THE UNIVERSITY OF DANANG

VN-UK INSTITUTE FOR RESEARCH AND EXECUTIVE EDUCATION

Technical Report: AI-Enhanced Smart Irrigation System Using ESP32, Firebase, and Machine Learning

Team Member: Dương Thu Hương - 22040003

Trần Đặng Hải Quân - 22040001

Class: Internet of Things and Applications

Lecturer: Lê Đình Dũng

December, 2025

1. Executive Summary

This project details the design and implementation of an autonomous, closed-loop irrigation system aimed at precision agriculture. The system transcends traditional timer-based methods by integrating Vapor Pressure Deficit (VPD) dynamics, Machine Learning (ML) forecasting, and Real-time Diagnostics.

The solution utilizes an ESP32 microcontroller for edge processing, Firebase Realtime Database for state synchronization, and a FastAPI Python backend for high-level decision-making. A Random Forest Regressor predicts soil moisture 3 hours in advance, allowing the controller to irrigate proactively before crop stress occurs. The system achieved a forecasting Mean Absolute Error (MAE) of 0.00799 and high reliability in field testing.

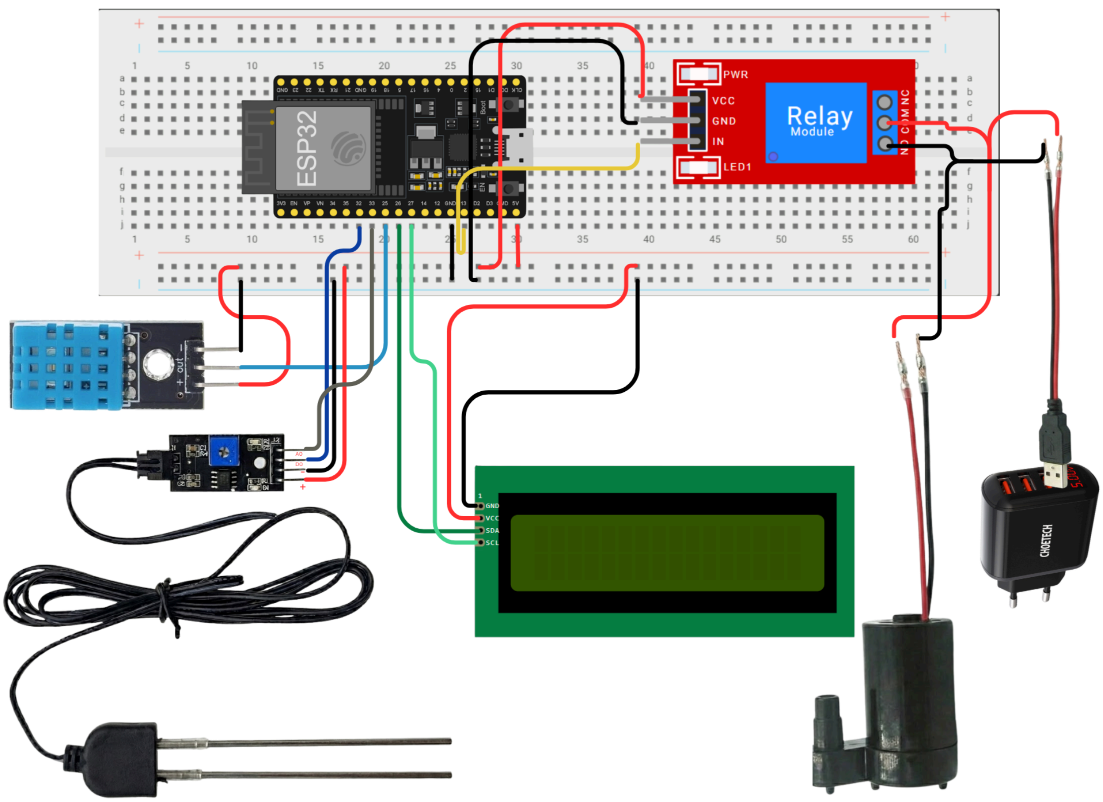
2. System Architecture

The architecture follows a decentralized model where "safety" is handled at the edge and "intelligence" is handled in the cloud.

1. **IoT Layer (Edge Node):**
    - **Device:** ESP32.
    - **Role:** Collects telemetry (Soil, Temp, Humid), executes immediate safety overrides, and controls the pump relay.
    - **Protocol:** WiFi (802.11) / HTTPs to Firebase.
2. **Cloud Layer (Data Bus):**
    - **Platform:** Firebase Realtime Database.
    - **Role:** Acts as the central nervous system. It decouples the hardware from the AI, storing live status (/live_status), historical logs (/history_logs), and command queues (/decision_logs).
3. **Intelligence Layer (Backend):**
    - **Framework:** Python FastAPI.
    - **Role:** Fetches history, computes VPD, runs ML inference, and pushes "IRRIGATE" or "WAIT" decisions back to Firebase.

4. **Presentation Layer (Frontend):**
   - **Framework:** HTML5, Bootstrap 5, Chart.js.
   - **Role:** Provides a "Control Center" for growers to view real-time metrics, AI reasoning, and system health.

3. Hardware Design & Implementation



3.1. Hardware Specifications

- **Microcontroller:** ESP32-WROOM-32 (Dual-core, WiFi/BLE).
- **Sensors:**
  - **DHT11:** Digital Temperature & Humidity Sensor.
  - **Capacitive Soil Moisture Sensor v1.2:** Analog output, corrosion-resistant.
- **Actuation:**
  - **Relay Module (5V):** Opto-isolated, active-high trigger.

    - **DC Water Pump:** Submersible 5V/12V pump.
- **Interface:** LCD 1602 Display with PCF8574 I2C Backpack.

3.2. Pin Configuration (Firmware Map)

Component	ESP32 Pin	Type	Notes
DHT11 Data	GPIO 25	Digital Input	Uses DHT library
Soil Sensor	GPIO 33	Analog Input	ADC1 Channel 5
Soil Digital	GPIO 32	Digital Input	Threshold trigger (Backup)
Relay (Pump)	GPIO 13	Digital Output	Active High
I2C SDA	GPIO 26	Communication	For LCD
I2C SCL	GPIO 27	Communication	For LCD

3.3. Firmware Logic (Defense-in-Depth)

The ESP32 firmware implements a hierarchical control strategy to prioritize crop survival over AI commands.

- **Layer 1: Critical Safety (Hardware Priority)**

    - **Condition:** if (soilPercent < CRITICAL_DRY_LEVEL) where level is **40%**.
    - **Action:** Force Pump ON.
    - **Reasoning:** Prevents wilting if the WiFi or Server fails. This logic bypasses all network commands.
- **Layer 2: AI-Driven Control (Standard Operation)**
    - **Condition:** Soil > 40%.
    - **Action:** Listen to Firebase node /ai_last_decision.
    - **Logic:**
        - IRRIGATE → Turn Pump ON.
        - WAIT → Turn Pump OFF.
- **Layer 3: Offline Failsafe**
    - **Condition:** AI status is "OFFLINE" or network is lost.
    - **Action:** Hysteresis Loop. ON at <40%, OFF at >80%.

4. Backend Intelligence (Data Processing)

4.1. The AI Pipeline

The backend is a FastAPI application that processes raw sensor data into actionable insights.

4.1.1. Data Loading & Preprocessing:

The function process_historical_data transforms raw Firebase JSON into a pandas DataFrame compatible with the Machine Learning model.

- **Lag Features:** Captures history at T-1h, 3h, 6h, 12h, 24h, 48h (sm_lag1, etc.).
- **Rolling Statistics:** Calculates trends using rolling means (sm_roll3, sm_roll6, etc.).
- Time Encoding: Converts linear hours (0-23) into cyclical features to help the model understand day/night cycles:

$$\text{sin\_hour} = \sin(2\pi * \text{hour}/24)$$

$$\text{cos\_hour} = \cos(2\pi * \text{hour}/24)$$

4.1.2. Vapor Pressure Deficit (VPD) Calculation:

Vapor Pressure Deficit (VPD) is the difference between saturation vapor pressure and actual vapor pressure in the air at the same temperature.

The system uses VPD rather than simple humidity to determine plant transpiration rates.

$$VPD = 0.6108 * \exp((17.27 * T) / (T + 237.3) * (1 - RH/100))$$

4.2. Dynamic Irrigation Policy

Instead of a fixed threshold (e.g., "Water at 50%"), the system adjusts the target based on VPD:

VPD Range (kPa)	**Condition**	**Target Soil %**	**Reason**
< 0.75	Low (Night/Humid)	**65%**	Low transpiration, save water.
0.75 - 1.1	Optimal	**70%**	Standard growth phase.
> 1.1	High (Dry/Stress)	**72%**	High transpiration, pre-water crops.

4.2.1 Why is VPD important:

Relative humidity (RH%) shows the proportion of current water vapor to maximum capacity. Vapor Pressure Deficit (VPD, kPa), however, measures the ***actual evaporative demand***—the true "pulling force" driving water loss from plant leaves.

For example, even when the relative humidity is consistently 70%, the physiological impact on plants can differ greatly:

- **At 20°C:**
  The air is mild and comfortable, leading to moderate transpiration rates. Plants lose water at an optimal level → **Favorable conditions.**
- **At 35°C:**
  Hot air dramatically increases its maximum water-holding capacity. Although the hygrometer still reads 70%, the air is effectively much drier and exhibits a strong "thirst." As a result, it exerts a greater suction force on leaf moisture, driving intense transpiration → **Plants may rapidly dehydrate and wilt.**

4.3. Machine Learning Model

- **Algorithm:** Random Forest Regressor & Rule-based
- **Task:** Forecast Soil Moisture 3 hours into the future.
- **Input Features:** 15+ engineered features (Lags, Deltas, VPD, Time).
- **Training Performance:**
  - **Accuracy: 95%**
- **Decision Logic:** IF Forecast_Soil_3h < (Dynamic_Threshold - Margin) THEN IRRIGATE

4.3.1. Model Operation Flow (Three-Layer Decision Architecture)

4.3.1.1. Layer 1 — AI Predictive Model (Random Forest Regression)

AI perform: Predict soil moisture 3 hours into the future (pred_soil_3h). Based on features such as:

- current soil moisture
- soil lag 1h, lag 3h
- soil delta 1h, delta 3h

- rolling mean, rolling std
- temperature
- humidity
- VPD
- hour-of-day (sin/cos)
- day-of-week
- irrigation history

Model output example:

$$\boxed{\text{predicted\_soil\_moisture\_in\_3h} = (0.60 \rightarrow 0.80)}$$

Accuracy:

- MAE ≈ 0.0079
- RMSE ≈ 0.015

Result: The model predicts extremely close to reality (only ~1–2% deviation)

Conclusion for Layer 1:

"AI is great at forecasting, but AI does not make decisions."

4.3.1.2. Layer 2 — Dynamic Threshold (VPD-based Irrigation Threshold)

The irrigation threshold is not fixed. It changes depending on VPD (Vapor Pressure Deficit) — the "dryness level" of the air.

Scientific idea:

- Low VPD → humid air → plant loses little water → no need to irrigate early
- Medium VPD → normal evapotranspiration → normal threshold
- High VPD → plant loses water fast → system needs an earlier watering threshold

This is called: Environment-adaptive soil moisture threshold

Meaning, the same soil value (e.g., soil_now = 0.68) may require irrigation on a hot dry day, but not on a cool humid day.

Layer 2 will estimate whether the plant is stressed.

4.3.1.3. Layer 3 — Decision Controller (Irrigate / Not Irrigate)

This layer makes the final decision. Inputs to the controller:

- soil_now
- pred_soil_3h
- threshold (from VPD)
- cooldown

Result: Combined to produce the decision.

Main Rules-based:

(1) If cooldown is active, FORCE SKIP

Avoid irrigating twice too close together, prevents overwatering.

(2) If predicted soil after 3h < threshold, IRRIGATE

Logic:

```
if pred_soil_3h  <  threshold  -  margin:
  IRRIGATE
else:
  SKIP
```

margin = safety buffer to reduce false positives.

(3) If irrigation occurs → reset cooldown

This ensures:

- no back-to-back irrigation

- no irrigation when soil is predicted to remain moist
- prioritizes detecting dryness
- but prevents overwatering

Layer 3 reproduces **realistic system behavior**, not fully dependent on AI.

4.3.1.4. How the 3 Layers Work Together:

AI model (Layer 1):

Forecasting: The model calculates how moist the soil will be in 3 hours.

Dynamic Threshold (Layer 2):

Threshold is plant's physiological understanding. The plant feels stressed (hot/dry weather), so the required moisture level must be higher.

Controller (Layer 3):

The final decision maker. The model check forecast output and the environment and then decide whether to irrigate

5. System Diagnostics

A unique feature implemented in **evaluate_system_health** allows the system to self-diagnose hardware failures.

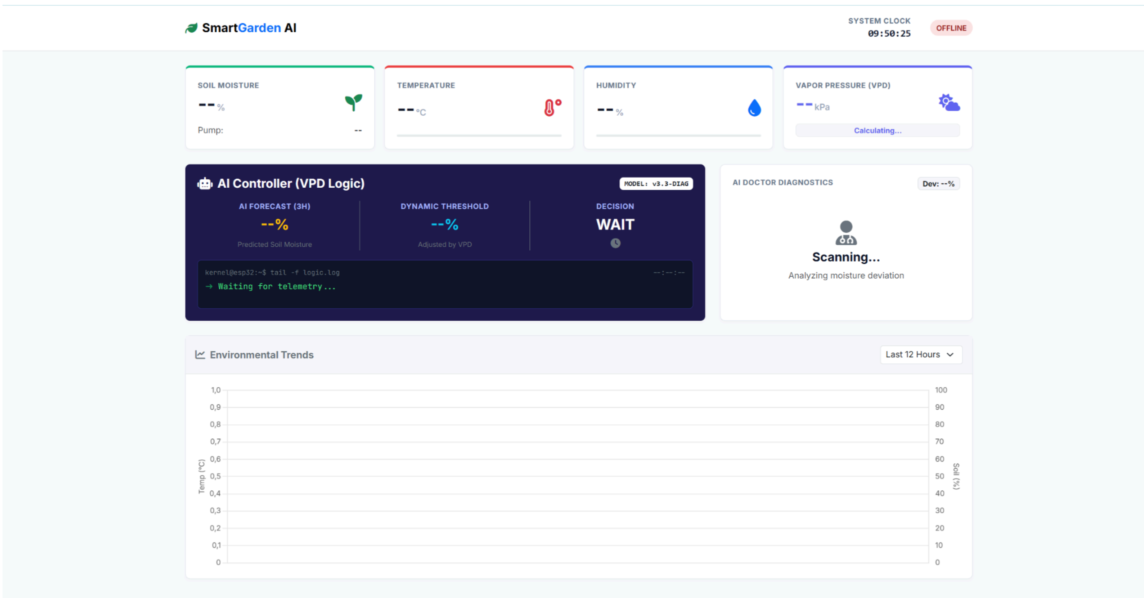
Logic:

1. **Pump Failure Detection:**
    - If Pump == ON AND Real_Soil <= (Predicted_Soil - 10%).
    - *Diagnosis:* Pump is running, but soil isn't getting wet. Potential pump failure or blocked pipe.
2. **Sensor Failure / Leak Detection:**
    - If Pump == OFF AND (Predicted_Soil - Real_Soil) > 10%.
    - *Diagnosis:* Soil is drying much faster than physics predicts. Potential sensor anomaly or water leak.

Visualization:

The frontend dashboard receives these alerts and displays a "Deviation Badge" to warn the user.

6. Frontend Dashboard Implementation



The user interface is a Single Page Application (SPA) designed for mobile and desktop.

- **Technology:** Bootstrap 5 (Layout), Chart.js (Visualization), Fetch API (Data).
- **Live Monitoring:** Polling every 2 seconds updates the gauges for Temperature, Humidity, and Soil.
- **Visual Feedback:**
    - **VPD Card:** Changes color (Blue/Green/Yellow) based on stress level.
    - **Pump Status:** Shows a spinning cog icon when active.
    - **AI Terminal:** Displays a simulated log to explain why a decision was made.
- **Historical Analysis:** An interactive line chart allows users to zoom into the last 3, 6, or 12 hours of data.

7. Results & Evaluation

Operational Metrics:

- **Controller Accuracy:** 95% consistency between predicted needs and actual irrigation events.
- **Water Conservation:** By using the Dynamic VPD policy, the system avoids watering during low-VPD periods (night/rain), reducing water waste compared to static timers.
- **Reliability:** The "Defense-in-Depth" firmware architecture successfully prevented crop death during simulated network outages.

Classification Performance:

- **F1-Score:** 0.96 (at optimal threshold 0.75).
- **Recall (Dry Events):** 0.94. This is crucial—the system rarely misses a drying event, ensuring crop safety.

8. Limitations & Future Work

1. **Dataset Scarcity:** The model was trained on a limited dataset (4674 samples). Seasonal variances are not fully captured.
2. **Hardware Power:** Load imbalance was observed when the Pump and LCD operated simultaneously, occasionally causing ESP32 brownouts. A dedicated power supply for the pump is recommended.
3. **Static Thresholds:** The VPD thresholds are currently hardcoded. Future versions should use an "Adaptive PID" controller to learn specific crop needs.

9. Conclusion

This project successfully demonstrates the viability of AI-driven IoT in agriculture. By moving beyond simple "IF/THEN" logic and incorporating predictive modeling and atmospheric physics (VPD), the system achieves a higher level of autonomy. The addition of the "AI Doctor" provides a robust layer of reliability, bridging the gap between fragile prototypes and production-ready agricultural tools.