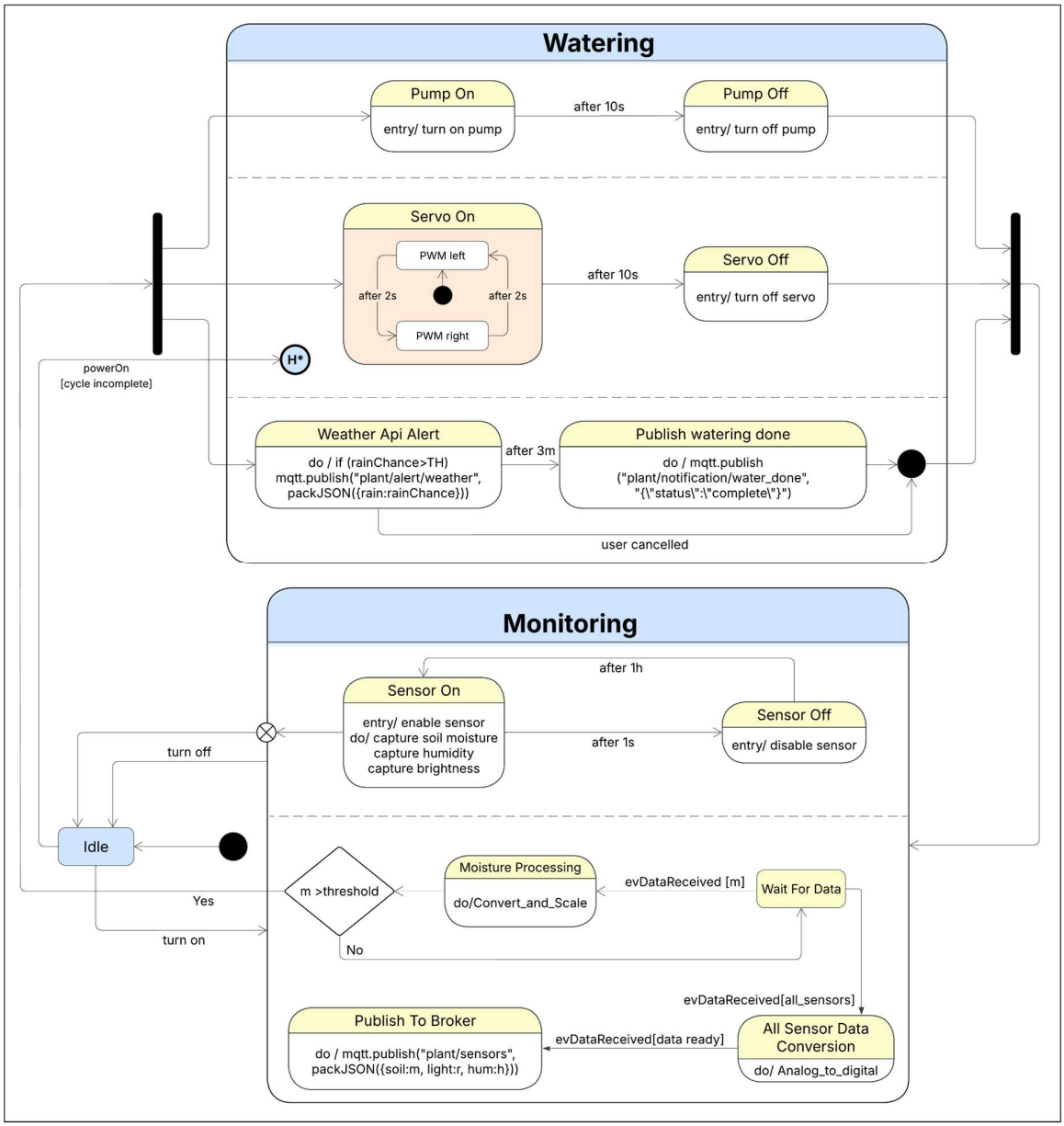


Fig:7 (For STM)

On reset the controller sits in Idle. Every 500 ms SysTick fires a SensorOn transition: it powers ADC/DHT, converts soil, light, and humidity, then on all_sensors done enters AllSensorDataConversion, scales values, publishes to plant/sensors, and returns to Idle. Hourly the same SysTick also triggers a nested PumpOn/ServoOn substate relay and servo start, auto-off via 10 s timers or manual-stop/global 60 s timeout, then back to Idle. In parallel a WeatherApiAlert loop runs every 3 min on rainChance>TH it publishes an alert then after 3 min or user cancel moves to PublishWateringDone, sends a water-done notification, and rejoins Idle.

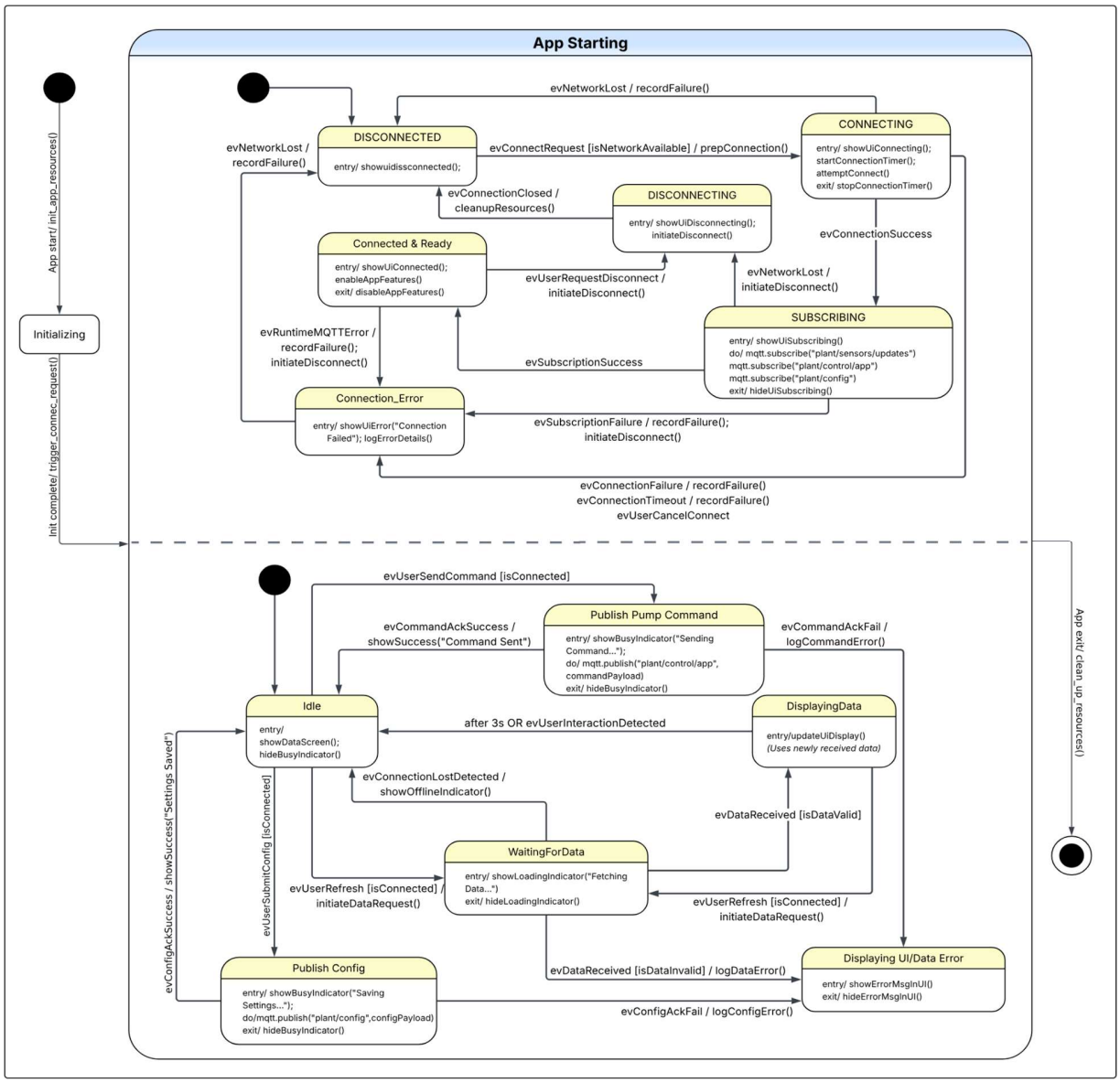


Fig:8 (For Android)

At launch the app enters Initializing (init resources), then CONNECTING (show “connecting”, attemptConnect, timeout guard). On evConnectionSuccess it goes to SUBSCRIBING (show “subscribing”, subscribe to sensor/update, control and config topics), then into Connected & Ready (show “connected”, enable UI). From Ready: Publish Pump Command (on user tap: busy, publish, wait ack, clear busy), WaitingForData (on refresh: loading, request, then DisplayingData or DisplayingError), and Publish Config (on save: busy, publish config, wait ack, clear busy), all returning to Ready.

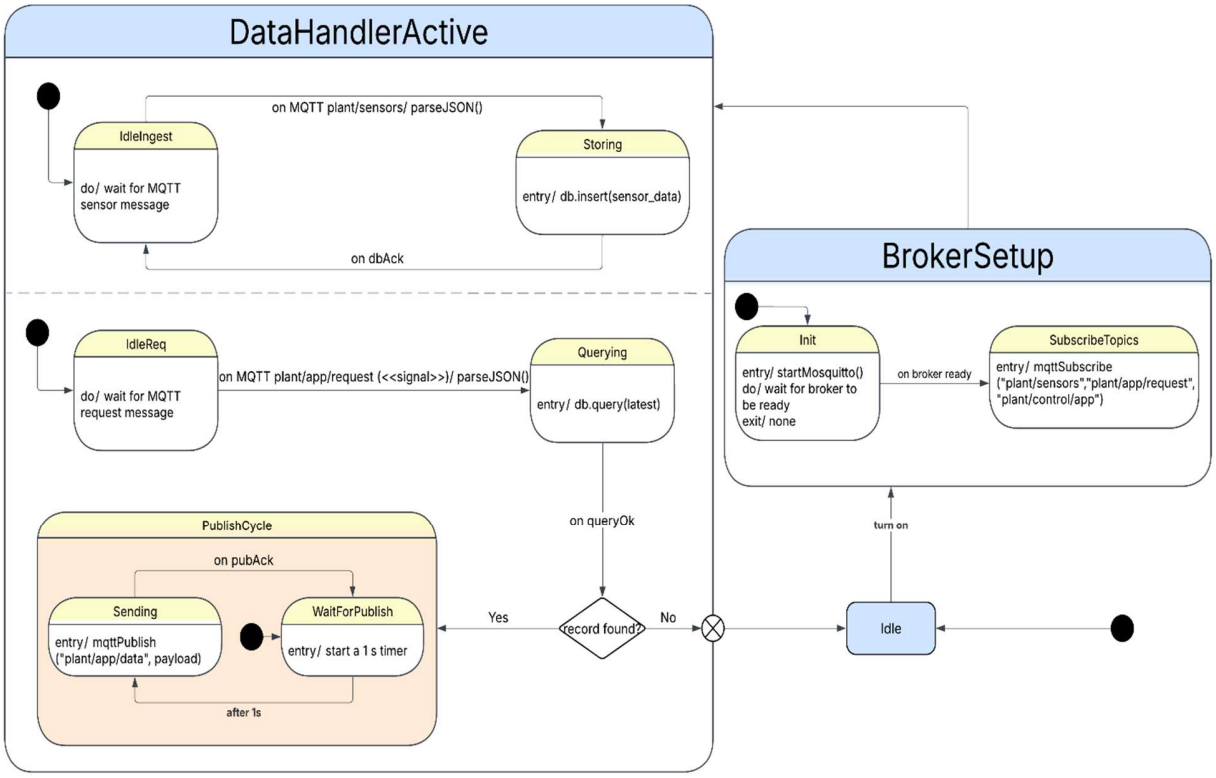


Fig:9 (For Raspberry Pi)

On power-up the Pi runs BrokerSetup→Init (start Mosquitto), then SubscribeTopics (subscribe plant/sensors, plant/app/request, plant/control/app), then into top-level Idle. In Idle two parallel regions operate: IdleIngest loops on plant/sensors, entering Storing (db.insert) then back; IdleReq loops on plant/app/request, transitions to Querying (db.query), then if a record exists enters PublishCycle (publish data, await ack or timeout, repeat) or else returns to Idle ensuring continuous ingestion and on-demand servicing.

Block Diagram

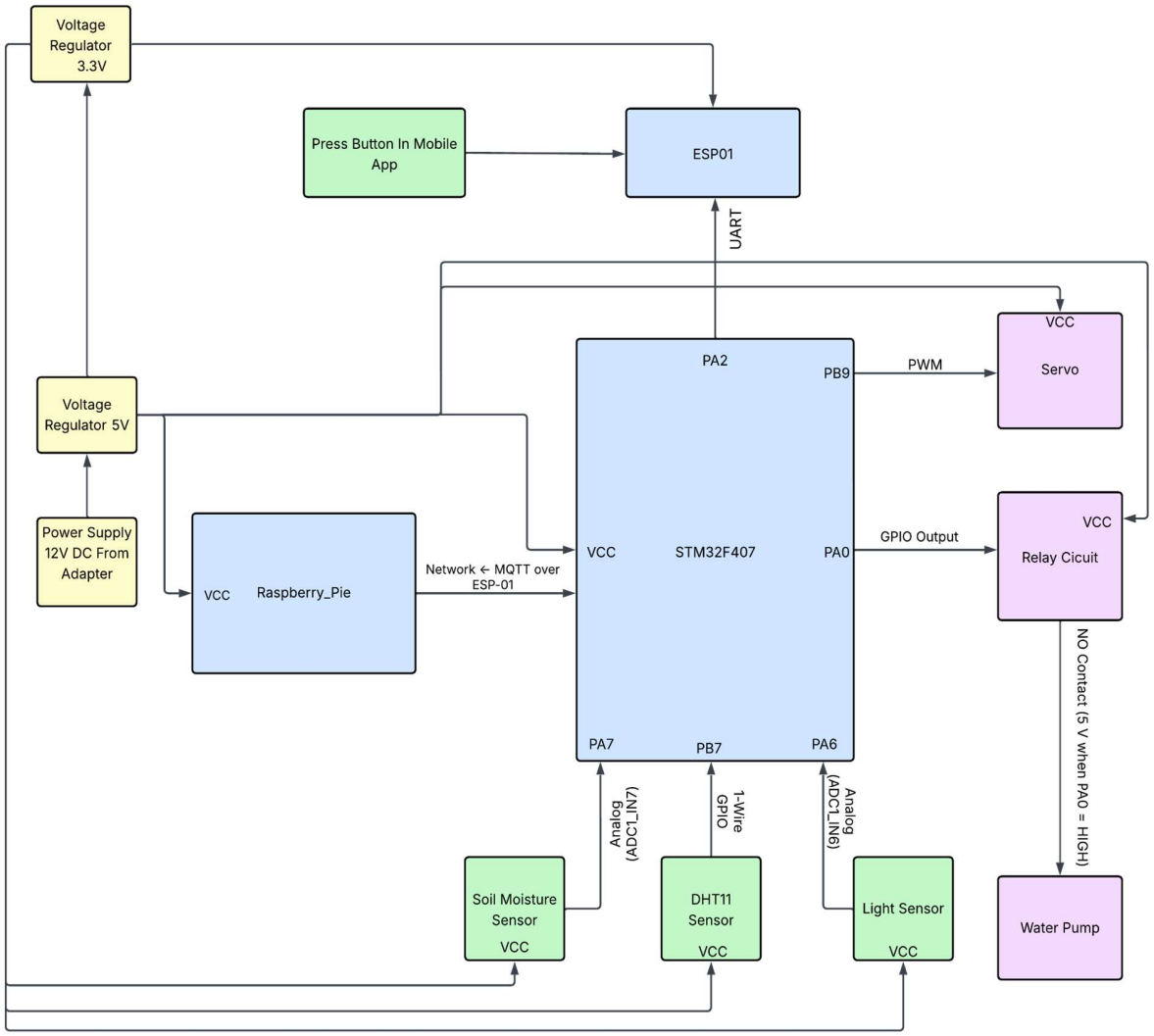


Fig:10

A single 12 V adapter feeds a 5 V regulator and, downstream, a 3.3 V regulator. The 5 V rail powers the Raspberry Pi , the relay board coil, and the DC water-pump motor, while the 3.3 V rail powers the STM32F407 discovery board, ESP-01 Wi-Fi module, and all three sensor modules (soil-moisture, DHT11, LDR). All grounds are common. The STM32's PA7 and PA6 ADC pins sample soil and light, PB7 reads humidity, and PA2 UART connects to the ESP-01 for bidirectional MQTT over Wi-Fi to the Pi. Pump actuation flows from PA0 (driving the 5 V relay coil) and the servo sweep is driven by PB9 PWM. Thus sensor data travels: STM32 ADC → ESP-01 UART → Wi-Fi → Mosquitto → database → Android app; and commands return: Android app → Pi broker → ESP-01 → STM32 → immediate pump or threshold action. This topology concentrates high-current loads on the 5 V rail, keeps logic at 3.3 V, uses one supply, minimizes wiring, and maintains a common-ground for noise resistance.

Class Diagram

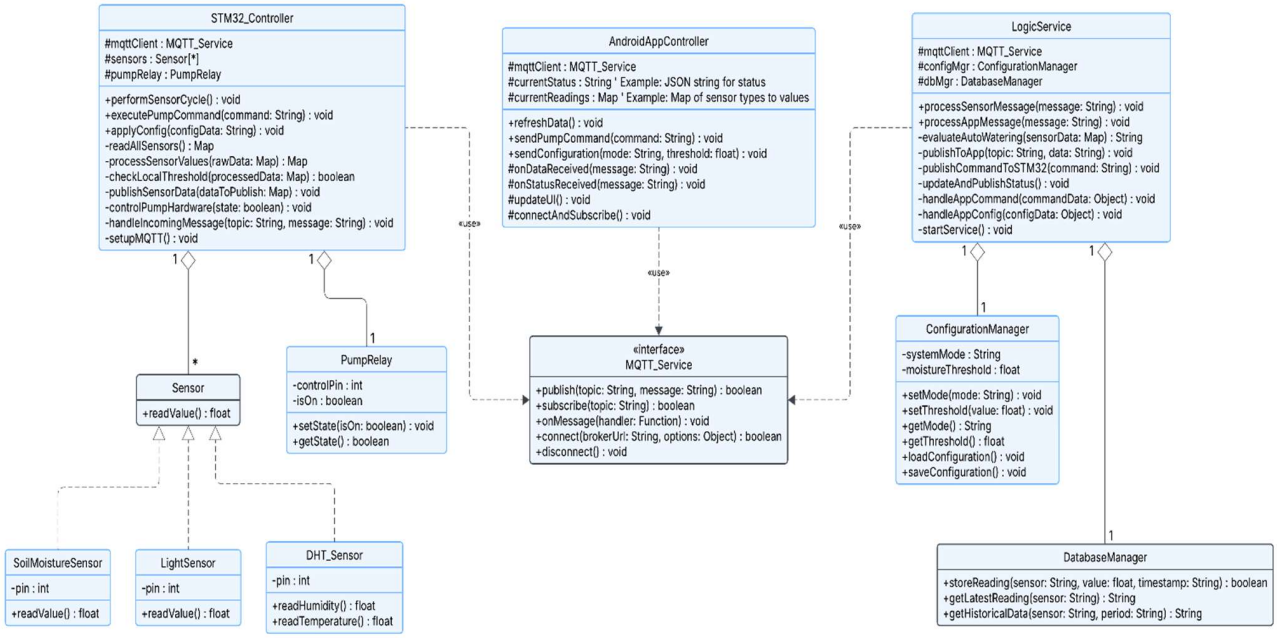


Fig:11

PlantWise is built around a unified MQTT_Service interface that decouples three controllers STM32_Controller, LogicService (on the Pi), and AndroidAppController. On the STM32, STM32_Controller composes multiple Sensor subclasses (SoilMoistureSensor, LightSensor, DHT_Sensor) and a PumpRelay: it polls sensors, enforces local threshold logic, actuates the pump, and publishes or subscribes to MQTT topics. The Pi's LogicService subscribes to sensor, control, config, and data-request topics, persists readings via DatabaseManager, applies auto-watering decisions using threshold values stored in ConfigurationManager, forwards commands to the STM32, and publishes status updates to the app. The AndroidAppController subscribes to sensor updates and config acknowledgments, renders the UI dashboard, sends manual-water or config messages, and handles incoming status messages. ConfigurationManager encapsulates system mode and threshold persistence, while DatabaseManager manages time stamped storage and retrieval of historical and latest sensor data.

Architectural Description

On-Chip Peripherals

- SysTick (1 ms tick)
- ADC1 (soil-moisture, LDR)
- I²C1 (DHT11 temperature & humidity)
- TIM3_PWM (servo sweep)
- USART2 + DMA1_Channel6 (ESP-01 Wi-Fi)
- GPIO (pump-relay drive, optional manual-button IRQ)

Module 1: Sensor Acquisition & Processing (STM32F407)

- **SysTick_Handler**
  - ms / hr counters → s500_flag, h_flag
- **ADC_IRQHandler**
  - m_flag = 1 (soil ADC ready)
- **I²C Read**
  - fetch DHT11 data → dht_flag
- **do ConvertAndScale()**
  - raw ADC & DHT readings → engineering units
- **do mqtt_publish("plant/sensors", JSON{soil,light,hum})**

Module 2: Irrigation Control & Local Logic (STM32F407)

- **USART2_IRQHandler**
  - parseTopic(msg):

    • "plant/control/app" → manual_flag=1
    • "plant/config" → threshold=payload; config_flag=1

- **do EvaluateWatering()**
  - if (m_raw < threshold) ∨ manual_flag:

    • Pump_ON(); Servo_Sweep()
    • Fork-wait on {moisture≥threshold; manual_stop; timeout(60 s)}

• Pump_OFF(); Servo_OFF()

- **do mqtt_publish("plant/notification", “watering started|stopped”)**

Module 3: Broker & Automation Engine (Raspberry Pi)

- **Mosquitto** MQTT broker
- **Python Service**
  - subscribe: plant/sensors, plant/notification, plant/control/app, plant/config
  - do store incoming data/events in SQLite
  - do on config update → publish to plant/config → await ack → publish plant/config/ack
  - do in Automatic mode → evaluate latest soil vs. config → publish “start”/“stop” to plant/control/app
  - do respond to app’s data-fetch requests by republishing latest sensor JSON

Module 4: Mobile UI & Configuration (Android App)

- **MQTT Client (Paho)**
  - subscribe: plant/sensors, plant/notification, plant/config/ack
  - do Display real-time dashboard (soil, light, hum, pump status)
  - do Crop-selection → mqtt_publish("plant/config", {cropId})
  - do Manual controls → mqtt_publish("plant/control/app","start" / "stop")
  - do History view → request via plant/sensors/request → display past readings
  - do Surface alerts: low-light, threshold breach, watering events

Interrupt Priorities

1. SysTick
2. ADC
3. USART_RX / DMA
4. GPIO (manual button)

Power & Clock Considerations

1. Sleep between ticks; wake on SysTick IRQ
2. Clock-gate ADC & I²C when idle
3. Debounce GPIO in software (20 ms)