Wireless Weather Station using ESP32, ESP8266, OLED, and DHT Sensor (with ESP-NOW)

What Is This Project?

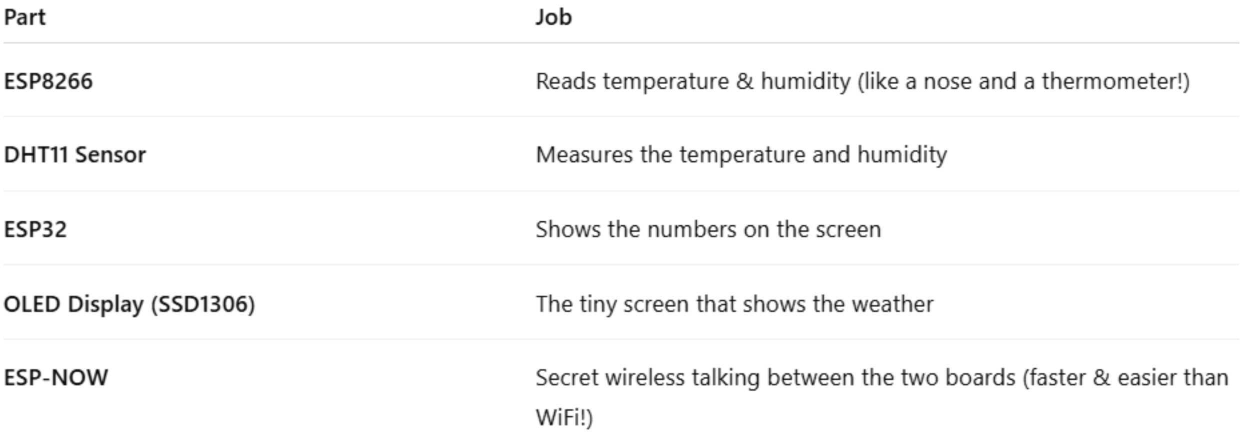
You made a little weather station!

It tells you **temperature and humidity** — like how hot and how wet the air is — using two small computers (called **microcontrollers**):

- **ESP8266** (the one with the sensor)
- **ESP32** (the one with the screen)

The two talk to each other **wirelessly** using a magic called **ESP-NOW**.

What Are the Parts?



Part	Job
ESP8266	Reads temperature & humidity (like a nose and a thermometer!)
DHT11 Sensor	Measures the temperature and humidity
ESP32	Shows the numbers on the screen
OLED Display (SSD1306)	The tiny screen that shows the weather
ESP-NOW	Secret wireless talking between the two boards (faster & easier than WiFi!)

How Does It All Work Together?

The ESP8266:

- Has the **DHT11 sensor** connected to it.
- Read the **temperature** and **humidity**.
- Send that data to the **ESP32** using **ESP-NOW**.

The ESP32:

- Receives the weather data from ESP8266.
- Prints the data on the **OLED screen**.

Step by Step Explanation (For Each Board)

A) ESP8266 (The Sender – with the DHT11)

Step 1: Start Serial Monitor

- To see messages on the computer.

Step 2: Setup Wi-Fi in STA Mode (Station Mode)

- ESP8266 acts like a simple device, not a WiFi access point.
- Important because **ESP-NOW** only works in this mode.

Step 3: Initialize ESP-NOW

- Set up ESP-NOW to start sending data.
- If this fails, it prints “ESP-NOW init failed”.

Step 4: Add the ESP32 as a “friend” (Peer)

- You told the ESP8266 the MAC Address of the ESP32:
 F4:65:0B:4A:83:E0
- This is like telling it: “Hey, only send messages to this buddy!”

Step 5: Read DHT11 Sensor

- Measure **temperature** and **humidity**.
- If reading fails (sensor error), it waits and tries again.

Step 6: Send Data Using ESP-NOW

- Packages the temperature and humidity into a little box (called a “struct”).
- Send this package to the ESP32 friend.

Step 7: Wait and Repeat

- Wait 2 seconds, then do everything again (so it keeps updating).

B) ESP32 (The Receiver – with the OLED Display)

Step 1: Start Serial Monitor

- Prints info on the computer screen.

Step 2: Setup OLED Display

- Start the tiny screen.
- If the screen fails to start, it prints “SSD1306 allocation failed” and stops.

Step 3: Setup Wi-Fi in STA Mode

- Like the ESP8266, it needs to be ready for ESP-NOW.

Step 4: Initialize ESP-NOW

- Start listening for messages.
- If this fails, it prints “ESP-NOW init failed”.

Step 5: Register Receive Callback

- Tells ESP32:
 “When you get data, run this special function” (called `OnDataRecv()`).

Step 6: Handle Received Data (OnDataRecv)

- When ESP32 receives weather data:
    1. It unpacks the little box (struct) to get temperature & humidity.
    2. Prints the data in Serial Monitor.
    3. Shows the data on OLED screen:
        - Big numbers showing **temperature** and **humidity**!

Step 7: Do Nothing in Loop

- No need to repeat or check — it waits for ESP8266 to send data.

5. How ESP-NOW Helps?

ESP-NOW is a cool feature:

- No WiFi router needed!
- No passwords or networks.
- They talk directly, like walkie-talkies!
- Very fast and low power.

You gave each board the other's **MAC Address**:

- So they know **exactly who to send to**.
- Like whispering to only your best friend in class.

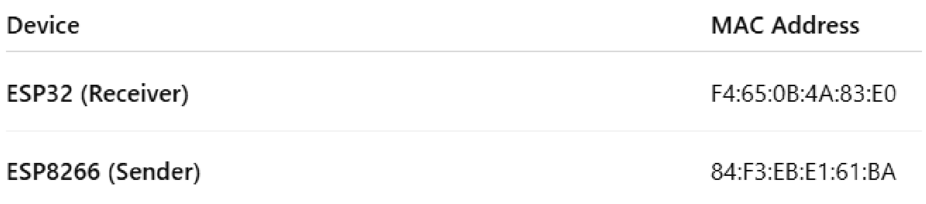
6. What Happens When You Turn It On?

1. **ESP8266** reads temperature & humidity.
2. Send this data via ESP-NOW to the ESP32.
3. **ESP32** receives this data.
4. Prints it on the **OLED screen**.
5. Every 2 seconds — repeat!

Important Notes:

- DHT11 is not super accurate but fine for demo.
- ESP32 and ESP8266 need to be close (or signal may be lost).
- The OLED I2C address is **0x3C** — correct in your code.
- ESP-NOW works best if WiFi is disconnected first (you did this!).

8. Your MAC Addresses Used Correctly:

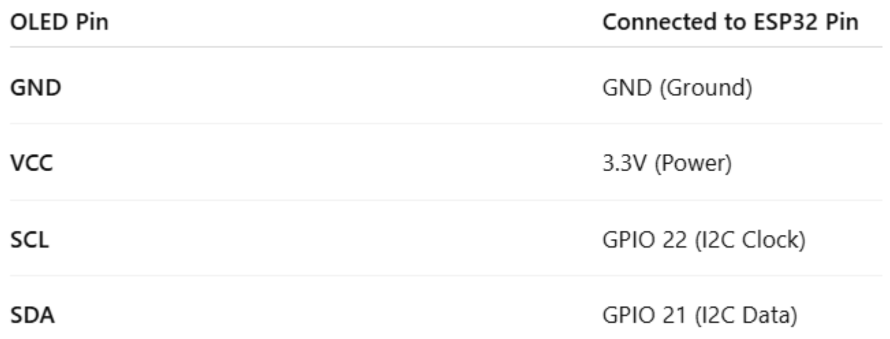


Device	MAC Address
ESP32 (Receiver)	F4:65:0B:4A:83:E0
ESP8266 (Sender)	84:F3:EB:E1:61:BA

Pin Connections for Wireless Weather Station

ESP32 WROOM-32 Dev Kit (Receiver with OLED Display)

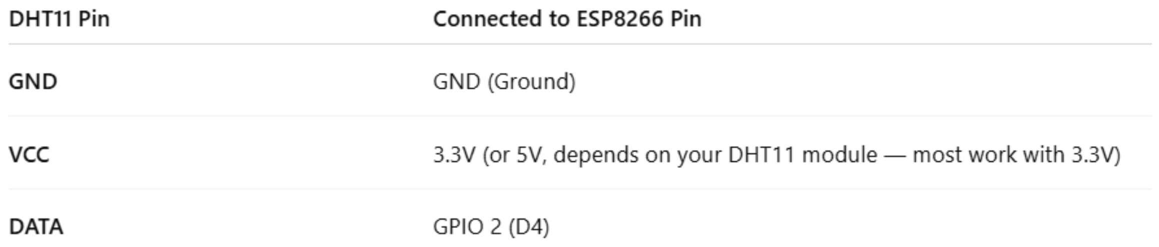
OLED Display (SSD1306) — I2C Connection:



OLED Pin	Connected to ESP32 Pin
GND	GND (Ground)
VCC	3.3V (Power)
SCL	GPIO 22 (I2C Clock)
SDA	GPIO 21 (I2C Data)

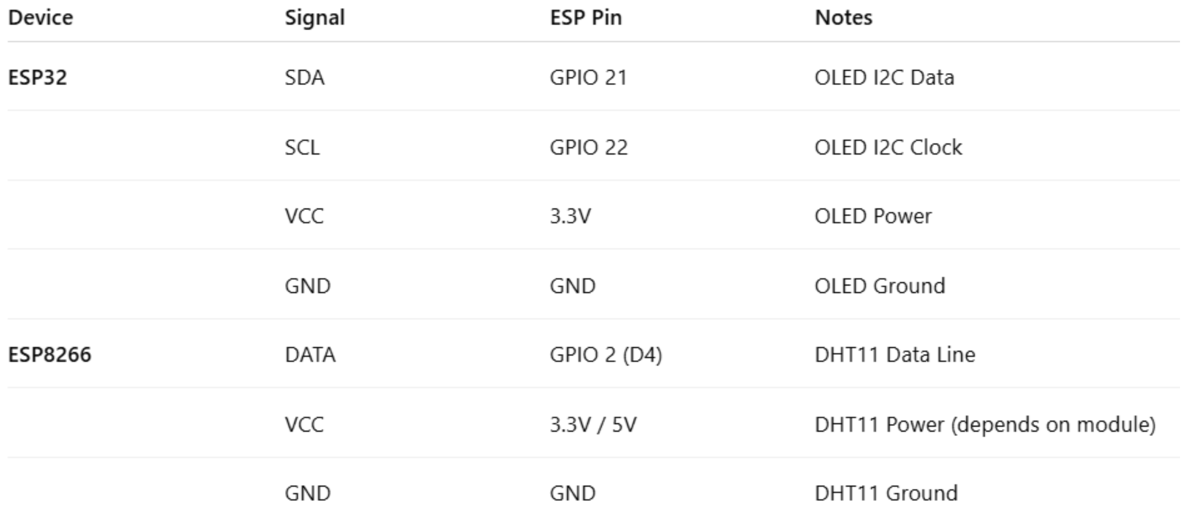
ESP8266 Dev Kit (Sender with DHT11 Sensor)

DHT11 Sensor Connection:



DHT11 Pin	Connected to ESP8266 Pin
GND	GND (Ground)
VCC	3.3V (or 5V, depends on your DHT11 module — most work with 3.3V)
DATA	GPIO 2 (D4)

Summary Table: All Connections at a Glance



Device	Signal	ESP Pin	Notes
ESP32	SDA	GPIO 21	OLED I2C Data
	SCL	GPIO 22	OLED I2C Clock
	VCC	3.3V	OLED Power
	GND	GND	OLED Ground
ESP8266	DATA	GPIO 2 (D4)	DHT11 Data Line
	VCC	3.3V / 5V	DHT11 Power (depends on module)
	GND	GND	DHT11 Ground

Detailed Line-by-Line Explanation Document: Wireless Weather Station Project Code

Part 1: ESP32 Code (Receiver with OLED)

```
#include <Arduino.h>
#include <WiFi.h>
#include <esp_now.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

typedef struct struct_message {
  float temperature;
  float humidity;
} struct_message;

struct_message incomingData;

// ESP8266 Sender MAC Address
uint8_t senderMac[] = {0x84, 0xF3, 0xEB, 0xE1, 0x61, 0xBA};

void OnDataRecv(const uint8_t * mac, const uint8_t *incomingDataBytes, int len) {
  memcpy(&incomingData, incomingDataBytes, sizeof(incomingData));
  Serial.printf("Received => Temp: %.2f°C, Hum: %.2f%%\n", incomingData.temperature, incomingData.humidity);
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.printf("T %.2f C", incomingData.temperature);

  display.setCursor(0, 30);
  display.printf("H %.2f %%", incomingData.humidity);
  display.display();
}

void setup() {
  Serial.begin(115200);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // 0x3C is common OLED I2C addr
    Serial.println(F("SSD1306 allocation failed"));
    while(1);
  }
  display.clearDisplay();
  display.display();

  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  if (esp_now_init() != ESP_OK) {
    Serial.println("ESP-NOW init failed");
    return;
  }



  esp_now_register_recv_cb(OnDataRecv);

}



void loop() {

  // Nothing here; everything happens in callback.

}
```

Header Files (Libraries)

```
#include <Arduino.h>
```

This lets us use the **basic Arduino functions** like `setup()` and `loop()`.

```
#include <WiFi.h>
```

Needed because **ESP-NOW** requires the **WiFi hardware** to work (even if not using the internet).

```
#include <esp_now.h>
```

This is the magic **ESP-NOW library** — allows the two boards to talk without WiFi.

```
#include <Wire.h>
```

Let us use the **I2C communication** protocol — required to talk to the OLED screen.

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

These are libraries to control the **OLED display** — they help print things like text or shapes.

OLED Display Settings

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
```

Tells the program the size of the OLED screen: **128 pixels wide, 64 pixels tall**.
OLED_RESET is not used here, so set to `-1`.

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

Creates an object called `display` — lets us control the OLED easily.

ESP-NOW Data Structure

```
typedef struct struct_message {
  float temperature;
  float humidity;
} struct_message;
```

We create a **box** (called a “struct”) to hold two pieces of data:

1. Temperature (`float`)
2. Humidity (`float`)

```
struct_message incomingData;
```

Makes an **actual box (variable)** to store the data when received.

Sender MAC Address (ESP8266)

```
uint8_t senderMac[] = {0x84, 0xF3, 0xEB, 0xE1, 0x61, 0xBA};
```

The **MAC address** of the ESP8266 (the sender). ESP32 checks this to see who sent the data.

ESP-NOW Receive Callback

```
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingDataBytes, int len) {
```

This special function runs **automatically** when ESP32 gets a message via ESP-NOW.

```
memcpy(&incomingData, incomingDataBytes, sizeof(incomingData));
```

Copies the received data into `incomingData` — unpacks the temperature & humidity.

```
Serial.printf("Received => Temp: %.2f°C, Hum: %.2f%%\n", incomingData.temperature,
incomingData.humidity);
```

Prints received temperature and humidity to the **Serial Monitor**.

```
display.clearDisplay();
```

Clears whatever is on the OLED screen.

```
display.setTextSize(2);
display.setTextColor(WHITE);
display.setCursor(0,0);
```

Sets **big white text**, starts drawing at the **top-left corner**.

```
display.printf("T %.2f C", incomingData.temperature);
```

Prints temperature on screen.

```
display.setCursor(0, 30);
display.printf("H %.2f %%", incomingData.humidity);
```

Prints humidity below temperature.

```
display.display();
```

Sends the text to the OLED — now visible!

Setup Function

```
void setup() {
  Serial.begin(115200);
```

Starts the serial communication (for debugging on computer).

```
 if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
```

Starts the OLED display at I2C address **0x3C**.

```
   Serial.println(F("SSD1306 allocation failed"));
    while(1);
```

If the OLED fails to start, print error and stop everything forever.

```
 display.clearDisplay();
  display.display();
```

Clear any old stuff from OLED.

```
 WiFi.mode(WIFI_STA);
  WiFi.disconnect();
```

Set WiFi to **STA (Station)** mode — important for ESP-NOW.

```
 if (esp_now_init() != ESP_OK) {
    Serial.println("ESP-NOW init failed");
    return;
  }
```

Start ESP-NOW. If it fails — print error and stop setup.

```
 esp_now_register_recv_cb(OnDataRecv);
```

Tell ESP-NOW to use our **OnDataRecv() function** when a message is received.

}

Loop Function

```
void loop() {
  // Nothing here; everything happens in callback.
}
```

Loop is empty — all work happens when a message comes in!

Part 2: ESP8266 Code (Sender with DHT11)

```
#include <ESP8266WiFi.h>
#include <espnow.h>
#include <DHT.h>

#define DHTPIN 2       // GPIO2 (D4)
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

typedef struct struct_message {
  float temperature;
  float humidity;
} struct_message;

struct_message sensorData;

// ESP32 Receiver MAC Address (ESP32 WROOM-32 Dev Kit)
uint8_t receiverMac[] = {0xF4, 0x65, 0x0B, 0x4A, 0x83, 0xE0};

void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Send Status: ");
  Serial.println(sendStatus == 0 ? "Success" : "Fail");
}

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  dht.begin();

  if (esp_now_init() != 0) {
    Serial.println("ESP-NOW init failed");
    return;
  }

  esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);
  esp_now_add_peer(receiverMac, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);
  esp_now_register_send_cb(OnDataSent);
}

void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();

  if (isnan(temp) || isnan(hum)) {
    Serial.println("DHT11 Read Failed");
    delay(2000);
    return;
  }

  sensorData.temperature = temp;
  sensorData.humidity = hum;

  esp_now_send(receiverMac, (uint8_t *)&sensorData, sizeof(sensorData));

  Serial.printf("Sent => Temp: %.2f°C, Hum: %.2f%%\n", temp, hum);

  delay(2000);

}
```

Header Files (Libraries)

```
#include <ESP8266WiFi.h>
#include <espnow.h>
#include <DHT.h>
```

Libraries for:

1. **ESP8266 WiFi**
2. **ESP-NOW communication**
3. **DHT sensor control**

DHT11 Sensor Settings

```
#define DHTPIN 2
#define DHTTYPE DHT11
```

DHT sensor is connected to GPIO2 (D4 pin).
Sensor type is **DHT11**.

```
DHT dht(DHTPIN, DHTTYPE);
```

Create DHT sensor object for reading temperature & humidity.

ESP-NOW Data Structure

```
typedef struct struct_message {
  float temperature;
  float humidity;
} struct_message;
```

Like ESP32 — create a box (struct) for temperature & humidity.

```
struct_message sensorData;
```

Actual box to hold sensor readings.

Receiver MAC Address (ESP32)

```
uint8_t receiverMac[] = {0xF4, 0x65, 0x0B, 0x4A, 0x83, 0xE0};
```

MAC address of **ESP32** — so ESP8266 knows who to send data to.

ESP-NOW Send Callback

```
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
```

This function runs **after sending** data.

```
Serial.print("Send Status: ");
Serial.println(sendStatus == 0 ? "Success" : "Fail");
```

Print whether sending succeeded.

Setup Function

```
void setup() {
  Serial.begin(115200);
```

Start serial communication.

```
 WiFi.mode(WIFI_STA);
  WiFi.disconnect();
```

Set to **STA Mode** — needed for ESP-NOW.

```
 dht.begin();
```

Start DHT11 sensor.

```
 if (esp_now_init() != 0) {
    Serial.println("ESP-NOW init failed");
    return;
  }
```

Start ESP-NOW — print error if failed.

```
 esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);
```

Set this ESP8266 as the **controller** (sender).

```
 esp_now_add_peer(receiverMac, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);
```

Add ESP32 as the **receiver** (peer).

```
 esp_now_register_send_cb(OnDataSent);
```

Use the **OnDataSent() function** to check if sending worked.

}

Loop Function

```
void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
```

Read **temperature and humidity** from DHT11.

```
if (isnan(temp) || isnan(hum)) {
    Serial.println("DHT11 Read Failed");
    delay(2000);
    return;
  }
```

If the sensor fails, print error, wait 2 seconds, skip sending.

```
 sensorData.temperature = temp;
  sensorData.humidity = hum;
```

Store readings in our **struct box**.

```
 esp_now_send(receiverMac, (uint8_t *)&sensorData, sizeof(sensorData));
```

Send data to ESP32 via ESP-NOW.

```
 Serial.printf("Sent => Temp: %.2f°C, Hum: %.2f%%\n", temp, hum);
  delay(2000);
}
```

Print what was sent, wait 2 seconds, repeat.

Final Summary:

- **ESP8266** measures weather, sends it wirelessly to **ESP32**.
- **ESP32** receives data, shows it on an OLED screen.
- No internet or WiFi router needed — just **ESP-NOW magic**!