Project Documentation: Temperature, Humidity, and Water Level Monitoring System using ESP32, DHT22, and Water Level Sensor

Table of Contents

Introduction

This project presents the development of a Temperature, Humidity, and Water Level Monitoring System using an ESP32 microcontroller, a DHT22 sensor for environmental data, and an analog water level sensor for liquid measurement. The goal of this system is to provide real-time monitoring of environmental conditions that are critical in applications such as smart agriculture, water tank management, and environmental research.

The ESP32 serves as the core controller, offering built-in Wi-Fi and Bluetooth capabilities, which allow for future scalability such as wireless data transmission or cloud-based logging. The DHT22 sensor is used to accurately measure ambient temperature and humidity, while the water level sensor monitors fluid levels in a container or reservoir. The system is designed to be low-cost, modular, and suitable for embedded and IoT applications.

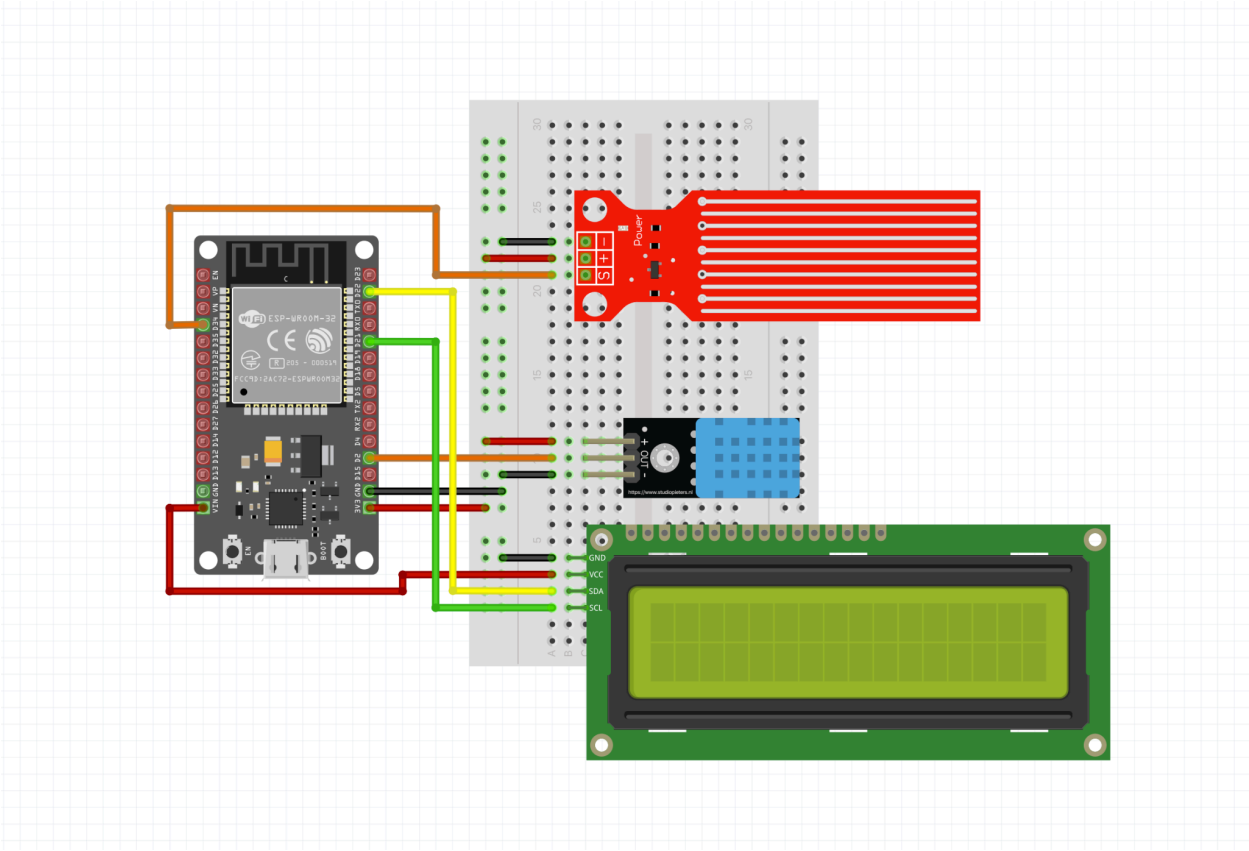
By integrating these components, the system enables reliable and efficient environmental monitoring, supporting both local display and potential remote access features for further development.

Components Used

List of components with specifications:

- **ESP32 Dev Module** – 32-bit dual-core microcontroller with built-in Wi-Fi and Bluetooth
- **DHT11 Sensor** – Digital temperature and humidity sensor (±0.5°C accuracy, 0–100% RH)
- **Analog Water Level Sensor** – Measures liquid level based on resistance
- **OLED Display** (Optional) – For local real-time display (e.g., 0.96” I2C SSD1306)
- **Jumper Wires** – For connections
- **Power Supply** or **USB Cable** – To power the ESP32

Circuit Diagram



Wiring Instructions:

DHT11	Water Level Sensor	LCD Display
VCC → 3.3V on ESP32 DATA → GPIO 4 GND → GND	Signal → GPIO 34 VCC → 3.3V GND → GND	SDA → GPIO 21 SCL → GPIO 22 VCC → 5V GND → GND

Code Explanation

```
#include "thingProperties.h"              // Contains cloud variables and setup for Arduino
Cloud
#include <WiFi.h>                         // Required for ESP32 WiFi functionality
#include <Wire.h>                         // Required for I2C communication
#include <LiquidCrystal_I2C.h>            // Library for I2C LCD control
#include <Adafruit_Sensor.h>              // Core Adafruit sensor library
#include <DHT.h>                          // Library for DHT temperature/humidity sensors

LiquidCrystal_I2C lcd(0x27, 16, 2);       // Initialize LCD at I2C address 0x27 with 16
columns and 2 rows

#define DHTPIN 4                          // DHT11 sensor connected to GPIO 4
#define DHTTYPE DHT11                     // Define sensor type
DHT dht(DHTPIN, DHTTYPE);                 // Initialize DHT sensor

#define WATER_LEVEL_PIN 34                // Water level sensor connected to analog pin GPIO
34

void setup() {
 pinMode(WATER_LEVEL_PIN, INPUT);       // Set water level pin as input
 Serial.begin(115200);                  // Start serial communication for debugging

 lcd.init();                            // Initialize LCD
 lcd.backlight();                       // Turn on LCD backlight
 lcd.print("Connecting...");            // Display connection status

 // Initialize Arduino IoT Cloud and properties
 initProperties();
 ArduinoCloud.begin(ArduinoIoTPreferredConnection);

 // Wait for cloud connection with 15-second timeout
 unsigned long start = millis();
 while (!ArduinoCloud.connected() && millis() - start < 15000) {
   Serial.print(".");
   delay(500);
 }

 lcd.clear();
 if (ArduinoCloud.connected()) {
   lcd.print("Cloud Connected!");       // Success message
 } else {
   lcd.print("Cloud Failed!");          // Failure message
 }

 dht.begin();                           // Start DHT sensor
 delay(2000);                           // Short delay before proceeding

 lcd.clear();
 lcd.print("Ready!");                   // System ready message
}

void loop() {
 ArduinoCloud.update();                 // Keep connection and sync with Arduino Cloud
 updateDisplay();                       // Update LCD display with sensor values
 delay(2000);                           // Refresh every 2 seconds
}

void updateDisplay() {
 float tempC = dht.readTemperature();       // Read temperature in Celsius
 float tempF = dht.readTemperature(true);   // Read temperature in Fahrenheit
 float hum = dht.readHumidity();            // Read humidity

 // If sensor readings are valid, update cloud variables
 if (!isnan(tempC) && !isnan(tempF) && !isnan(hum)) {
     temperature_Celsius = tempC;
     temperature_Farenheit = tempF;
     humidity = hum;
```

```
}

int rawWaterLevel = analogRead(WATER_LEVEL_PIN); // Read raw analog value
water_Level = map(rawWaterLevel, 0, 4095, 0, 100); // Map to 0-100% range

static int lastMode = -1;                        // Keep track of last display mode
if (lastMode != display_Mode) {
    lastMode = display_Mode;
    lcd.clear();                                 // Clear LCD when display mode changes
}

lcd.setCursor(0, 0);                             // Set cursor to first row
switch (display_Mode) {
    case 1:
        lcd.print("Temp: " + String(temperature_Farenheit) + "F"); // Show Fahrenheit
        break;
    case 2:
        lcd.print("Temp: " + String(temperature_Celsius) + "C");   // Show Celsius
        break;
    case 3:
        lcd.print("Humidity: " + String(humidity) + "%");          // Show humidity
        break;
    case 4:
        lcd.print("Water: " + String(water_Level) + "%");          // Show water level
        break;
    default:
        lcd.print("Invalid Mode");                                 // Fallback case
}
}
```

Integration with Arduino Cloud

The monitoring system is integrated with Arduino Cloud to enable real-time remote observation of environmental data such as temperature, humidity, and water level. This is accomplished using the built-in Wi-Fi capabilities of the ESP32 microcontroller, which allows the device to connect directly to the internet and sync data with the Arduino IoT Cloud.

Variables such as temperature_Celsius, temperature_Farenheit, humidity, water_Level, and display_Mode are defined in the cloud platform and are synchronized using the thingProperties.h configuration file. These variables are updated in the main program loop, and the Arduino Cloud automatically transmits the data to a web dashboard where users can visualize readings remotely. This integration provides a user-friendly and scalable solution for IoT-based environmental monitoring without requiring additional services like Blynk.

Testing and Results

The system was successfully assembled and tested for basic functionality. The DHT11 sensor reliably measured both temperature and humidity in an indoor environment, showing responsive changes when exposed to different temperatures (e.g., warm hands or ambient airflow). The LCD displayed accurate readings according to the selected display mode.

The analog water level sensor was tested using finger contact to simulate moisture or water presence. Although this is not a precise method of calibration, it confirmed that the sensor was responsive to conductivity changes. Further testing with actual liquid levels in a container is recommended for more accurate and practical data.

One challenge encountered during development was ensuring stable Wi-Fi and cloud connection within the 15-second timeout. Occasionally, the ESP32 would fail to connect, requiring power cycling or improved Wi-Fi signal. Future improvements could include retry logic and more robust connection handling.

References

Adafruit. (n.d.). *DHT sensor library*. GitHub. https://github.com/adafruit/DHT-sensor-library

Arduino. (n.d.). *Arduino IoT Cloud*. Arduino Documentation. https://docs.arduino.cc/cloud/iot-cloud/

Espressif Systems. (n.d.). *ESP32 technical reference manual*. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

Rickman, J. (n.d.). *LiquidCrystal I2C library*. GitHub. https://github.com/johnrickman/LiquidCrystal_I2C