

---

# ENVIRONMENTAL MONITORING SYSTEM FOR POULTRY MANAGEMENT

Prepared by: Chirag Anirudh S, Anirudhhan R

<b>SUMMARY</b>	2
Objective	2
Goals	2
Solution	2
Project Outline	2
1) Create Firebase Project	3
2) Set Authentication Methods	5
3) Get Project API Key	8
4) Set up Realtime Database	10
5) Set up Database Security Rules	12
6) ESP32 Datalogging (Firebase Realtime Database)	14
Installing libraries	15
CODE	15

---

# SUMMARY

## Objective

The main objective of this project is to obtain readings of various environmental factors which affect the health of poultry.

## Goals

The goal of this project is to produce a reliable hardware which collects the temperature, pressure, altitude and the presence of organic gases in different places.

## Solution

We have come up with a reliable hardware prototype which consists of a microcontroller, ESP32 and an environmental sensor, BME680.

The ESP32 is a feature-rich MCU with integrated Wi-Fi and Bluetooth connectivity for a wide-range of applications.

The BME680 is an environmental sensor which can detect the temperature, pressure, altitude and the presence of gas in the area it is placed.

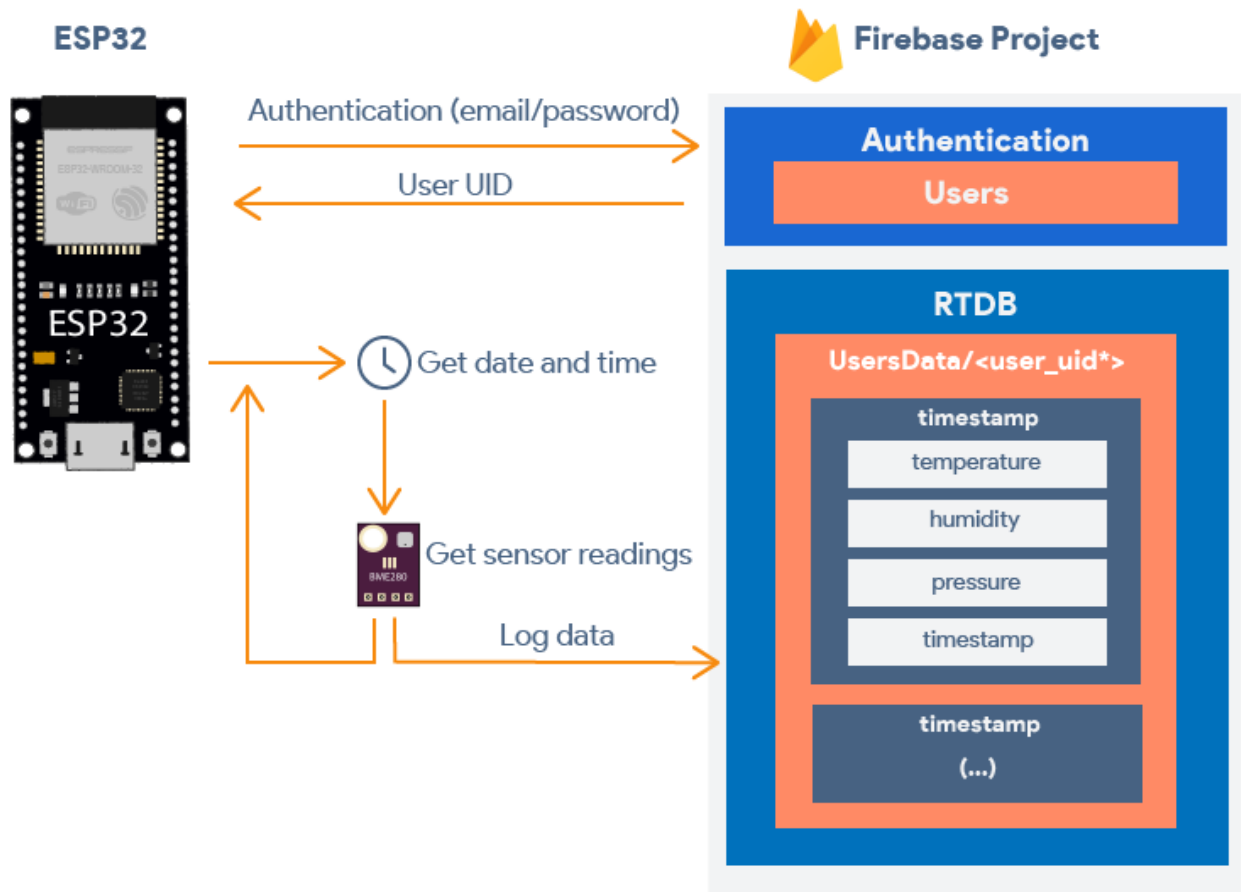
All the data received is going to be saved in a firebase database.

## Project Outline

- The ESP32/ESP8266 authenticates as a user with email and password (that user must be set on the Firebase authentication methods).
- After authentication, the ESP gets the user UID.
- The database is protected with security rules. The user can only access the database nodes under the node with its user UID. After getting the user UID, the ESP can publish data to the database.
- The ESP sends temperature, humidity and pressure to the database.

---

# SETTING UP FIREBASE SERVER



\*this is the unique UID of the authorized user

The following diagram depicts the data flow between the ESP32 and the server

## 1) Create Firebase Project

- 1) Go to [Firebase](#) and sign in using a Google Account.
- 2) Click *Get Started* and then **Add project** to create a new project.
- 3) Give a name to your project, for example, *ESP Firebase Demo*.


---

✕ Create a project (Step 1 of 3)

Let's start with a name for  
your project<sup>?</sup>

Project name

ESP Firebase Demo

 esp-firebase-demo

Continue

4) click **Create project**.

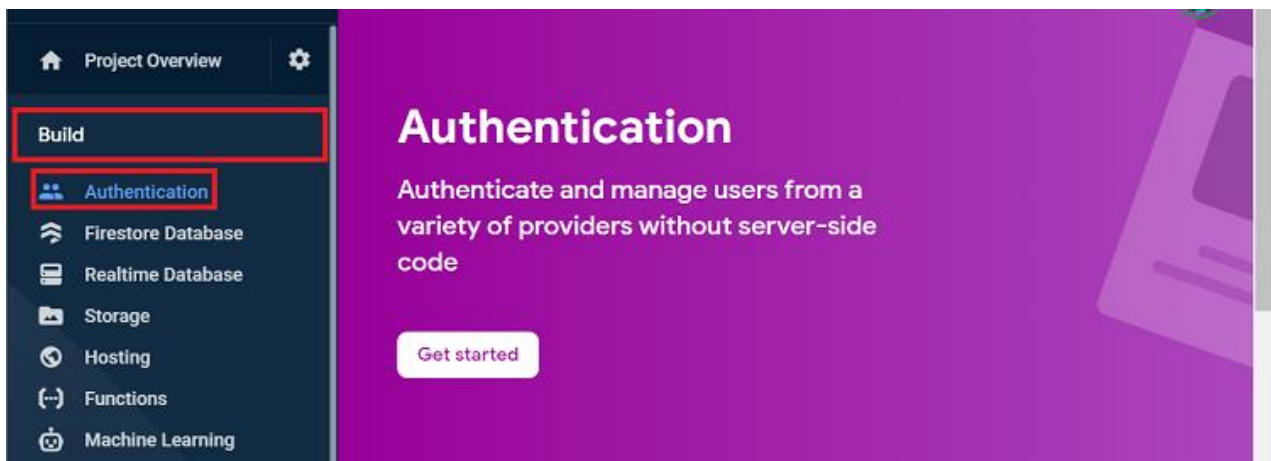
5) It will take a few seconds to set up your project. Then, click **Continue** when it's ready.

6) You'll be redirected to your Project console page.

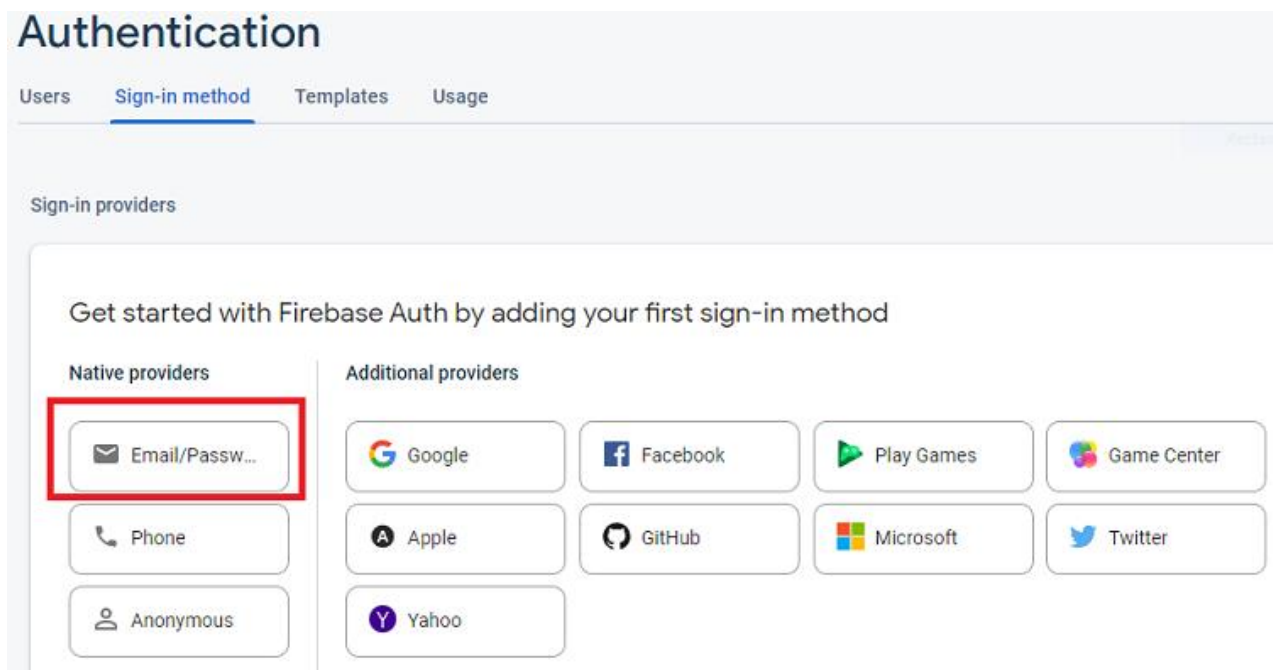
---

## 2) Set Authentication Methods

1) On the left sidebar, click on **Authentication** and then on **Get started**.




2) Select the Option **Email/Password**.



3) Enable that authentication method and click **Save**.

---

Sign-in providers

 Email/Password

Enable

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives.  
[Learn more](#)

Email link (passwordless sign-in)

Enable



Cancel

Save

4) The authentication with email and password should now be enabled.

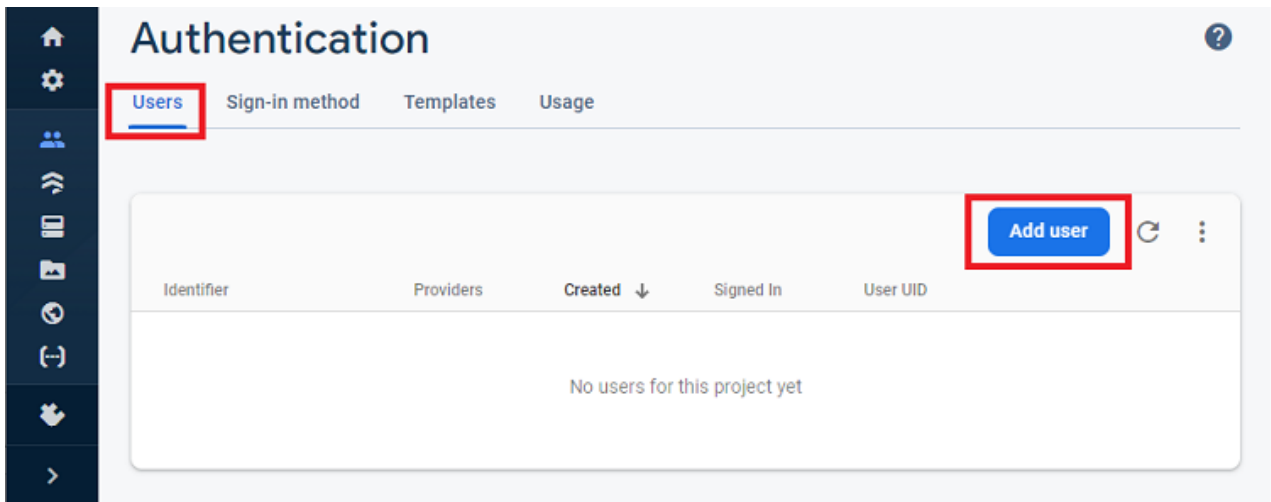
Sign-in providers

Add new provider

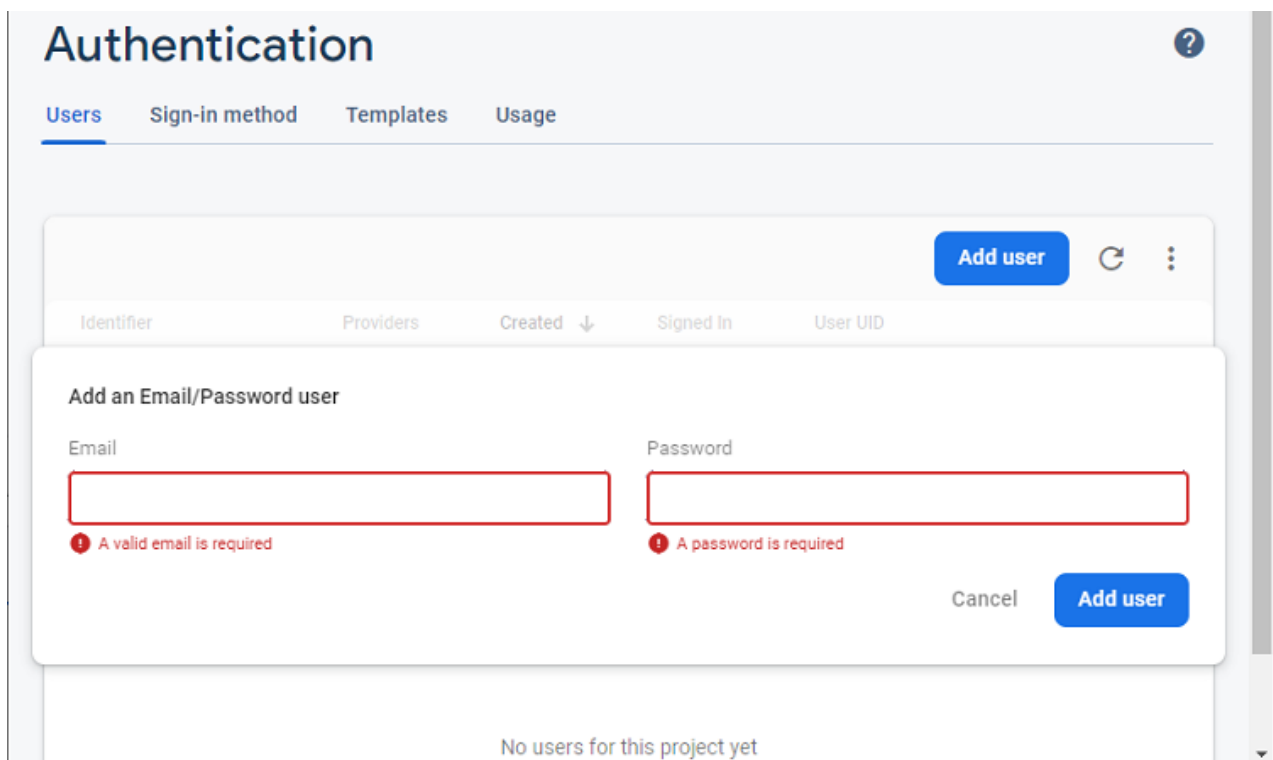
Provider	Status
 Email/Password	 Enabled

5) Now, you need to add a user. On the **Authentication** tab, select the **Users** tab at the top. Then, click on **Add User**.

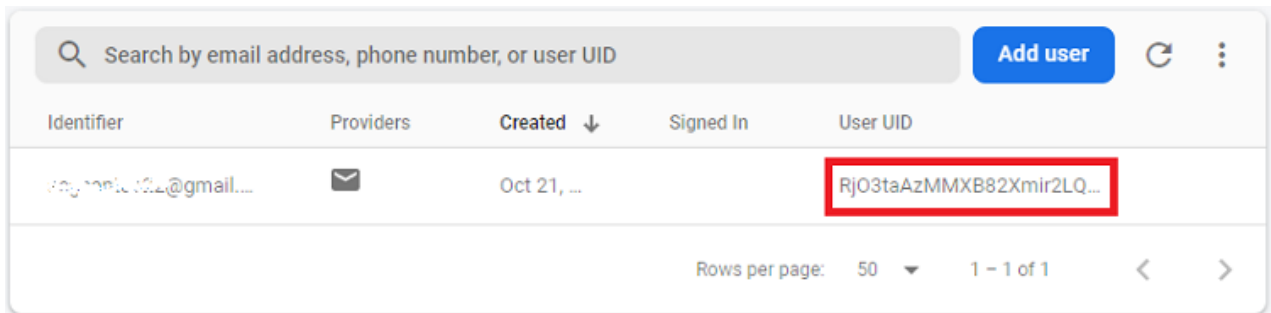
6) Add an email address for the authorized user. It can be your google account email or any other email. You can also create an email for this specific project. Add a password that will allow you to sign in to your



app and access the database. Don't forget to save the password in a safe place because you'll need it later. When you're done, click **Add user**.



7) A new user was successfully created and added to the **Users** table.



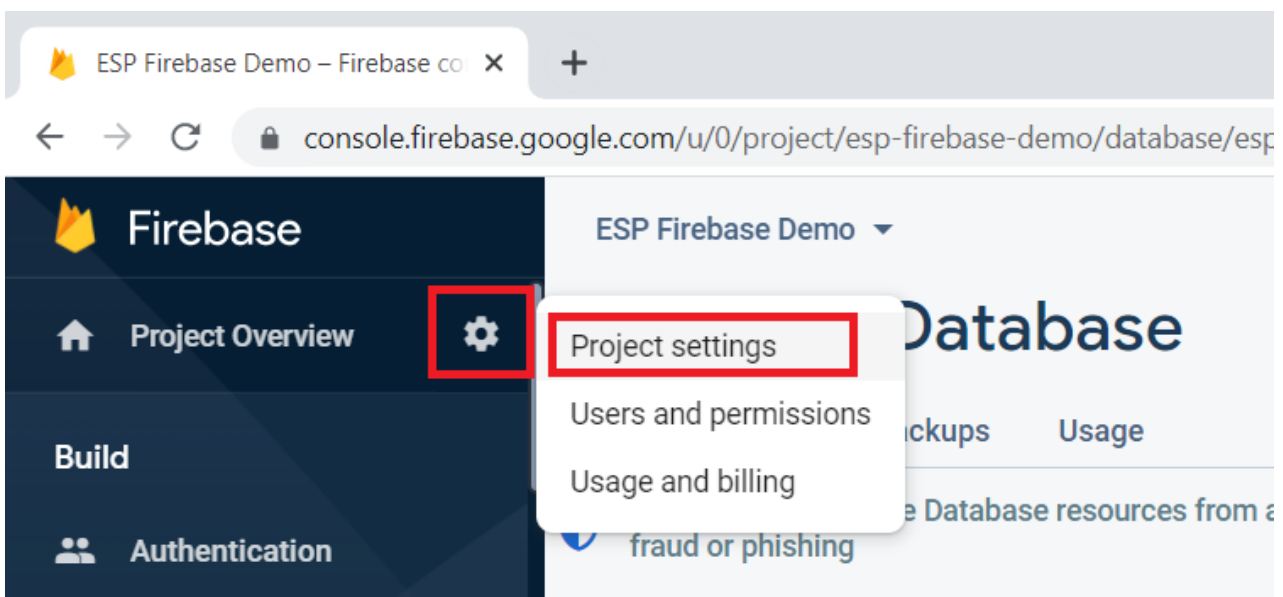
Identifier	Providers	Created ↓	Signed In	User UID
esp.firebase@gmail.com	📧	Oct 21, ...		RjO3taAzMMXB82Xmir2LQ...

Rows per page: 50 1 - 1 of 1

Notice that Firebase creates a unique UID for each registered user. The user UID allows us to identify the user and keep track of the user to provide or deny access to the project or the database. There's also a column that registers the date of the last sign-in. At the moment, it is empty because we haven't signed in with that user yet.

### 3) Get Project API Key

1) On the left sidebar, click on **Project Settings**.



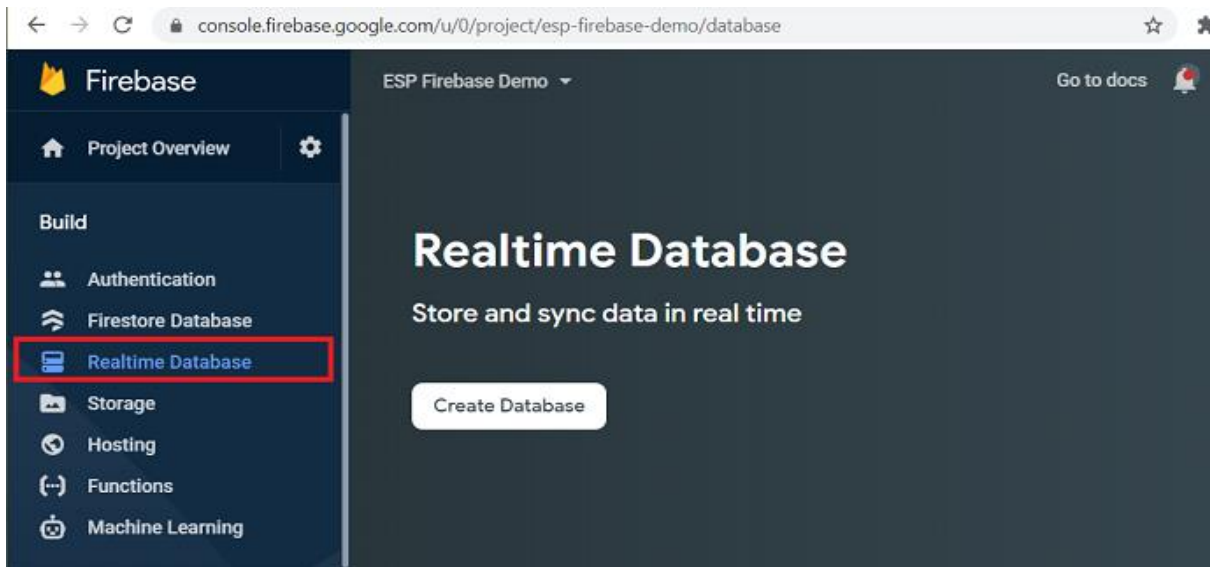




---

## 4) Set up Realtime Database

1) On the left sidebar, click on **Realtime Database** and then click on **Create Database**.



2) Select your database location. It should be the closest to your location.

3) Set up security rules for your database. You can select **Start in test mode**. We'll change the database rules later.

## Set up database



1 Database options

2 Security rules

Once you have defined your data structure **you will have to write rules to secure your data.**

[Learn more](#)

☐ Start in **locked mode**

Your data is private by default. Client read/write access will only be granted as specified by your security rules.

☒ Start in **test mode**

Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

```
{
  "rules": {
    ".read": "now < 1630537200000", // 2021-9-2
    ".write": "now < 1630537200000", // 2021-9-2
  }
}
```

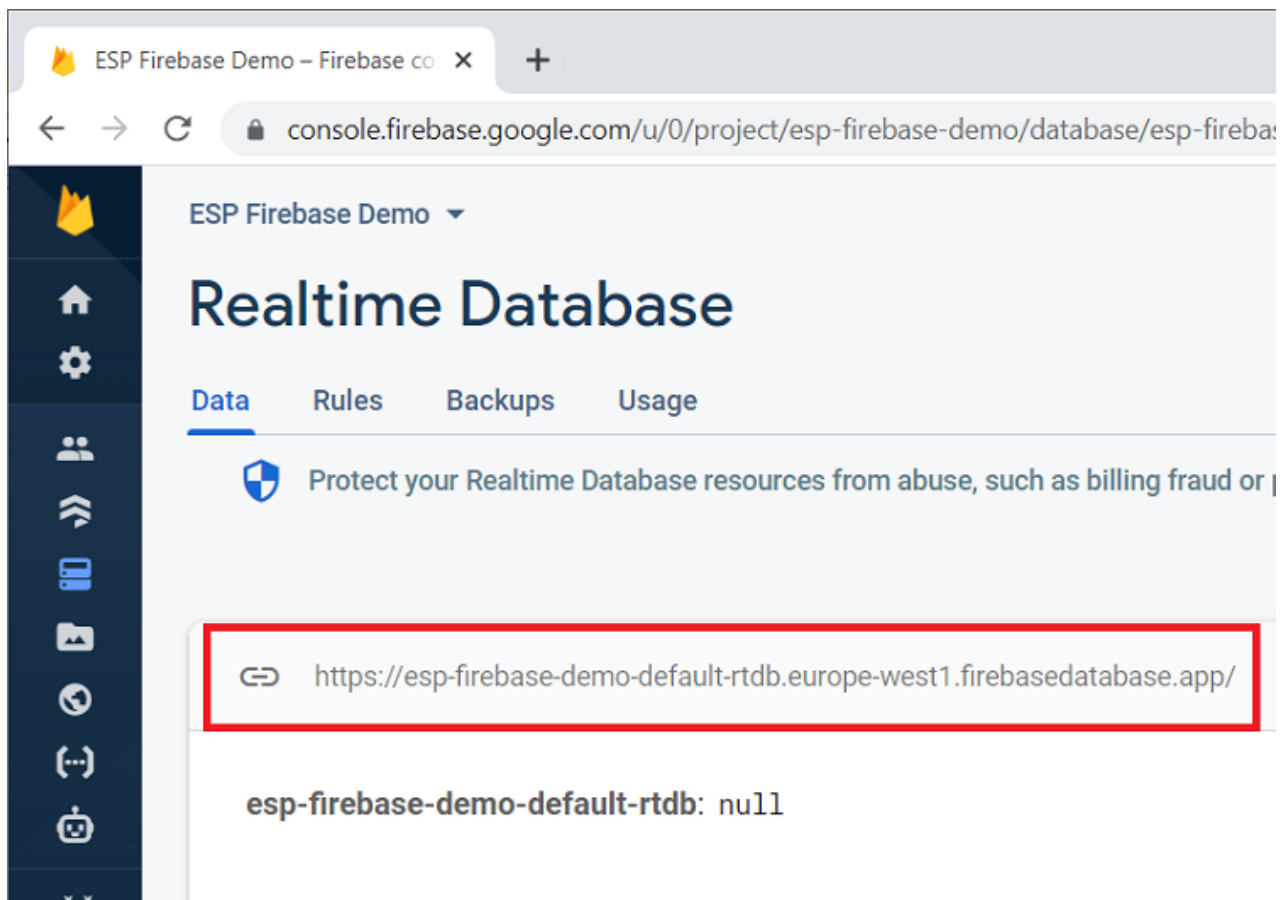


The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

Cancel

Enable

4) Your database is now created. You need to copy and save the database URL—highlighted in the following image—because you'll need it later in your ESP32 code.




## 5) Set up Database Security Rules

Now, let's set up the database rules. On the **Realtime Database** tab, select the **Rules** tab at the top. Then, click on **Edit rules**, copy the following rules and then click **Publish**.

```
// These rules grant access to a node matching the authenticated
// user's ID from the Firebase auth token
{
  "rules": {
    "UsersData": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

}



## Realtime Database

Data **Rules** Backups Usage

Edit rules Monitor rules

Rules Playground

```
1 // These rules grant access to a node matching the authenticated
2 // user's ID from the Firebase auth token
3 {
4   "rules": {
5     "UsersData": {
6       "$uid": {
7         ".read": "$uid === auth.uid",
8         ".write": "$uid === auth.uid"
9       }
10    }
11  }
12 }
```

These rules grant access to a node matching the authenticated user's UID. This grants that each authenticated user can only access its own data. This means the user can only access the nodes that are under a node with its corresponding user UID. If there are other data published on the database, not under a node with the users' UID, that user can't access that data.

For example, imagine our user UID is RjO3taAzMMXBB2Xmir2LQ. With our security rules, it can read and write data to the database under the node UsersData/RjO3taAzMMXBB2Xmir2LQ.

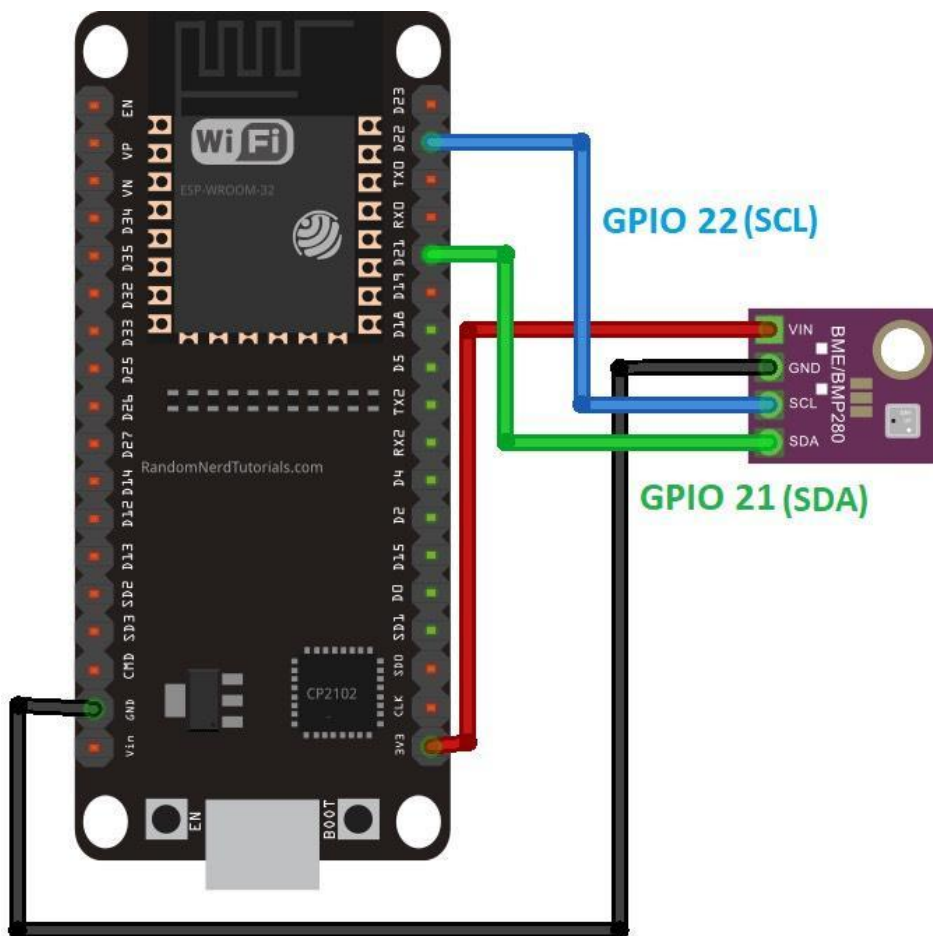
---

## 6) ESP32 Datalogging (Firebase Realtime Database)

### COMPONENTS REQUIRED

- An ESP32 microprocessor
- A BME680 environmental sensor
- Jumper cables

### SCHEMATIC DIAGRAM



---

# INSTALLING LIBRARIES

For running this project, we will need to install 3 libraries.

- Firebase ESP client Library
- Adafruit BME680 library
- Adafruit Unified sensor library

*For installing these libraries,*

1. Open the Arduino IDE
2. On the top corner, navigate to Sketch > Include Library > Manage Libraries. This will open a new window.
3. Search for the required libraries and install them.

*Installing the files required to compile the code in ESP32*

1. In the Arduino IDE, navigate to File > Preferences.
2. In the “Additional Board Manager URL’s”, paste this link “[https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)”
3. Restart the Arduino IDE
4. Navigate to Tools > Board > Board Manager
5. Search for ESP32 and install the library
6. Navigate to Tools > Board > Board Manager > ESP32 and select “DOIT ESP32 DEVKIT V1”

# CODE

```
#include <Arduino.h>
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include "Adafruit_BME680.h"
#include "time.h"
```

```
// Provide the token generation process info.
```

```
#include "addons/TokenHelper.h"
```

```
// Provide the RTDB payload printing info and other helper functions.
```

---

```
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "ENTER YOUR WIFI SSID"
#define WIFI_PASSWORD "ENTER YOUR WIFI PASSWORD"

// Insert Firebase project API Key
#define API_KEY "ENTER YOUR FIREBASE PROJECT API KEY"

// Insert Authorized Email and Corresponding Password
#define USER_EMAIL "ENTER AUTHORISED EMAIL ID"
#define USER_PASSWORD "ENTER THE PASSWORD"

// Insert RTDB URLdefine the RTDB URL
#define DATABASE_URL "ENTER THE REAL TIME DATABASE URL"

#define SEALEVELPRESSURE_HPA (1013.25)
#define DEVICE_ID "001" //CHANGE THE DEVICE ID EVERY TIME YOU DUMP THE CODE TO A NEW ESP32 BOARD

// Define Firebase objects
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// Variable to save USER UID
String uid;

// Database main path (to be updated in setup with the user UID)
String databasePath;
// Database child nodes
String tempPath = "/temperature";
String humPath = "/humidity";
String presPath = "/pressure";
String gasPath = "/gas";
String altPath = "/altitude";
String timePath = "/timestamp";
String devicePath = "/Device_ID";

// Parent Node (to be updated in every loop)
String parentPath;

int timestamp;
FirebaseJson json;
```

---



---

```
const char* ntpServer = "pool.ntp.org";

// BME680 sensor
Adafruit_BME680 bme; // I2C
float temperature;
float humidity;
float pressure;
float gas;
float altitude;

// Timer variables (send new readings every three minutes)
unsigned long sendDataPrevMillis = 0;
unsigned long timerDelay = 5000;

// Initialize BME680
void initBME(){
  if (!bme.begin()) {
    Serial.println("Could not find a valid BME680 sensor, check wiring!");
    while (1);
  }
}

// Initialize WiFi
void initWiFi() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
  Serial.println();
}

// Function that gets current epoch time
unsigned long getTime() {
  time_t now;
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    //Serial.println("Failed to obtain time");
    return(0);
  }
  time(&now);
  return now;
}

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

---

```
// Initialize BME280 sensor
initBME();
initWiFi();
configTime(0, 0, ntpServer);

// Assign the api key (required)
config.api_key = API_KEY;

// Assign the user sign in credentials
auth.user.email = USER_EMAIL;
auth.user.password = USER_PASSWORD;

// Assign the RTDB URL (required)
config.database_url = DATABASE_URL;

Firebase.reconnectWiFi(true);
fbdo.setResponseSize(4096);

// Assign the callback function for the long running token generation task */
config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

// Assign the maximum retry of token generation
config.max_token_generation_retry = 5;

// Initialize the library with the Firebase authen and config
Firebase.begin(&config, &auth);

// Getting the user UID might take a few seconds
Serial.println("Getting User UID");
while ((auth.token.uid) == "") {
  Serial.print('.');
  delay(1000);
}
// Print user UID
uid = auth.token.uid.c_str();
Serial.print("User UID: ");
Serial.println(uid);

// Update database path
databasePath = "/UsersData/" + uid + "/readings";
}

void loop(){

// Send new readings to database
if (Firebase.ready() && (millis() - sendDataPrevMillis > timerDelay || sendDataPrevMillis == 0)){
```

---

```
sendDataPrevMillis = millis();

//Get current timestamp
timestamp = getTime();
Serial.print ("time: ");
Serial.println (timestamp);

parentPath= databasePath + "/" + String(timestamp);

json.set(tempPath.c_str(), String(bme.readTemperature()));
json.set(humPath.c_str(), String(bme.readHumidity()));
json.set(presPath.c_str(), String(bme.readPressure()/100.0F));
json.set(gasPath.c_str(), String(bme.gas_resistance/1000.0));
json.set(altPath.c_str(), String(bme.readAltitude(SEALEVELPRESSURE_HPA)));
json.set(timePath, String(timestamp));
json.set(devicePath, DEVICE_ID);
Serial.printf("Set json... %s\n", Firebase.RTDB.setJSON(&fbdo, parentPath.c_str(), &json) ? "ok" :
fbdo.errorReason().c_str());
}
}
```

Now, compile and run the code. Once the code is dumped into the ESP32, press the magnifying glass on the top right corner of the Arduino IDE. This opens the serial monitor. In this screen, set the baud rate to 115200 and check if it works as expected.

Once the status changes to ready, open the firebase server. All the data will be stored here.