🌱 Complete Guide: Real Sensor Integration for Smart Irrigation

Table of Contents

---

1. Hardware Requirements

Essential Components

Component	Model Options	Purpose
Microcontroller	ESP32 DevKit V1	Main processor + WiFi
Soil Moisture	Capacitive v1.2 or v2.0	Measures soil water content
Temperature	DHT22 or BMP280	Ambient/soil temperature
Relay Module	5V 1-Channel	Controls water pump/valve
Water Pump	12V DC submersible	Delivers water
Power Supply	12V 2A + Buck converter	Powers system

Why Capacitive Soil Moisture Sensor?

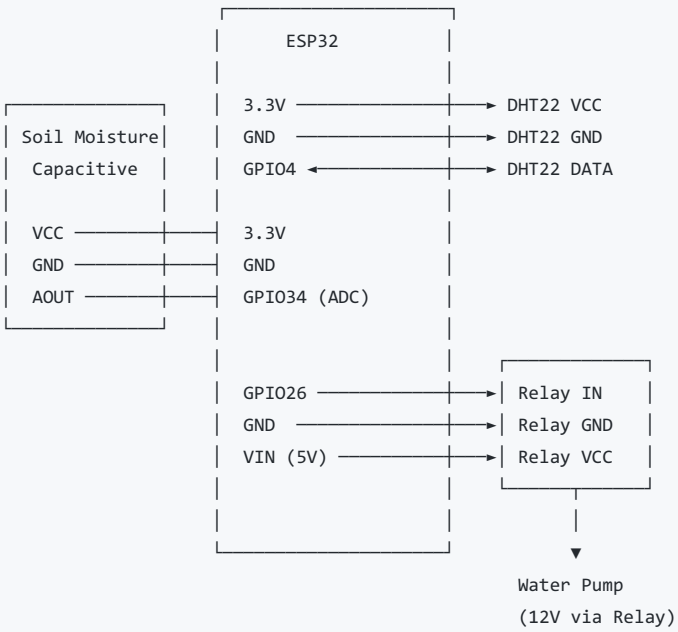
DO NOT use resistive sensors (the ones with exposed metal probes). They corrode quickly in soil and give unreliable readings.

Capacitive sensors (sealed, no exposed metal):

- ✅ Long lifespan (years vs weeks)
- ✅ Stable readings
- ✅ No electrolysis/corrosion
- ✅ Works in various soil types

---

2. Wiring Diagram



Pin Assignments

ESP32 Pin	Connected To	Notes
GPIO34	Soil Moisture AOUT	ADC1 channel (use 34-39 for analog)
GPIO4	DHT22 Data	With 10kΩ pull-up resistor
GPIO26	Relay IN	Digital output
3.3V	Sensors VCC	**NOT 5V** for soil sensor
GND	Common ground	All components share ground

---

3. Sensor Calibration

Temperature Sensor (DHT22)

DHT22 is factory calibrated. Just verify readings:

```
// Verification: Compare with a known thermometer
// Acceptable error: ±0.5°C
```

Soil Moisture Sensor - CRITICAL!

You MUST calibrate for YOUR specific sensor and soil type.

Step 1: Find Dry Value (Air)

```
// Hold sensor in AIR (completely dry)
// Record the analog reading
int DRY_VALUE = ???;  // Typically 2800-3500
```

Step 2: Find Wet Value (Water)

```
// Submerge sensor in WATER (not past the line!)
// Record the analog reading
int WET_VALUE = ???;  // Typically 1000-1500
```

Step 3: Calculate Your Mapping

```
// Your calibration values (REPLACE WITH YOUR READINGS)
#define MOISTURE_DRY    3100   // Reading in air
#define MOISTURE_WET    1400   // Reading in water

// Map to 0-100% scale
float readMoisturePercent() {
    int raw = analogRead(SOIL_PIN);
    float percent = map(raw, MOISTURE_DRY, MOISTURE_WET, 0, 100);
    return constrain(percent, 0, 100);
}
```

Step 4: Map to Training Data Scale

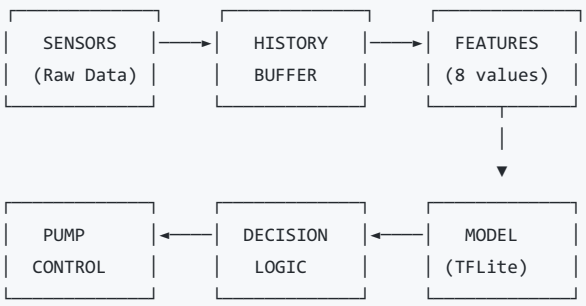
Your model was trained on data with moisture values roughly 100-400. You need to map your 0-100% readings to this scale:

```
// Map percentage to training data scale
float mapToTrainingScale(float percent) {
    // Training data range: ~100 (very dry) to ~400 (very wet)
    // 0% moisture → 100 (dry)
    // 100% moisture → 400 (wet)
    return 100 + (percent * 3.0);  // Maps 0-100% to 100-400
}
```

---

4. Software Architecture

Data Flow



Timing Requirements

Action	Interval	Reason
Sensor Reading	1-5 minutes	Soil changes slowly
Model Inference	After each reading	Make prediction
History Buffer	Stores last 4 readings	For trend calculation
Pump Check	After each inference	Act on prediction

---

5. Feature Calculation from Real Sensors

The 8 Features Your Model Needs

Every time you run inference, you must provide these 8 values **in this exact order**:

```
Index  Feature              How to Calculate
─────  ─────────────────    ──────────────────────────────────
[0]    temperature          Current DHT22 reading
[1]    soilmoisture         Current soil sensor (mapped to 100-400)
[2]    temperature_mean     Average of last 4 temperature readings
[3]    soilmoisture_mean    Average of last 4 moisture readings
[4]    temperature_trend    Current temp - temp from 4 readings ago
[5]    soilmoisture_trend   Current moisture - moisture from 4 readings ago
[6]    soilmoisture_lag_1   Moisture from 1 reading ago
[7]    soilmoisture_lag_2   Moisture from 2 readings ago
```

Visual Example

```
Time:        T-3    T-2    T-1    T-0 (now)
             ───    ───    ───    ────────
Temperature: 28.5   29.0   29.5   30.0
Moisture:    320    290    260    230

Calculated Features:
──────────────────────────────────────
[0] temperature      = 30.0           (current)
[1] soilmoisture     = 230            (current)
[2] temperature_mean = (28.5+29.0+29.5+30.0)/4 = 29.25
[3] moisture_mean    = (320+290+260+230)/4 = 275
[4] temp_trend       = 30.0 - 28.5 = +1.5  (rising)
[5] moisture_trend   = 230 - 320 = -90     (dropping fast!)
[6] moisture_lag_1   = 260            (previous reading)
[7] moisture_lag_2   = 290            (2 readings ago)
```

Important: Normalization

Before feeding to the model, normalize each feature:

```
normalized[i] = (features[i] - mean[i]) / std[i];
```

Use the mean and std values from your training (in `irrigation_model.h`).

---

6. Reading Interval Considerations

What Interval Should You Use?

Your training data determines this. Check your original dataset:

```
# In your Colab notebook, check time between readings:
df['datetime'] = pd.to_datetime(df['date'] + ' ' + df['time'])
time_diff = df['datetime'].diff().median()
print(f"Median interval: {time_diff}")
```

Common Intervals

Interval	Use Case
1 minute	Testing/demonstration
5 minutes	Active monitoring, hot days
15 minutes	Normal operation
30-60 minutes	Low-power/battery operation

Matching Training Data

Critical: Your prediction quality depends on using similar intervals to your training data. If training data was collected every 10 minutes, use ~10 minute intervals in production.

---

7. Decision Thresholds

Hysteresis for Pump Control

Don't turn pump on/off at exactly 50% probability. Use hysteresis:

```
#define THRESHOLD_ON   0.70   // Turn ON above 70%
#define THRESHOLD_OFF  0.30   // Turn OFF below 30%

// This prevents rapid on/off cycling when probability hovers around 50%
```

Minimum Run Time

Protect your pump:

```
#define MIN_PUMP_ON_TIME    30000   // 30 seconds minimum
#define MIN_PUMP_OFF_TIME   60000   // 60 seconds minimum between runs
#define MAX_PUMP_RUN_TIME  300000   // 5 minutes maximum continuous
```

---

8. Error Handling

Sensor Failure Detection

```
bool isSensorValid(float temp, float moisture) {
    // Temperature sanity check
    if (temp < -10 || temp > 60) return false;

    // Moisture sanity check
    if (moisture < 50 || moisture > 450) return false;

    // Check for stuck sensor (same value 10 times)
    static float lastMoisture = 0;
    static int sameCount = 0;
    if (abs(moisture - lastMoisture) < 1) {
        sameCount++;
    } else {
        sameCount = 0;
    }
    lastMoisture = moisture;

    if (sameCount > 10) return false;  // Sensor stuck

    return true;
}
```

Fallback Mode

If ML model fails, use simple thresholds:

```
bool fallbackDecision(float temp, float moisture) {
    if (temp < 28) return moisture < 256;
    else if (temp < 30) return moisture < 237;
    else if (temp < 32) return moisture < 229;
    else return moisture < 240;
}
```

Summary: Checklist Before Deployment

- [ ] Capacitive soil moisture sensor (NOT resistive)
- [ ] Sensor calibrated in YOUR soil
- [ ] Calibration values mapped to training scale (100-400)
- [ ] DHT22 readings verified against known thermometer
- [ ] irrigation_model.h included in project
- [ ] Reading interval matches training data
- [ ] Hysteresis thresholds configured
- [ ] Pump protection (min/max run times)
- [ ] Fallback mode implemented
- [ ] Tested with simulated scenarios first