

Sending Data from ESP32 to Firebase

Introduction

This report explains how we implemented data transmission from an ESP32 microcontroller to the Firebase Realtime Database using Arduino IDE. Initially, we sent test data to ensure proper connectivity, and later, we modified the firmware to read data from a **DHT22 sensor** and upload temperature, humidity, and timestamp values to Firebase.

1. Initial Setup

Before programming the ESP32, we set up the necessary tools and libraries.

1.1 Required Software and Libraries

- **Arduino IDE** (For writing and uploading code to ESP32)
- **Libraries to install in Arduino IDE:**
 - **WiFi.h** (For ESP32 Wi-Fi connectivity)
 - **FirebaseESP32.h** (For Firebase communication)
 - **DHT.h** (For interfacing with the DHT22 sensor)
 - **time.h** (For timestamp formatting)

1.2 Setting up Firebase Realtime Database

1. Go to [Firebase Console](#) and sign in with your Google account.
2. Click on **Add Project** and enter a suitable project name.
3. Follow the setup process and enable **Realtime Database**. (See detailed setup guide from **FIREBASE SETUP GUIDE**)
4. Get the **API Key** and **Database URL** from Firebase.

2. Sending Test Data to Firebase

Once Firebase was set up, we tested connectivity by sending random values to the database.

2.1 Code Explanation

Including Required Libraries

```
#include <WiFi.h>
#include <FirebaseESP32.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"
```

These libraries handle **Wi-Fi connectivity** and **Firestore communication**.

Defining Constants (Replace Credentials)

```
#define WIFI_SSID "INSERT_YOUR_WIFI_SSID"
#define WIFI_PASSWORD "INSERT_YOUR_WIFI_PASSWORD"
#define API_KEY "INSERT_YOUR_API_KEY"
#define DATABASE_URL "INSERT_YOUR_DATABASE_URL"
```

These define the Wi-Fi credentials and Firebase API details (which must be replaced with actual values).

Connecting to Wi-Fi

```
void Wifi_Init() {
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to Wi-Fi");
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(300);
    }
    Serial.println("\nConnected with IP: " +
WiFi.localIP());
}
```

This function attempts to connect the ESP32 to the defined Wi-Fi network.

Connecting to Firebase

```
void firebase_init() {
    config.api_key = API_KEY;
    config.database_url = DATABASE_URL;
    Firebase.reconnectWiFi(true);

    Serial.println("Signing up new user...");
    if (Firebase.signUp(&config, &auth, "", "")) {
        Serial.println("Sign-up successful");
        isAuthenticated = true;
    } else {
        Serial.printf("Sign-up failed: %s\n",
config.signer.signupError.message.c_str());
        isAuthenticated = false;
    }

    config.token_status_callback = tokenStatusCallback;
    Firebase.begin(&config, &auth);
}
```

This function initializes Firebase and attempts to sign up the ESP32.

Sending Test Data

```
void database_test() {
    if (millis() - elapsedMillis > update_interval &&
        isAuthenticated && Firebase.ready()) {
        elapsedMillis = millis();

        String node = databasePath + "/value";
        if (Firebase.set(fbdo, node.c_str(), count++)) {
            Serial.println("Data sent successfully");
        } else {
            Serial.println("Data upload failed: " +
                fbdo.error_reason());
        }
    }
}
```

This function sends test data to Firebase every **10 seconds**.

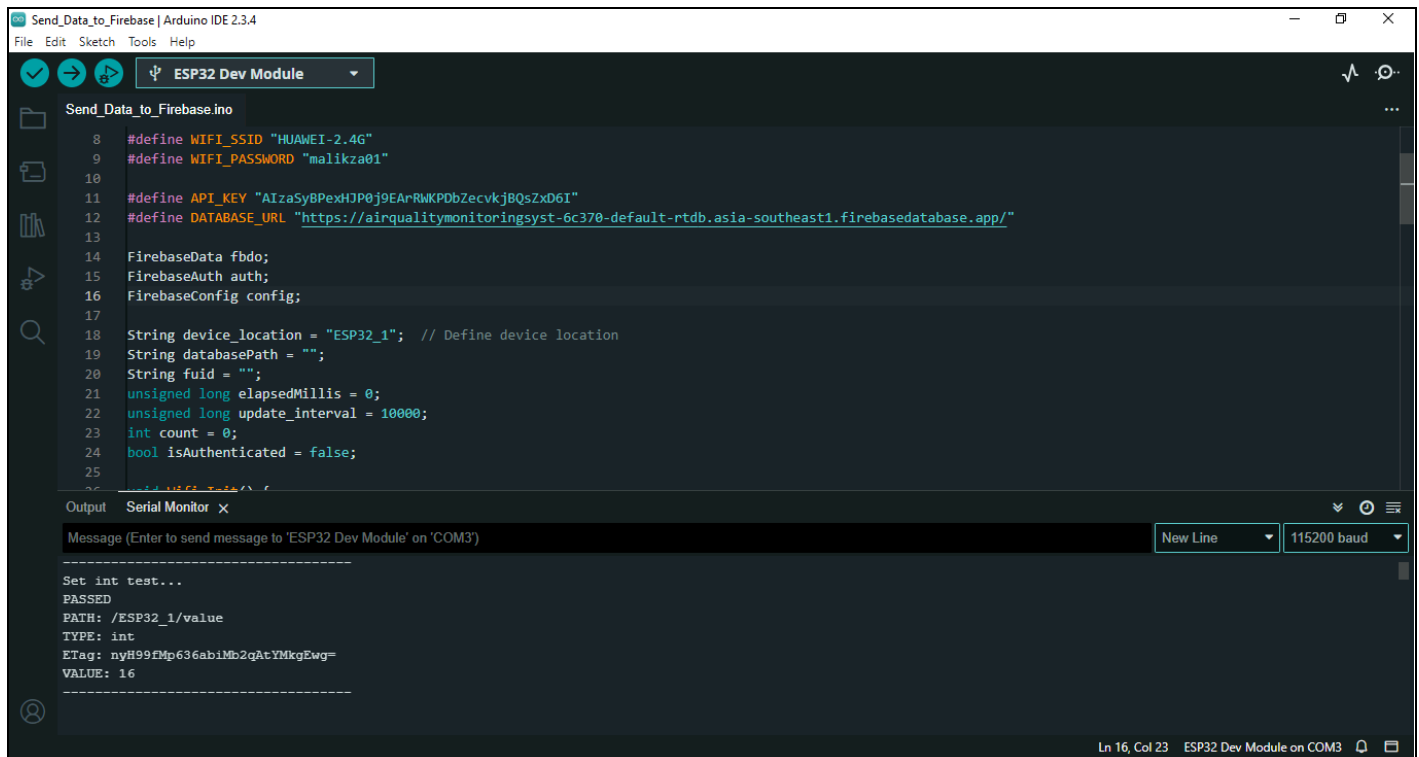
Main Program

```
void setup() {
    Serial.begin(115200);
    Wifi_Init();
    firebase_init();
}

void loop() {
    database_test();
}
```

The **setup()** function initializes Wi-Fi and Firebase, while **loop()** continuously sends test data.

Our code is all set for us to perform our dummy test. Connect your ESP32 board to your USB ports and upload your code.



The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar shows icons for compiling, uploading, and opening the serial monitor. The sketch is named 'Send_Data_to_Firebase.ino' and is for an 'ESP32 Dev Module'. The code defines WiFi credentials, Firebase API key, and database URL. It initializes FirebaseData, FirebaseAuth, and FirebaseConfig. It also defines device location, database path, and update interval. The Serial Monitor shows the output of the code, including the device location, database path, and the value of the 'test' variable.

```
8 #define WIFI_SSID "HUAWEI-2.4G"
9 #define WIFI_PASSWORD "malikza01"
10
11 #define API_KEY "AIzaSyBPexHJP0j9EArRMKPDbZecvkjBQsZxD6I"
12 #define DATABASE_URL "https://airqualitymonitoringsyst-6c370-default-rtdb.asia-southeast1.firebaseio.com/"
13
14 FirebaseData fbdo;
15 FirebaseAuth auth;
16 FirebaseConfig config;
17
18 String device_location = "ESP32_1"; // Define device location
19 String databasePath = "";
20 String fuid = "";
21 unsigned long elapsedMillis = 0;
22 unsigned long update_interval = 10000;
23 int count = 0;
24 bool isAuthenticated = false;
25
```

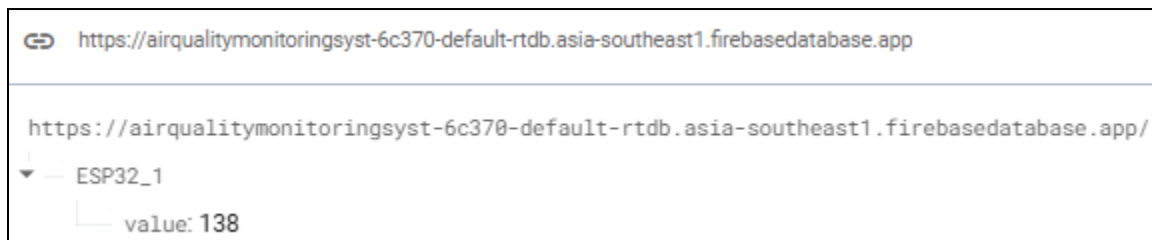
Output Serial Monitor x

Message (Enter to send message to 'ESP32 Dev Module' on 'COM3') [New Line] [115200 baud]

Set int test...
PASSED
PATH: /ESP32_1/value
TYPE: int
ETag: nyH99fMtp636abiMb2qAtYMkgEwg=
VALUE: 16

Ln 16, Col 23 ESP32 Dev Module on COM3

If everything went well, you should be able to go back to the Realtime Database database browser on the Firebase console, and you should see your updates coming every 10 seconds as shown below:



You can also use the serial communication interface to check that the following outputs are being sent every 10 seconds.

```

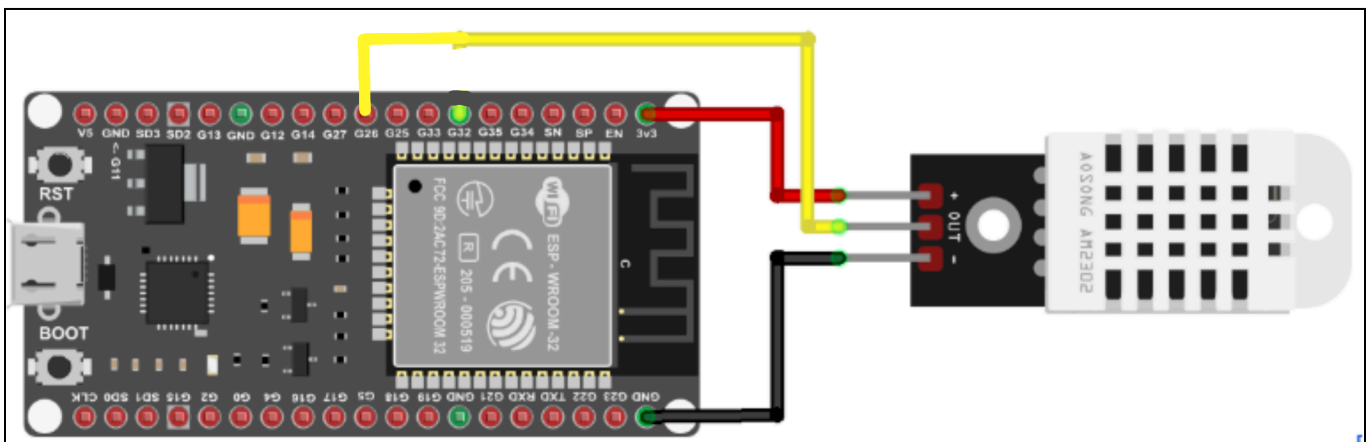
-----
Set int test...
PASSED
PATH: /ESP32_1/value
TYPE: int
ETag: VqXIJ8btVCXsGjrIwUCoY1TB3rg=
VALUE: 235
-----

Set int test...
PASSED
PATH: /ESP32_1/value
TYPE: int
ETag: CM2a8cFxxvJ93xmOBCD0qiIqbFEc=
VALUE: 236
-----

Set int test...
PASSED
PATH: /ESP32_1/value
TYPE: int
ETag: jTkXUH4IXuW4MnW6xJFjadGho9Y=
VALUE: 237
-----

```

3. Reading DHT22 Sensor Data and Sending to Firebase



Once test data was successfully sent, we modified the code to read real sensor data from a **DHT22** sensor and store temperature, humidity, and timestamps in Firebase. The hardware configuration for transmitting DHT22 sensor data to the cloud is not discussed in this document. For details on setting up the DHT22 sensor, refer to the **SENSOR CALIBRATION GUIDE**.

3.1 Additional Libraries for DHT22 and Timestamping

```

#include "DHT.h"
#include <time.h>

```

These libraries help read data from the **DHT22 sensor** and format timestamps.

3.2 Defining DHT22 Sensor Pins

```
#define DHTPIN 26
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

Defines the pin connected to the DHT22 sensor.

3.3 Reading Sensor Data

```
void updateSensorReadings() {
    temperature = dht.readTemperature();
    humidity = dht.readHumidity();

    if (isnan(temperature) || isnan(humidity)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    Serial.printf("Temperature: %.2f°C\n", temperature);
    Serial.printf("Humidity: %.2f%%\n", humidity);
}
```

This function reads temperature and humidity values from the sensor.

3.4 Formatting Timestamps

```
String getFormattedTimestamp() {
    time_t now = Firebase.getCurrentTime();
    if (now > 0) {
        struct tm *timeinfo = gmtime(&now);
        char buffer[25];
        strftime(buffer, sizeof(buffer),
"%Y-%m-%dT%H:%M:%SZ", timeinfo);
        return String(buffer);
    }
    return "0000-00-00T00:00:00Z";
}
```

This function retrieves and formats timestamps from Firebase.

3.5 Uploading Sensor Data

```
void uploadSensorData() {
    if (millis() - elapsedMillis > update_interval &&
isAuthenticated && Firebase.ready()) {
        elapsedMillis = millis();
    }
}
```

```

        updateSensorReadings();

        String timestamp = getFormattedTimestamp();
        FirebaseJson sensorData;
        sensorData.set("temperature", temperature);
        sensorData.set("humidity", humidity);
        sensorData.set("timestamp", timestamp);

        String nodePath = databasePath + "/" + timestamp;
        if (Firebase.setJSON(fbdo, nodePath.c_str(),
sensorData)) {
            Serial.println("Data uploaded successfully!");
        } else {
            Serial.println("Upload failed: " +
fbdo.errorReason());
        }
    }
}

```

This function uploads temperature, humidity, and timestamp data to Firebase.

Final Program

```

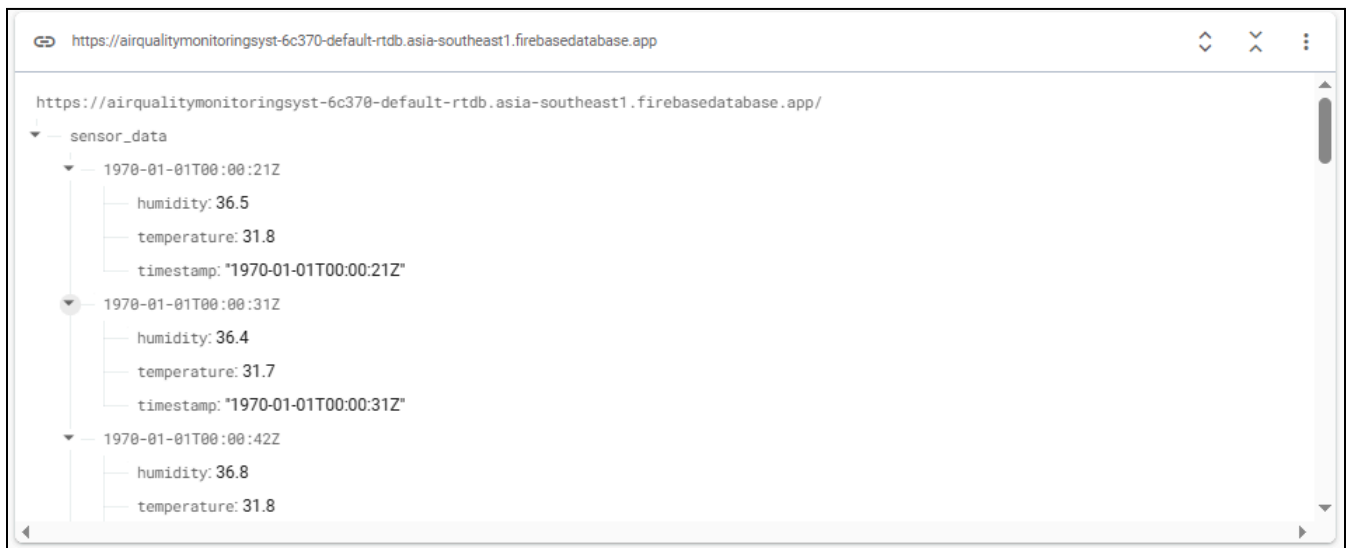
void setup() {
    Serial.begin(115200);
    Wifi_Init();
    firebase_init();
    dht.begin();
}

void loop() {
    uploadSensorData();
}

```

The `setup()` function initializes everything, and `loop()` continuously uploads sensor data.

Upload the code on your ESP32 and see the results.



Conclusion

This implementation successfully connects an **ESP32** to the **Firebase Realtime Database** and uploads **real-time sensor data**. Initially, we tested with random values and later integrated **DHT22 sensor readings** to store temperature, humidity, and timestamps in the cloud.