Author: Damien Rainbow
Team Member: Anais Ponsonnet
Date: 05/04/2025
55:036 Embedded Systems
Project Title: Offline Smart Plant Care System Report

Introduction

1.a Motivation and Background

In today's fast-paced world, many individuals struggle to maintain regular plant care due to busy schedules or forgetfulness. This often results in overwatering or underwatering, negatively affecting plant health. The **Offline Smart Plant Care System** addresses this problem by automating basic plant care using an embedded microcontroller and environmental sensors without needing an internet connection.

1.b Goals/Specifications

The main objectives of this project are:

- Automatically detect when a plant needs water based on soil moisture levels.
- Display real-time temperature, humidity, and soil moisture readings.
- Notify users visually (LED indicators) of plant status.
- Activate a water pump when moisture levels fall below a threshold.
- Operate entirely offline using a stand-alone embedded system.

Implementation

2.a Overview

The system integrates key hardware components ATmega328P microcontroller, a DHT11 sensor, a soil moisture sensor, a relay, a water pump, an LED, and an LCD display. The software reads and processes sensor inputs, controls irrigation via a relay, and provides real-time feedback through an LCD and LED indicators. All code is written in C using AVR libraries.

2.b Hardware Description

Hardware Components:

- **ATmega328P AVR Microcontroller:** Controls all I/O operations.
- **16x2 LCD Display:** Shows temperature, humidity, and soil moisture values.
- **Soil Moisture Sensor:** Analog sensor connected to ADC pin A0 to monitor soil hydration.
- **DHT11 Sensor:** Digital sensor connected to PD4 to read ambient temperature and humidity.
- **Relay Module (SRD-05VDC-SL-C):** Connected to PD2 to control the water pump.
- **Water Pump & Tube:** Waters the plant when moisture is low.
- **LED (PB5):** Indicates low moisture and blinks during watering.
- **Power battery:** 5V regulated source powering the pumps.
- **Wires & Resistors**

Schematic:

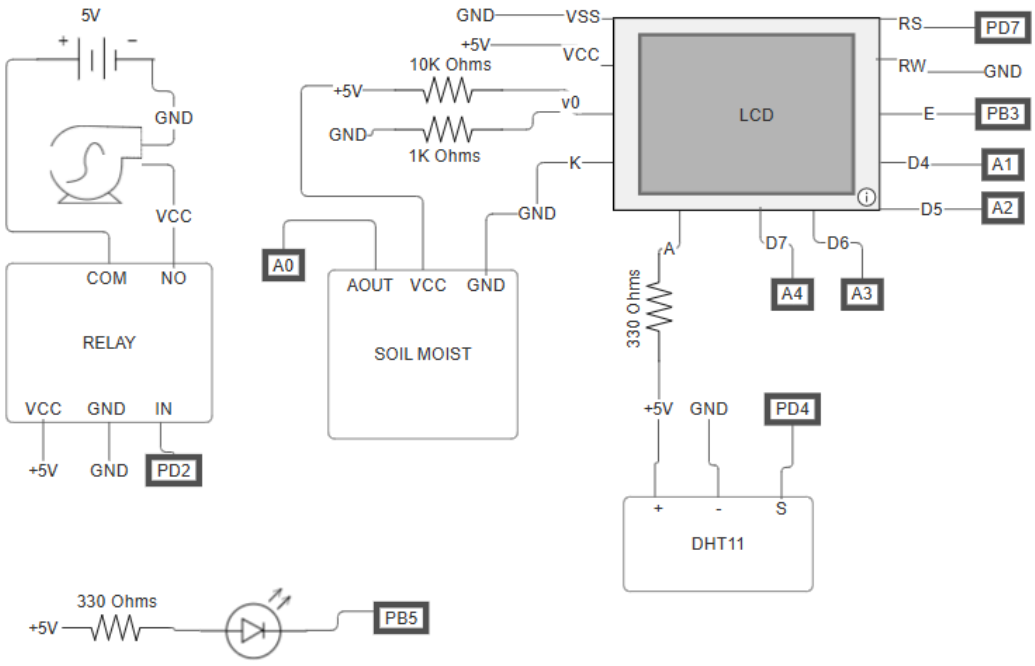


Figure 1: Circuit schematic implemented using SmartDraw

Figure 1 illustrates the complete circuit layout for the Offline Smart Plant Care System, showing how the various components are interconnected through the ATmega328P microcontroller. The 16x2 LCD display is connected to the microcontroller using analog pins A1 to A4, which correspond to data lines D4 to D7 on the LCD. Control signals for the LCD are provided via

digital pins 7 (PD7) and 11 (PB3), connected to the RS and E pins of the display, respectively, allowing real-time data such as soil moisture, temperature, and humidity to be shown to the user.

The soil moisture sensor is connected to analog pin A0, enabling the microcontroller to continuously monitor the water content in the soil. When the moisture level falls below a set threshold, the system sends a signal to a relay module connected to digital pin 2 (PD2), which in turn controls power to a water pump. The +5V from the battery is connected to the COM (common) terminal of the relay, and the Normally Open (NO) terminal is connected to the positive (VCC) input of the pump. The ground (GND) of the pump is connected to the battery's ground. The pump receives power and activates only when the signal from PD2 triggers the relay.

To monitor the ambient environment, a DHT11 temperature and humidity sensor is connected to digital pin 3 (PD4). A visual alert system is implemented using an LED connected to digital pin 13 (PB5), which can light up to indicate conditions such as low moisture. The LED will flash when the system is watering. The entire system, except the pump (powered with a 5V battery), is powered through a 5V supply (Arduino), with appropriate ground connections shared across the components.

2.c Software Description

Flowchart:

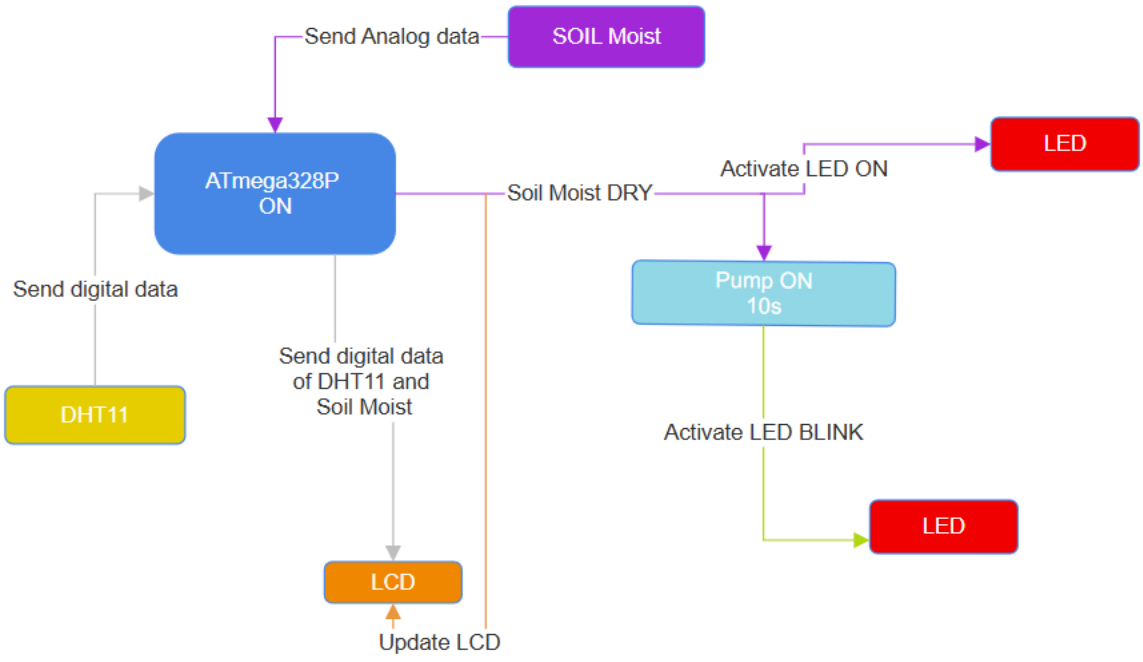


Figure 2: Data flow diagram using SmartDraw

Once the system is powered on, the ATmega328P microcontroller initializes all sensors and peripherals. The data flow begins with sensor readings: the soil moisture sensor outputs an analog voltage corresponding to the water content in the soil, which is read by the ADC (Analog-to-Digital Converter) on analog pin A0. Simultaneously, the DHT11 sensor sends digital signals carrying temperature and humidity data via digital PD3. The microcontroller reads and processes this information at regular intervals every 2 seconds.

After receiving the sensor data, the microcontroller compares the current soil moisture value against a preset threshold. If the soil is adequately moist, no action is taken. However, if the value falls below the threshold, the microcontroller activates digital PD2 to power the water pump. Water is delivered to the plant for a fixed duration of 10 seconds.

Throughout this process, the LCD display (driven via analog pins A1 to A4 and control pins 4 and 11) continuously updates to show real-time values of soil moisture, temperature, and humidity. If the moisture is low, the LED connected to pin 13 is illuminated, and if watering is in progress, it makes the LED blink.

After irrigation, the microcontroller reassesses the moisture level. If it's still below the threshold, the pump will activate again. Otherwise, the system returns to its idle monitoring state, ready to repeat the cycle.

Major Functions

1. Control Module
    - Function: Core loop that orchestrates sensor readings, display updates, and pump control.
    - Key Functions: main(), read_and_display_sensors(), check_soil_and_irrigate().

2. LCD Display Module
    - Purpose: Handles communication with the 16x2 LCD via a 4-bit interface.
    - Functions:
        - lcd_init(): Initializes the display.
        - lcd_write_cmd(), lcd_write_data(): Send commands and data.
        - lcd_gotoxy(), lcd_puts(): Position the cursor and display strings.

3. Soil Moisture Sensor Module
    - Purpose: Reads analog voltage from the soil sensor and converts it to digital.
    - Functions:
        - adc_init(): Set up ADC with appropriate reference voltage and prescaler.
        - adc_read(channel): Returns 10-bit moisture value from A0.

4. DHT11 Sensor Module
    - Purpose: Interfaces with DHT11 to retrieve temperature and humidity.
    - Functions:
        - dht11_read_data(): Triggers data request from the sensor.
        - dht11_get_device(): Initializes sensor structure.
        - Handles retries and error detection.

5. Irrigation Control Module
    - Purpose: Controls the relay connected to the water pump and the status LED.
    - Functions:
        - activate_pump(duration): Turns on the pump via the relay for a specific duration.
        - set_led(status): Turns LED on, off, or blinking to show watering state.

6. Timing and Delay Module
    - Purpose: Manages all timing operations using blocking delays.
    - Functions:
        - _delay_ms(): Used for fixed delays between cycles and watering intervals.
        - Constant values like UPDATE_INTERVAL, WATERING_DURATION define behavior.

Experimental Methods

Action	Expected Outcome	Actual Outcome	Pass/Fail
Power on the system	Temp:24C Hum:60% Moisture: OK	Temp:24C Hum:60% Moisture: OK	Pass
Soil is dry (low moisture level)	Temp:24C Hum:60% Moisture: DRY Watering triggered + LED ON and Blink	Temp:24C Hum:60% Moisture: DRY Watering triggered + LED ON and Blink	Pass
Soil is moist (above threshold)	Temp:24C Hum:60% Moisture: OK No watering trigger	Temp:24C Hum:60% Moisture: OK No watering trigger	Pass
DHT11 sensor reads temperature and humidity	Temp:24C Hum:60% Moisture: OK	Temp:24C Hum:60% Moisture: OK	Pass
LED alert on low moisture	LED turns on	LED turns on	Pass
Low moisture level Watering ON	LED Blink	LED Blink	Pass
Watering Time	Watering for 10s	Watering for 10s	Pass
After watering, the soil changes	Temp:24C Hum:60% Moisture: OK No watering trigger	Temp:24C Hum:60% Moisture: OK No watering trigger	Pass
After watering, the soil doesn't change	Watering for 10s	Watering for 10s	Pass
Relay control signal from microcontroller	Relay switches ON/OFF according to digital output	Relay switches ON/OFF according to digital output	Pass

Figure 3: Test of each characteristic

Results

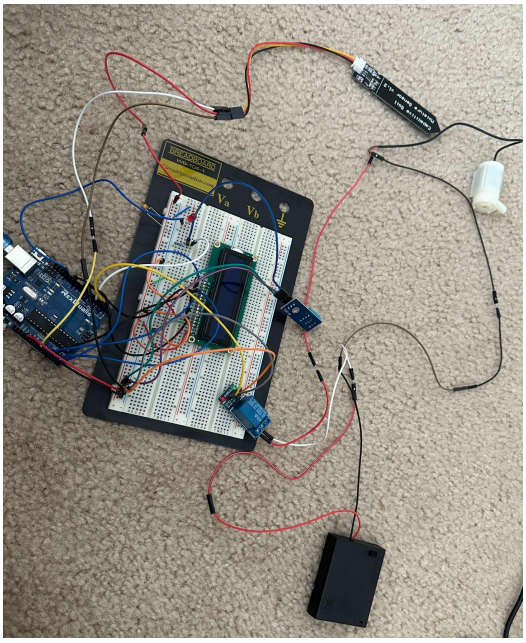


Figure 4: Hardware Setup

This image displays the complete hardware configuration of the Offline Smart Plant Care System, including the microcontroller, sensors, LCD display, relay, and water pump connections.

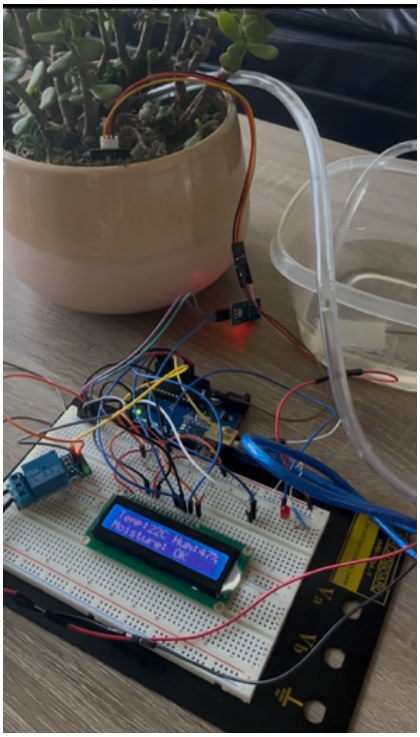
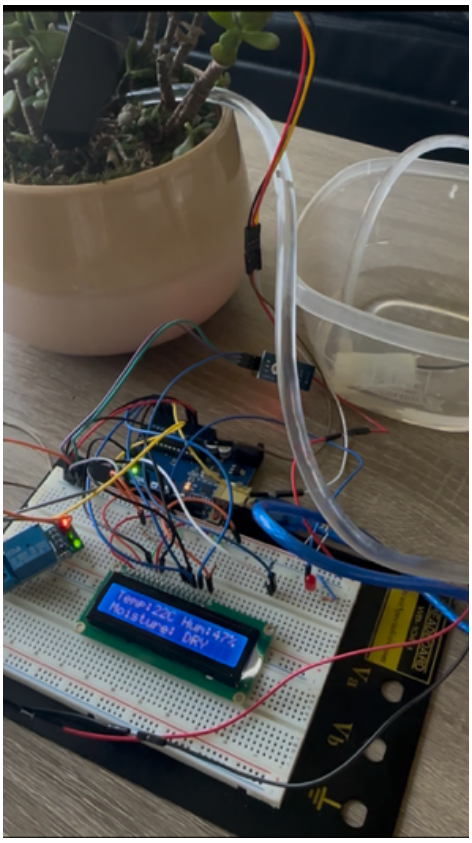


Figure 2: Normal Operating Mode

In this image, the system detects adequate soil moisture. The LCD display shows temperature, humidity, and moisture status as “OK.” No watering is triggered, and the LED remains off.



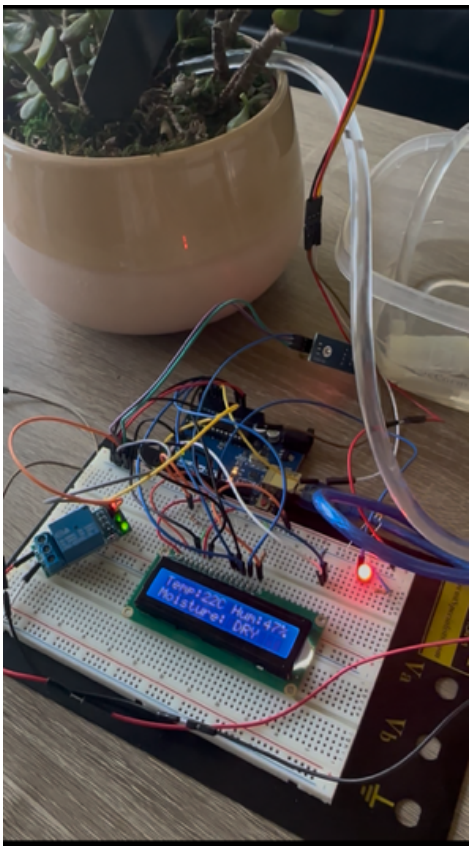


Figure 3: Watering Mode with Alert

Figures 3, with the two pictures together, demonstrate the system’s response when low soil moisture is detected. The relay is activated to trigger the water pump, and the LED blinks as a visual alert to indicate that irrigation is in progress.

Discussion of Results

The Offline Smart Plant Care System met all initial design goals:

- Sensors delivered reliable readings with minimal noise.
- Moisture threshold logic accurately triggered irrigation.
- Relay and pump control worked as expected.
- The LCD provided real-time feedback for users.
- LED notification and blinking behavior correctly represented system states.

Implementation Alternatives

To enhance the reliability and accuracy of the system, capacitive soil moisture sensors could be used instead of resistive ones, offering better longevity and resistance to corrosion. Replacing the DHT11 with a DHT22 sensor would also improve temperature and humidity accuracy. Additionally, implementing low-power modes or sleep cycles could significantly increase energy efficiency, particularly for battery-powered setups.

Current Limitations

The current system uses a fixed watering duration, which may not always provide the optimal amount of water depending on conditions. Incorporating adaptive control logic could make irrigation more efficient. The system also lacks a real-time clock, preventing scheduled watering, and it does not support data logging or memory, limiting historical analysis and performance tracking.

Ideas for Improvement

Future versions of the system include an SD card for local data logging or use wireless connectivity (Bluetooth or Wi-Fi) for syncing data to an external device. Adding a touch interface or buttons would allow users to easily adjust thresholds and system settings. Finally, the system could be expanded to support multiple plant modules, making it suitable for larger or more diverse plant care applications.

Conclusion

Our Offline Smart Plant Care System successfully automated irrigation based on soil moisture readings, while displaying real-time temperature, humidity, and moisture data. The system operated fully offline, making it accessible and reliable. This project has potential applications in both personal and small-scale agricultural settings, especially where internet connectivity is

limited. It promotes sustainable water use and can serve as a foundation for future smart farming solutions or educational demonstrations in embedded systems.

Acknowledgements

Bing Videos
Automatic-Plant-Watering-System-Tutorial/Arduino Plant Watering System/1 Canal Système d'arrosage des plantes à 1 canal avec Arduino UNO R3 - Français.pdf at master · WayinTop/Automatic-Plant-Watering-System-Tutorial
Bing Videos
rq s003 is it a temperature you have S - - in the back - Search Images

References

6.a Datasheets

- DHT11 Sensor: DHT11 Humidity & Temperature Sensor
- SRD-05VDC-SL-C Relay: Cross reference for Songle Relay SRD-05VDC-SL-C - Electromechanical / Relays - DigiKey TechForum - An Electronic Component and Engineering Solution Forum
- Arduino Uno Datasheet ICON
- LCD Datasheet ICON

6.b Software Libraries

DHT sensor library AVR-DHT11/main.c at master · ryanj1234/AVR-DHT11

Appendix-Source Code

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdio.h>
#include "dht11.h"

#define LCD_RS_PORT PORTB
#define LCD_RS_PIN 0
#define LCD_E_PORT PORTB
#define LCD_E_PIN 3
#define LCD_DATA_PORT PORTC
#define LCD_D4_PIN 1
#define LCD_D5_PIN 2
#define LCD_D6_PIN 3
#define LCD_D7_PIN 4
```

```
#define LED_PIN PB5
#define SOIL_SENSOR_CHANNEL 0
#define RELAY_PIN PD1
#define DHT11_PIN 4
#define MOISTURE_THRESHOLD 550 // Calibrate: >550 = DRY, <550 = OK
#define WATERING_DURATION 10000 // 10s
#define UPDATE_INTERVAL 2000 // 2s for testing, revert to 10000 for final

void lcd_write_nibble(uint8_t data) {
	LCD_DATA_PORT &= ~((1 << LCD_D4_PIN) | (1 << LCD_D5_PIN) |
	(1 << LCD_D6_PIN) | (1 << LCD_D7_PIN));
	if (data & 0x01) LCD_DATA_PORT |= (1 << LCD_D4_PIN);
	if (data & 0x02) LCD_DATA_PORT |= (1 << LCD_D5_PIN);
	if (data & 0x04) LCD_DATA_PORT |= (1 << LCD_D6_PIN);
	if (data & 0x08) LCD_DATA_PORT |= (1 << LCD_D7_PIN);
	LCD_E_PORT |= (1 << LCD_E_PIN);
	_delay_us(1000);
	LCD_E_PORT &= ~(1 << LCD_E_PIN);
	_delay_us(1000);
}

void lcd_write_cmd(uint8_t cmd) {
	LCD_RS_PORT &= ~(1 << LCD_RS_PIN);
	lcd_write_nibble(cmd >> 4);
	lcd_write_nibble(cmd & 0x0F);
	_delay_ms(2);
}

void lcd_write_data(uint8_t data) {
	LCD_RS_PORT |= (1 << LCD_RS_PIN);
	lcd_write_nibble(data >> 4);
	lcd_write_nibble(data & 0x0F);
	_delay_ms(2);
}

void lcd_init(void) {
	DDRC |= (1 << LCD_D4_PIN) | (1 << LCD_D5_PIN) |
	(1 << LCD_D6_PIN) | (1 << LCD_D7_PIN);
	DDRB |= (1 << LCD_RS_PIN) | (1 << LCD_E_PIN);
	_delay_ms(1000);
	LCD_RS_PORT &= ~(1 << LCD_RS_PIN);
	lcd_write_nibble(0x03);
	_delay_ms(100);
	lcd_write_nibble(0x03);
	_delay_ms(100);
	lcd_write_nibble(0x03);
	_delay_ms(100);
	lcd_write_nibble(0x02);
	_delay_ms(100);
	lcd_write_cmd(0x28);
	_delay_ms(100);
	lcd_write_cmd(0x0C);
	_delay_ms(100);
	lcd_write_cmd(0x01);
	_delay_ms(100);
	lcd_write_cmd(0x06);
	_delay_ms(100);
}

void lcd_puts(const char *str) {
	while (*str) {
		lcd_write_data(*str++);
	}
}

void lcd_gotoxy(uint8_t x, uint8_t y) {
	uint8_t addr = (y == 0) ? 0x80 : 0xC0;
	addr += x;
	lcd_write_cmd(addr);
}
```

```
void lcd_reset(void) {
	lcd_write_cmd(0x01); // Clear display
	_delay_ms(2);
	lcd_write_cmd(0x0C); // Display on, cursor off
	_delay_ms(2);
	lcd_write_cmd(0x06);
	_delay_ms(2);
}

void adc_init(void) {
	ADMUX = (1 << REFS0); // AVcc as reference
	ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // Enable ADC, prescaler 128
}

uint16_t adc_read(uint8_t ch) {
	ch &= 0b00000111;
	ADMUX = (ADMUX & 0xF8) | ch;
	ADCSRA |= (1 << ADSC);
	while (ADCSRA & (1 << ADSC));
	return ADC;
}

void init_io(void) {
	DDRD |= (1 << RELAY_PIN);
	DDRB |= (1 << LED_PIN);
	PORTD |= (1 << RELAY_PIN); // Relay off (active low)
	PORTB &= ~(1 << LED_PIN);  // LED off
	PORTD |= (1 << PD4);       // DHT11 internal pull-up
}

int main(void) {
	DDRC |= (1 << LCD_D4_PIN) | (1 << LCD_D5_PIN) |
	(1 << LCD_D6_PIN) | (1 << LCD_D7_PIN);
	DDRB |= (1 << LCD_RS_PIN) | (1 << LCD_E_PIN);
	DDRD |= (1 << RELAY_PIN);
	PORTD |= (1 << RELAY_PIN);

	lcd_init();
	adc_init();
	init_io();

	// Initial delay for DHT11 stability
	_delay_ms(3000);

	uint16_t last_temperature = 0, last_humidity = 0;
	uint8_t last_moisture_state = 0; // 0 = OK, 1 = DRY
	uint8_t dht_error_count = 0; // Track consecutive DHT11 errors
	while (1) {
		char buffer[16];
		uint16_t temperature = last_temperature;
		uint16_t humidity = last_humidity;
		uint16_t moisture = 0;

		// Read DHT11 with retries
		_delay_ms(2000); // Ensure 2s between reads
		dht11_device dev = dht11_get_device(DHT11_PIN);
		uint8_t dht_result = DHT11_READ_TIMEOUT;
		for (uint8_t i = 0; i < 3 && dht_result != DHT11_READ_SUCCESS; i++) {
			dht_result = dht11_read_data(dev, &humidity, &temperature);
			if (dht_result != DHT11_READ_SUCCESS) {
				_delay_ms(1000); // Wait before retry
			}
		}
		if (dht_result != DHT11_READ_SUCCESS) {
			dht_error_count++;
			if (dht_error_count >= 3) { // Show error only after 3 consecutive failures
				lcd_reset();
				lcd_gotoxy(0, 0);
				lcd_puts("DHT11 Error     ");
```

```
                lcd_gotoxy(0, 1);
                snprintf(buffer, 16, "Code: %u        ", dht_result);
                lcd_puts(buffer);
                _delay_ms(2000); // Show error briefly
            }
            temperature = last_temperature;
            humidity = last_humidity;
            } else {
            dht_error_count = 0; // Reset error count
            last_temperature = temperature;
            last_humidity = humidity;
        }

        // Read soil moisture
        moisture = adc_read(SOIL_SENSOR_CHANNEL);
        uint8_t current_moisture_state = (moisture > MOISTURE_THRESHOLD) ? 1 : 0;

        // Clear LCD before update if state changes
        if (current_moisture_state != last_moisture_state) {
            lcd_reset(); // Clear display for clean transition
            last_moisture_state = current_moisture_state;
        }

        // Update LCD
        lcd_reset(); // Clear display for smooth update
        lcd_gotoxy(0, 0);
        snprintf(buffer, 16, "Temp:%uC Hum:%u", temperature, humidity);
        lcd_puts(buffer);
        lcd_write_data('%'); // Explicitly write %

        lcd_gotoxy(0, 1);
        if (moisture > MOISTURE_THRESHOLD) {
            snprintf(buffer, 16, "Moisture: DRY   ");
            lcd_puts(buffer);
            PORTB |= (1 << LED_PIN); // LED on for low moisture
            } else {
            snprintf(buffer, 16, "Moisture: OK    ");
            lcd_puts(buffer);
            PORTB &= ~(1 << LED_PIN); // LED off
            PORTD |= (1 << RELAY_PIN); // Relay off
        }

        // Watering logic
        if (moisture > MOISTURE_THRESHOLD) {
            // Activate relay and blink LED
            PORTD &= ~(1 << RELAY_PIN); // Relay on (active low)
            for (uint16_t i = 0; i < WATERING_DURATION / 1000; i++) {
                PORTB |= (1 << LED_PIN);
                _delay_ms(500);
                PORTB &= ~(1 << LED_PIN);
                _delay_ms(500);
            }
            PORTD |= (1 << RELAY_PIN); // Relay off
        }

        _delay_ms(UPDATE_INTERVAL);
    }
}
```