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.

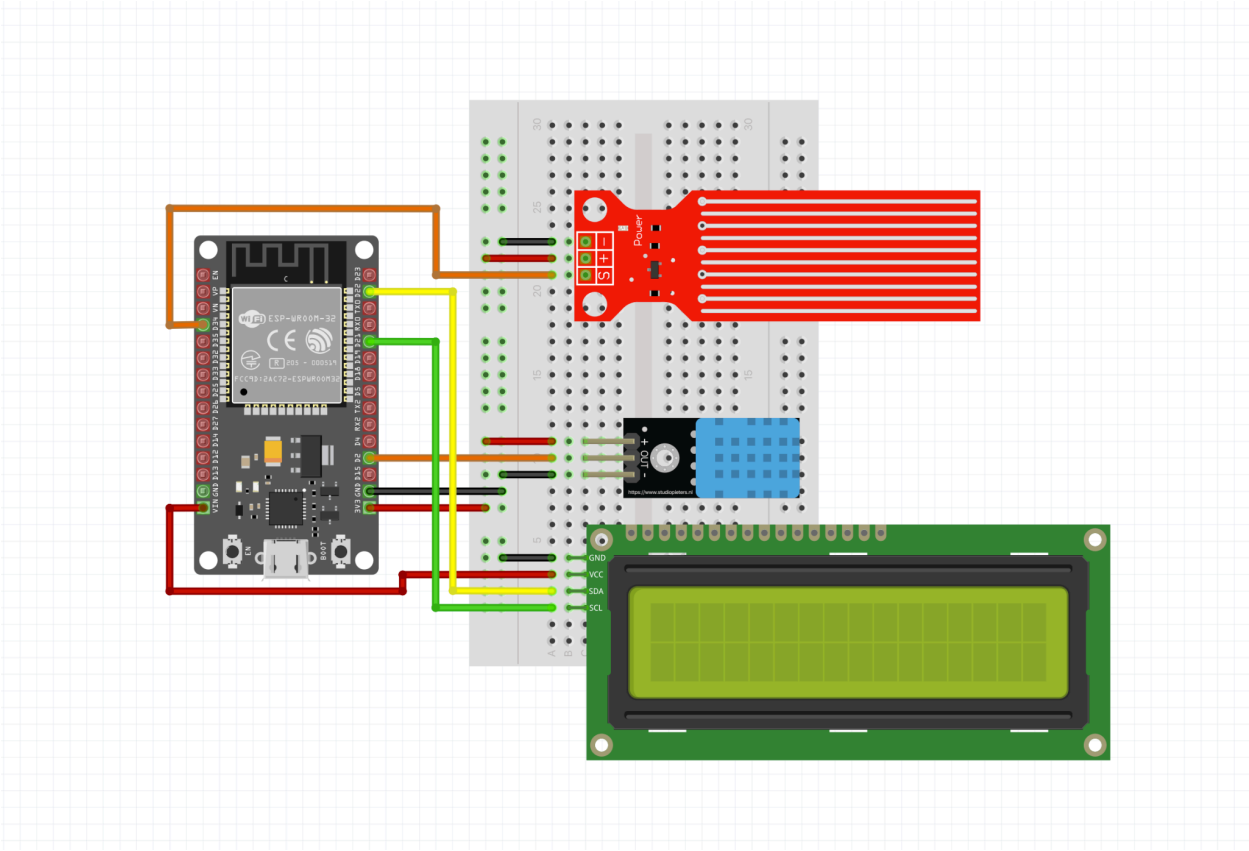
By integrating these components, the system enables reliable and efficient environmental monitoring, supporting both local display and potential remote access features for further development.

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

```
#define BLYNK_TEMPLATE_ID "id"
#define BLYNK_TEMPLATE_NAME "Esp32"
#define BLYNK_AUTH_TOKEN "key"

#include <Wire.h>
#include <LiquidCrystal_I2C.h>         // Library for I2C LCD
#include <Adafruit_Sensor.h>
#include <DHT.h>                       // DHT sensor library
#include <BlynkSimpleEsp32.h>          // Blynk library for ESP32

// Wi-Fi credentials
char ssid[] = "ssid";
char password[] = "pass";

// Initialize 16x2 LCD at I2C address 0x27
LiquidCrystal_I2C lcd(0x27, 16, 2);

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

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

BlynkTimer timer;                      // Create a timer object for periodic tasks
int displayMode = 1;                   // Default LCD display mode

// Handle changes from Blynk (Virtual Pin V4 controls display mode)
BLYNK_WRITE(V4) {
 displayMode = param.asInt();        // Update displayMode from Blynk app
 updateDisplay();                    // Immediately update LCD
}

// Function to read sensors and update both LCD and Blynk app
void updateDisplay() {
 float tempC = dht.readTemperature();          // Celsius
 float tempF = dht.readTemperature(true);      // Fahrenheit
 float humidity = dht.readHumidity();          // Humidity
 int rawWaterLevel = analogRead(WATER_LEVEL_PIN);  // Raw analog input
 float waterLevel = map(rawWaterLevel, 0, 4095, 0, 100); // Convert to % scale

 // Fallback in case of sensor failure
 if (isnan(tempC) || isnan(tempF) || isnan(humidity)) {
   tempC = tempF = humidity = 0;
 }

 // Send sensor data to Blynk virtual pins
 Blynk.virtualWrite(V0, tempC);
 Blynk.virtualWrite(V1, tempF);
 Blynk.virtualWrite(V2, humidity);
 Blynk.virtualWrite(V3, waterLevel);

 // Update the LCD based on selected display mode
 lcd.clear();
 lcd.setCursor(0, 0);
 switch (displayMode) {
   case 1: lcd.print("Temp: " + String(tempF) + "F"); break;
   case 2: lcd.print("Temp: " + String(tempC) + "C"); break;
   case 3: lcd.print("Humidity: " + String(humidity) + "%"); break;
   case 4: lcd.print("Water: " + String(waterLevel)  + "%"); break;
   default: lcd.print("Invalid Mode");
 }
}

void setup() {
 Serial.begin(115200);             // Start serial communication
```

```
 lcd.init();                      // Initialize LCD
 lcd.backlight();                 // Turn on LCD backlight
 lcd.print("Connecting...");      // Initial LCD message

 lcd.clear();
 lcd.print("Starting Inputs!");   // Setup phase message

 Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password); // Connect to Wi-Fi and Blynk
 dht.begin();                     // Initialize DHT sensor
 delay(3000);                     // Delay for sensor stabilization
 pinMode(WATER_LEVEL_PIN, INPUT);  // Set water level pin as input

 lcd.clear();
 lcd.print("Done!");              // Setup complete message

 // Set timer to update display every 2 seconds
 timer.setInterval(2000L, updateDisplay);
}

void loop() {
 Blynk.run();    // Handle Blynk communication
 timer.run();    // Run scheduled tasks
 delay(1000);    // Slight delay (not strictly necessary for Blynk)
}
```

Integration with Blynk

The monitoring system integrates with Blynk, allowing real-time remote access to temperature, humidity, and water level readings from the ESP32. Sensor data is transmitted to the Blynk Cloud via Wi-Fi, where it is displayed on the Blynk mobile app using virtual pins. Users can view updated values every few seconds, making it ideal for monitoring environmental conditions remotely and conveniently.

Additionally, the system supports remote control through Blynk by allowing users to change the LCD display mode using a step slider. This enables swtiching between Fahrenheit, Celsius, humidity, and water level readings without physically interacting with the device. The Blynk integration simplifies the IoT experience by providing a reliable, mobile-first platform without the need for custom web dashboards.

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

Blynk Inc. (n.d.). Blynk IoT platform documentation. https://docs.blynk.io

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