Automated Plant Watering System with MQTT and ESP32-S3

1. Introduction

Plants require consistent and appropriate watering to grow and remain healthy. However, in daily life, particularly during travel or vacations, it's easy to forget or be unable to water plants. Overwatering and underwatering are common issues that can damage plant health. An automated watering system that responds to soil moisture levels helps ensure optimal plant hydration while minimizing manual intervention.

This project presents a smart plant watering system designed to operate semi-autonomously. It monitors the soil moisture and automatically activates a peristaltic pump via a latching relay to deliver water when needed. The system uses two ESP32-S3 microcontrollers connected over Wi-Fi, communicating through MQTT protocol to exchange sensor data and control messages.

The architecture is split into two subsystems:

* **Sensor Node:** Measures soil moisture and publishes data to an MQTT broker. * **Actuator Node:** Receives watering commands via MQTT and controls the pump.

Users can integrate remote notifications or dashboards via tools such as Node-RED, Telegram bots, or email alerts.

—

2. Materials and Methods

2.1 Materials

Component	Description
————————————	————————————————————
ESP32-S3 DevKit	Microcontroller with built-in Wi-Fi used for both sensor and actuator nodes
Capacitive Soil Moisture Sensor	Provides analog output corresponding to soil moisture
DFRobot Peristaltic Pump (DFR0523)	Waters the plant when triggered
2-Coil Latching Relay (HF0D2/005-S-L2)	Used to control the pump with minimal power consumption
Flexible PVC Tubing	Connects pump to plant pot
LED and resistor (optional)	Indicates system status (e.g., dry soil)
External Power Supply	Powers the pump and relay (e.g., 5V/12V DC adapter)

2.2 Software and Tools

* **Arduino IDE**: For ESP32 programming * **MQTT Broker (HiveMQ Public Broker)**: For communication between devices * **Node-RED (optional)**: For visualization, control, or automation * **Python (optional)**: For logic and notifications * **Telegram Bot (optional)**: Sends alerts to user *

Power Profiler Kit (optional): For measuring energy consumption

—

2.3 System Overview

The system operates in two parts:

Sensor Node (ESP32-S3 + Soil Sensor)

* Reads analog soil moisture value using GPIO35. * Publishes sensor data periodically to MQTT topic `plant/soilMoisture`. * Locally lights up LED if moisture is below threshold. * Could optionally be extended with temperature/humidity sensors.

Actuator Node (ESP32-S3 + Latching Relay + Pump)

* Subscribes to MQTT topic `plant/waterCommand`. * Activates the peristaltic pump via latching relay when commanded. * Could publish status back (e.g., `plant/pumpStatus`) for monitoring.

—

3. Results

3.1 Code Overview

Sensor Node Highlights:

* Uses `analogRead()` to read GPIO35. * Publishes moisture readings every 5 seconds. * LED indicates dryness when value > threshold (e.g., 2500).

Actuator Node Highlights:

* Subscribes to MQTT command `plant/waterCommand`. * Activates one coil of the latching relay to start pump, and another to stop it. * Provides quick response and avoids continuous current draw.

MQTT Topic Examples:

* `plant/soilMoisture` – Publishes sensor readings. * `plant/waterCommand` – Accepts values `“ON”` or `“OFF”`. * `plant/pumpStatus` – (Optional) Publishes `“WATERING”` or `“IDLE”`.

Optional Python Logic:

* Subscribes to `plant/soilMoisture` * If value > 2800 → publishes `“ON”` to `plant/waterCommand` * Sends email or Telegram alert

3.2 Testing and Observations

* In dry air, soil moisture value > 3300 * Wet soil: 1800–2000 * Saturated soil (in water): < 1700

Pump successfully turns on when soil is dry and stops when moisture reaches target range.

—

4. Discussion and Conclusion

This project successfully demonstrates a modular, scalable, and remotely-controllable plant watering system. By separating sensing and actuation, it allows for distributed architecture and clean integration with home automation systems.

Key advantages:

* **Low power**: Latching relay prevents continuous power usage * **Remote access**: MQTT allows external control or monitoring * **Extendibility**: Easily extendable with humidity sensors, temperature sensors, water tank monitoring, etc.

4.1 Improvements and Future Work

* Add **ultrasonic water level sensor** to detect refill needs * Integrate **web dashboard** or **Grafana** for long-term data visualization * Implement **fail-safe conditions** (e.g., prevent overwatering) * Enable **automatic refills** using water reservoir sensors * Use **deep sleep** to reduce ESP32 power consumption

—

5. Media and Demonstration

> \[Link to demo video]
> Fritzing sketch of the circuit
> Photos of the prototype and setup
> Screenshot of MQTT dashboard and Telegram messages

—

References

* [ESP32 Official Documentation](https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/) * [DFRobot Peristaltic Pump Wiki](https://wiki.dfrobot.com/Gravity_Peristaltic_Pump_SKU_DFR0523) * [HiveMQ MQTT Broker](https://www.hivemq.com/public-mqtt-broker/) * [Node-RED Documentation](https://nodered.org/docs/) * [Telegram Bot API](https://core.telegram.org/bots/api) * [ESP32 ADC Tutorial – Random Nerd Tutorials](https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/)

-