Technical Report: Weather Station Using ESP32

Table of Contents

---

1. Introduction

This project involves building a weather station using two ESP32 microcontrollers to measure and display parameters such as temperature, humidity, light intensity, and time. Communication between the devices is carried via the ESP-NOW protocol. Measurements are displayed on an LCD and alternate between local and remote data.

2. Methodology

The project was divided into the following stages:

1. Component Selection and Wiring:
   a) Selected ESP32 for its wireless capabilities (WiFi and Bluetooth)
   b) Integrated sensors (DHT11 for temperature and humidity, BH1750 for light intensity) and RTC DS1302 for timekeeping.
   c) Connected an LCD (4x20 with I2C interface) for data visualization.

2. Programming:
Two separate programs were written:
   a)Outdoor ESP32: Reads sensor data and transmits it using ESP-NOW.
   b)Indoor ESP32: Displays data on the LCD and handles time synchronization.

3. Testing and Debugging:
   a) Validated data readings from sensors.
   b) Ensured proper communication between ESP32 devices.

3. System Architecture

Components Used:

- ESP32 (x2): Handles communication and data processing.
- DHT11 Sensor (x2): Measures temperature and humidity.
- BH1750 Sensor (x2): Measures light intensity in lux.
- DS1302 RTC (x1): Maintains real-time clock functionality.
- LCD 4x20 with I2C (x1): Displays measurements.

Communication Protocol:

- ESP-NOW: Allows wireless, low-latency communication between the two ESP32 boards.

4. Code Explanation

Outdoor ESP32 Code

1. Initialization: The ESP32 initializes the DHT11 and BH1750 sensors, sets up ESP-NOW, and prepares to transmit data.

```
24  void setup() {
25    // Initialize sensors
26    dht.begin();
27    Wire.begin();
28    lightMeter.begin();
29
30    // Initialize Wi-Fi
31    WiFi.mode(WIFI_STA);
32
33    // Initialize ESP-NOW
34    if (esp_now_init() != ESP_OK) {
35      Serial.println("ESP-NOW Init Failed");
36      return;
37    }
```

2. Registering the master as a peer to exchange data.

```
39    // Register the master as a peer
40    esp_now_peer_info_t peerInfo = {};
41    memcpy(peerInfo.peer_addr, masterMacAddr, 6);
42    peerInfo.channel = 0;
43    peerInfo.encrypt = false;
44
45    if (esp_now_add_peer(&peerInfo) != ESP_OK) {
46      Serial.println("Failed to add peer");
47      return;
48    }
49
50    Serial.begin(115200);
51  }
```

3. Data Collection: Periodically collects temperature, humidity, and light intensity readings.

```
53  void loop() {
54    // Read sensor data
55    float temperature = dht.readTemperature();
56    float humidity = dht.readHumidity();
57    float lux = lightMeter.readLightLevel();
```

4. Data Transmission: Packages data into a structured format and sends it to the Indoor ESP32.

```
59    // Prepare data to send
60    sensorData.temperature = temperature;
61    sensorData.humidity = humidity;
62    sensorData.lux = lux;
63
64    // Send data to the master device
65    esp_err_t result = esp_now_send(masterMacAddr, (uint8_t *)&sensorData, sizeof(sensorData));
66
67    if (result == ESP_OK) {
68      Serial.println("Data sent successfully");
69    } else {
70      Serial.println("Error sending data");
71    }
72
73    delay(1000); // Send data every 10 seconds
74  }
```

Libraries used in OutDoor

#include <DHT.h>
#include <BH1750.h>
#include <Wire.h>
#include <esp_now.h>
#include <WiFi.h>

Indoor ESP32 Code

1. Initialization: The ESP32 initializes the RTC module, LCD, and ESP-NOW for data reception.

```
50  void setup() {
51    // Initialize LCD, RTC, DHT, BH1750, and Wi-Fi
52    Rtc.Begin();
53    lcd.init();
54    lcd.backlight();
55    dht.begin();
56    Wire.begin(); // I2C communication
57    lightMeter.begin();
58
59    // Initialize ESP-NOW
60    WiFi.mode(WIFI_STA);
61    if (esp_now_init() != ESP_OK) {
62      lcd.setCursor(0, 0);
63      lcd.print("ESP-NOW Init Fail");
64      return;
65    }
66
67    // Register the receive callback function for ESP-NOW
68    esp_now_register_recv_cb(onDataReceive);
69
70    delay(1000);
71  }
```

2. Data Reception: Listens for incoming data from the Outdoor ESP32.

```
46  void onDataReceive(const esp_now_recv_info *info, const uint8_t *incomingData, int len) {
47    memcpy(&receivedData, incomingData, sizeof(receivedData));
48  }
```

```
67    // Register the receive callback function for ESP-NOW
68    esp_now_register_recv_cb(onDataReceive);
```

3. Display Logic: Alternates between displaying local sensor readings and data received from the Outdoor ESP32 every 10 seconds. InDoor:

```
73  void loop() {
74    unsigned long currentTime = millis();
75
76    // Switch display data every 10 seconds
77    if (currentTime - lastSwitchTime >= 10000) {
78      displayLocalData = !displayLocalData;
79      lastSwitchTime = currentTime;
80    }
81
82    if (displayLocalData) {
83      // Read local sensor data
84      temperature = dht.readTemperature();
85      humidity = dht.readHumidity();
86      lux = lightMeter.readLightLevel();
87
88      // Format and display local data
89      sprintf(display_dht, "T: %.1f\xDF""C H: %.1f%%", temperature, humidity);
90      if(lux<50){
91        sprintf(display_row3, "Dark %.1f          ", lux);
92      }
93      else if(lux>=50 && lux<500){
94        sprintf(display_row3, "Dusk %.1f          ", lux);
95      }
96      else if(lux>=500){
97        sprintf(display_row3, "Daylight %.1f      ", lux);
98      }
99      sprintf(display_source, "Source: InDoor ");
100   } else {
```

OutDoor:

```
    // Format and display received data
    sprintf(display_dht, "T: %.1f\xDF""C H: %.1f%%", receivedData.temperature, receivedData.humidity);
    if(receivedData.lux<50){
      sprintf(display_row3, "Dark %.1f          ", receivedData.lux);
    }
    else if(receivedData.lux>=50 && receivedData.lux<500){
      sprintf(display_row3, "Dusk %.1f          ", receivedData.lux);
    }
    else if(receivedData.lux>=500){
      sprintf(display_row3, "Daylight %.1f      ", receivedData.lux);
    }
    sprintf(display_source, "Source: OutDoor");
  }
```

4. Displaying date and time for InDoor and OutDoor

```
  lcd.setCursor(0, 1);
  lcd.print(display_dht);
  lcd.setCursor(0, 2);
  lcd.print(display_row3);
  lcd.setCursor(0, 3);
  lcd.print(display_source);

  // Display datetime
  RtcDateTime dt = Rtc.GetDateTime();
  sprintf(datetime, "%02u/%02u/%04u %02u:%02u",
          dt.Day(),
          dt.Month(),
          dt.Year(),
          dt.Hour(),
          dt.Minute()
          );

  lcd.setCursor(0, 0);
  lcd.print(datetime);

  delay(1000);
}
```

Libraries used in InDoor

#include <LiquidCrystal_I2C.h>
#include <ThreeWire.h>
#include <RtcDS1302.h>
#include <DHT.h>
#include <BH1750.h>
#include <Wire.h>
#include <esp_now.h>
#include <WiFi.h>

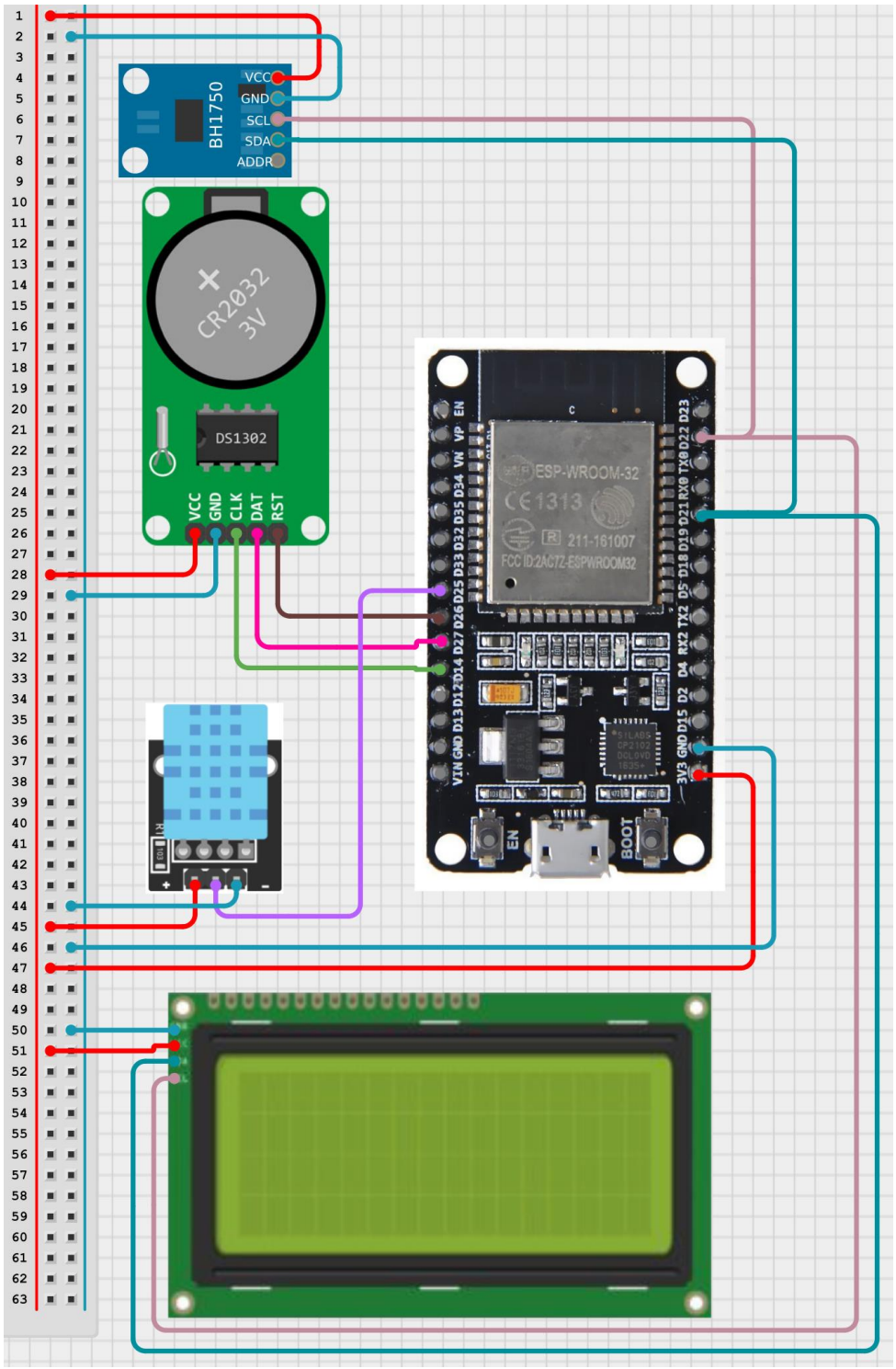
5. System Operation

Setup with assembly schemes:

1. Assemble components according to the wiring diagrams provided below:

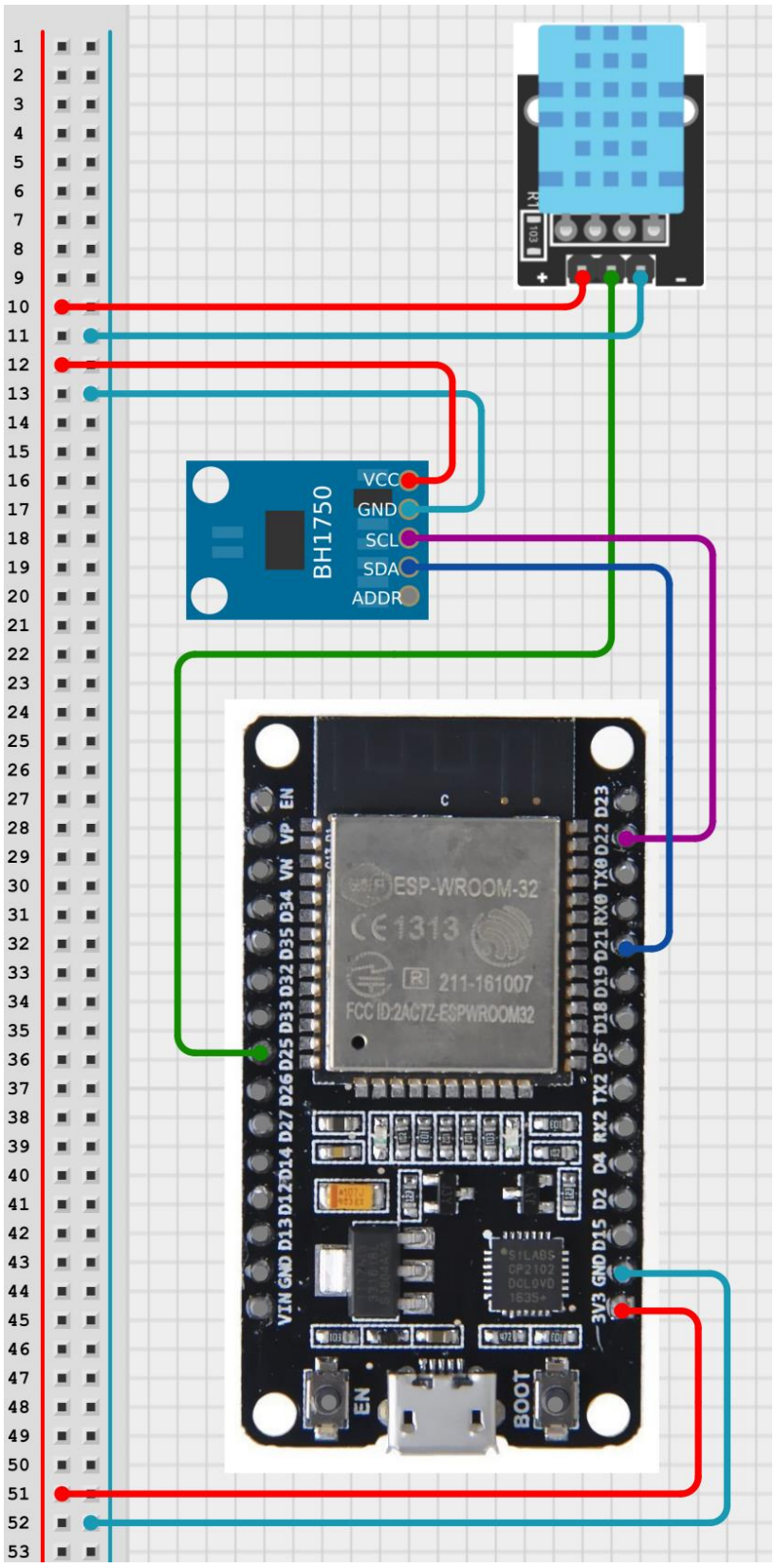
a) Indoor:

- ESP32 with RTC, LCD, DHT11 and BH1750.



b) Outdoor:

- ESP32 with DHT11 and BH1750.



2. Upload the appropriate code to each ESP32 using the Arduino IDE.

Functionality:
a) Data Collection:
 - Outdoor ESP32 continuously measures temperature, humidity, and light intensity.
b) Data Transmission:
 - Data is sent via ESP-NOW to the Indoor ESP32.
c) Data Display:
 - Indoor ESP32 shows both local and remote measurements, switching every 10 seconds.

6. Conclusion

This weather station effectively demonstrates wireless communication between two ESP32 boards and provides real-time environmental monitoring.

Project Created by:

Mateusz Kosek
Jakub Marchewczyk
Wojciech Midura