

Practical -9

Objective: Create program to connect Node MCU with the Google Firebase cloud and upload and download the sensor data to and from the Google Firebase cloud.

Introduction: Firebase is Google's mobile application development platform that includes many services to manage data from IOS, Android, or web applications. You'll create a Firebase project with a real-time database (RTDB), and you'll learn how to store and read values from the database with your ESP8266 board.



What is Firebase?

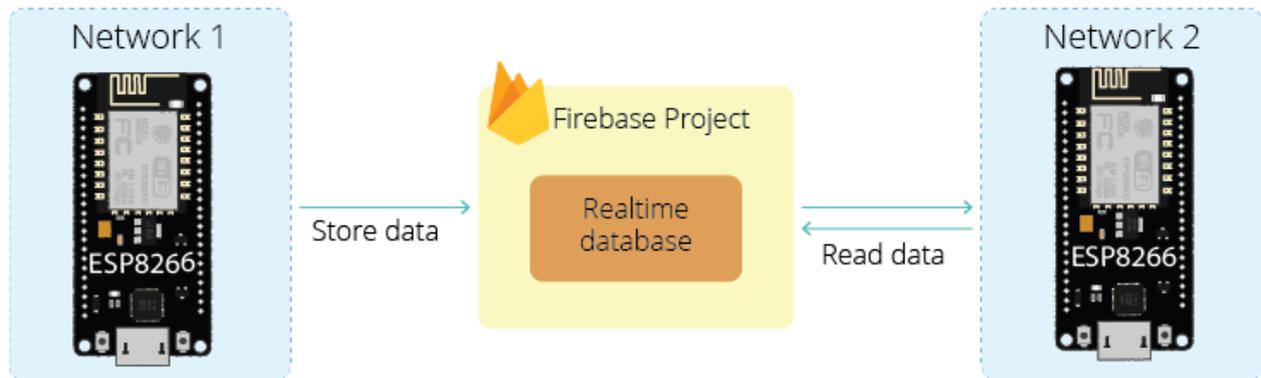
Firebase is Google's mobile application development platform that helps you build, improve, and grow your app. It has many services used to manage data from any android, IOS, or web application.

“Firebase is a toolset to “build, improve, and grow your app”, and the tools it gives you cover a large portion of the services that developers would normally have to build themselves but don't really want to build because they'd rather be focusing on the app experience itself. This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on. The services are hosted in the cloud and scale with little to no effort on the part of the developer.”

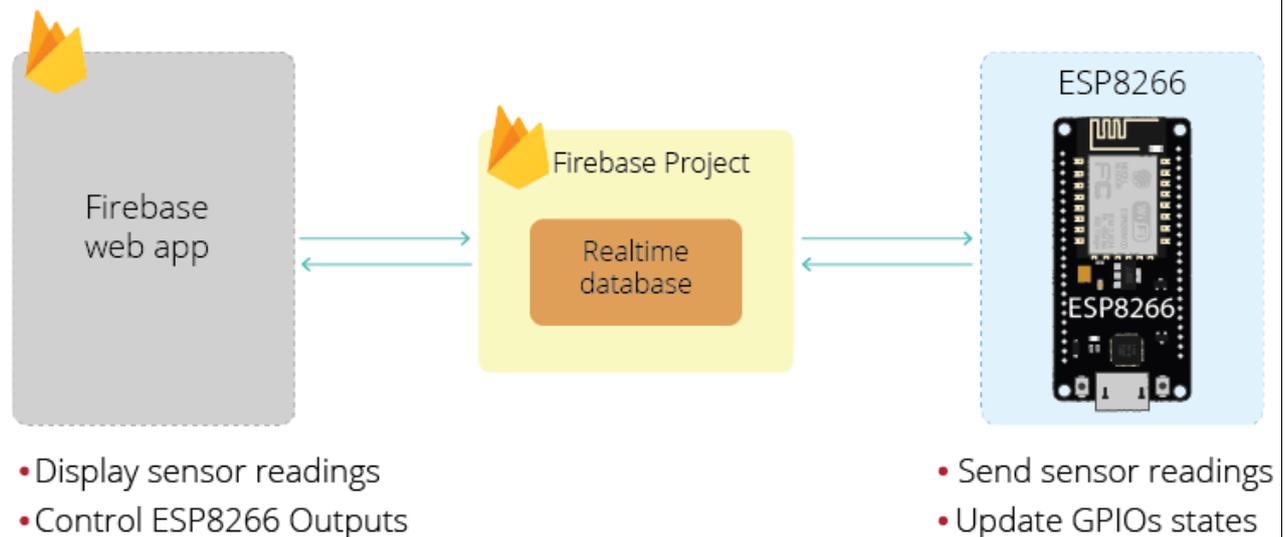
We can use the ESP8266 to connect and interact with our Firebase project, and we can create applications to control the ESP8266 via Firebase from anywhere in the world.

Now here, we'll create a Firebase project with a real-time database, and we'll use the ESP8266 to store and read data from the database. The ESP8266 can interact with the database from anywhere in the world as long as it is connected to the internet.

This means that we can have two ESP8266 boards in different networks, with one board storing data and the other board reading the most recent data, for example.



After, we'll create a web app using Firebase that will control the ESP8266 to display sensor readings or control outputs from anywhere in the world.



This tutorial is divided into three sections:

1. Create a Firebase Project
2. ESP8266: Store data to the Firebase Realtime Database
3. ESP8266: Read data from the Firebase Realtime Database

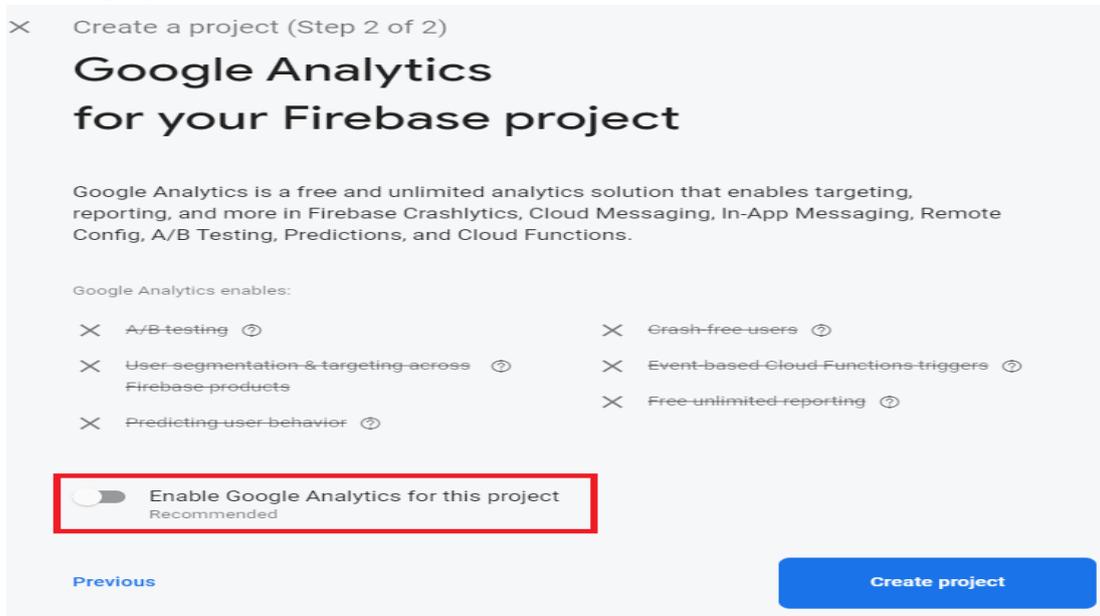
1. Create a Firebase Project:

Follow the next instructions to create a new project on Firebase.

- a) Go to Firebase and sign in using a Google Account.
- b) Click *Get Started*, and then *Add project* to create a new project.
- c) Give a name to your project, for example: *ESP Firebase Demo*.

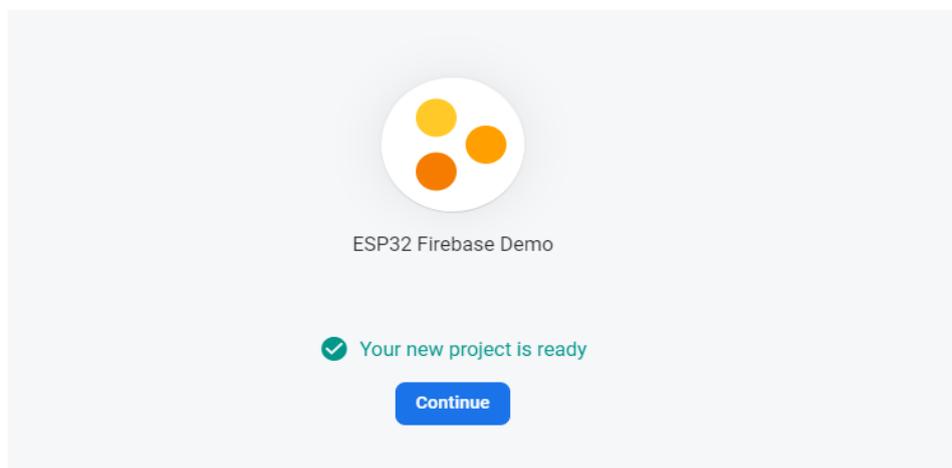


d) Disable the option *Enable Google Analytics* for this project as it is not needed and click *Create project*.

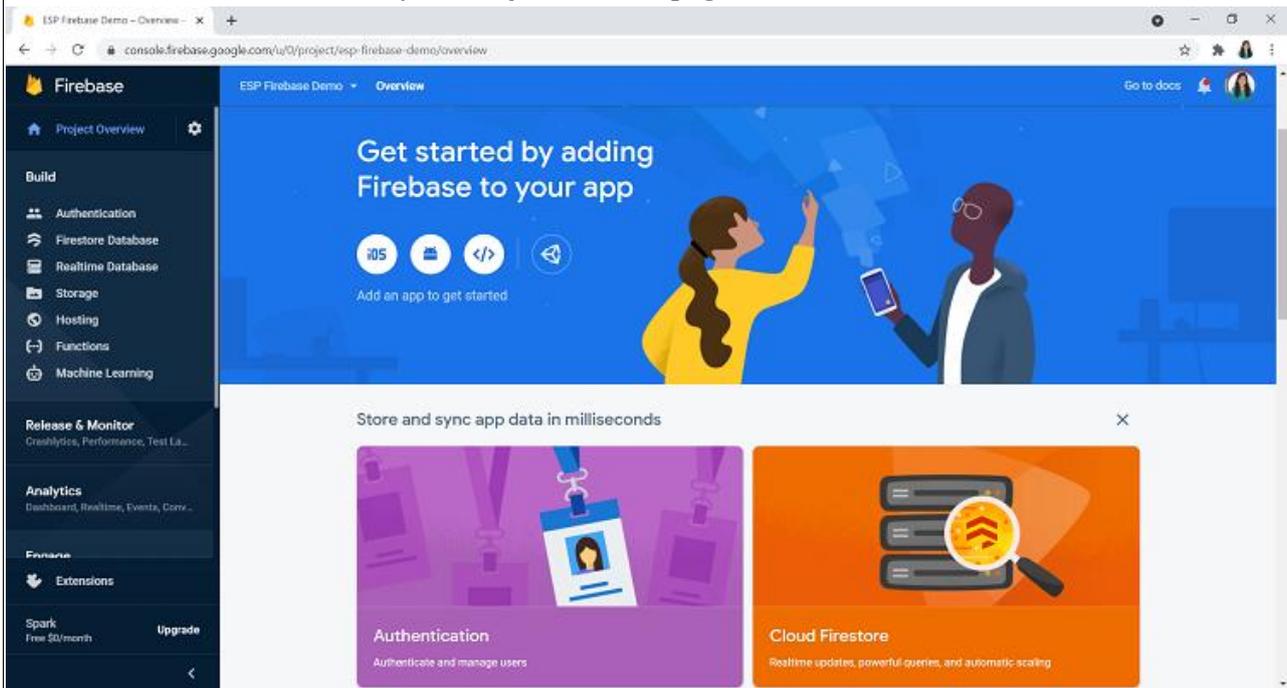


e) It will take a few seconds setting up your project. Then, click *Continue*

when it's ready.



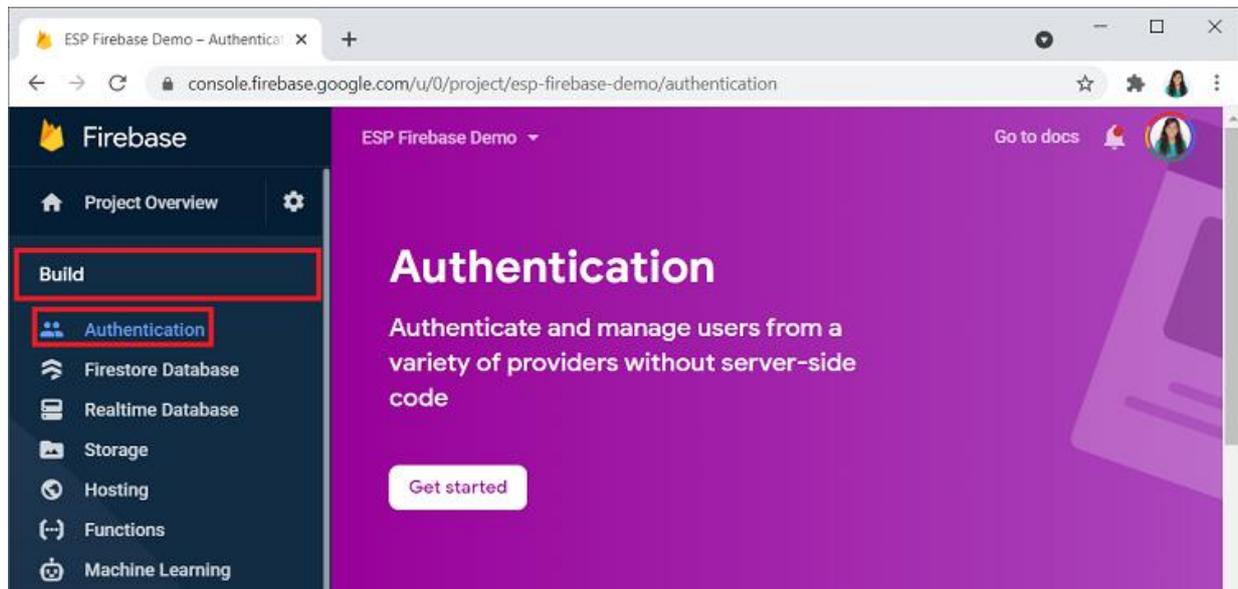
f) You'll be redirected to your Project console page.



2. Set Authentication Methods:

“Most apps need to know the identity of a user. In other words, it takes care of logging in and identify the users (in this case, the ESP8266). Knowing a user’s identity allows an app to securely save user data in the cloud and provide the same personalized experience across all of the user’s devices.”

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



II. There are several authentication methods like email and password, Google Account, Facebook account, and others.

Sign-in providers

Provider	Status	
 Email/Password	Disabled	
 Phone	Disabled	
 Google	Disabled	
 Play Games	Disabled	
 Game Center	Disabled	
 Facebook	Disabled	
 Twitter	Disabled	
 GitHub	Disabled	
 Yahoo	Disabled	
 Microsoft	Disabled	
 Apple	Disabled	
 Anonymous	Disabled	

III. For testing purposes, we can select the *Anonymous* user (require authentication without requiring users to sign in first by creating temporary anonymous accounts). Enable that option and click *Save*.

 Anonymous

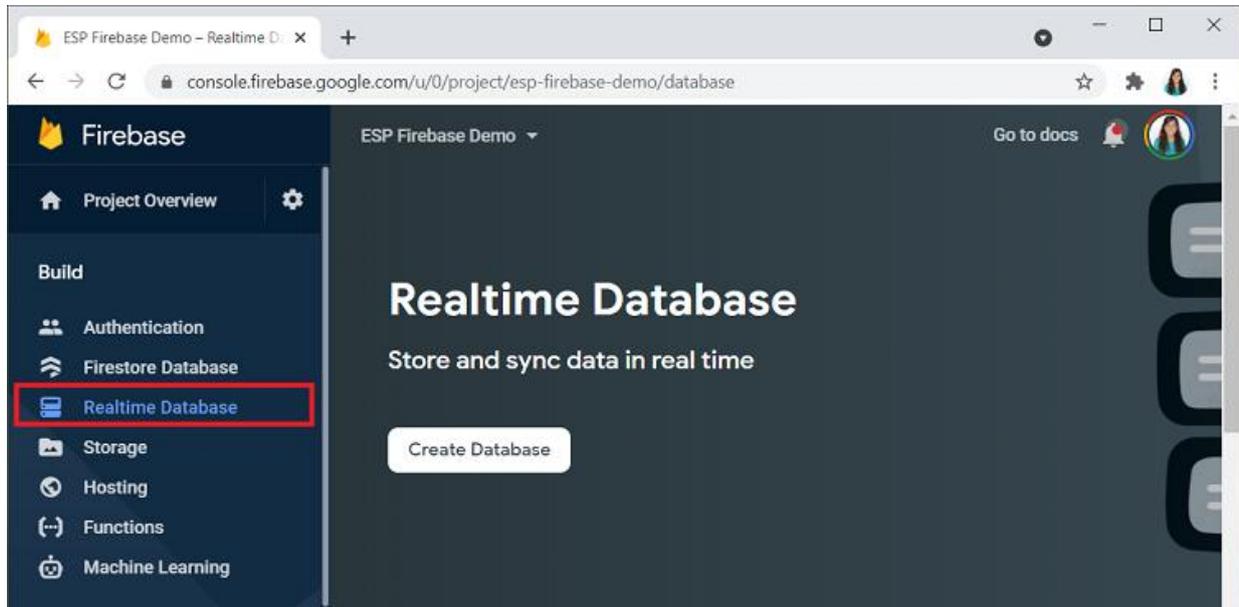
Enable

Enable anonymous guest accounts in your application, which lets you enforce user-specific Security and Firebase rules without requiring credentials from your users. [Learn more](#) 

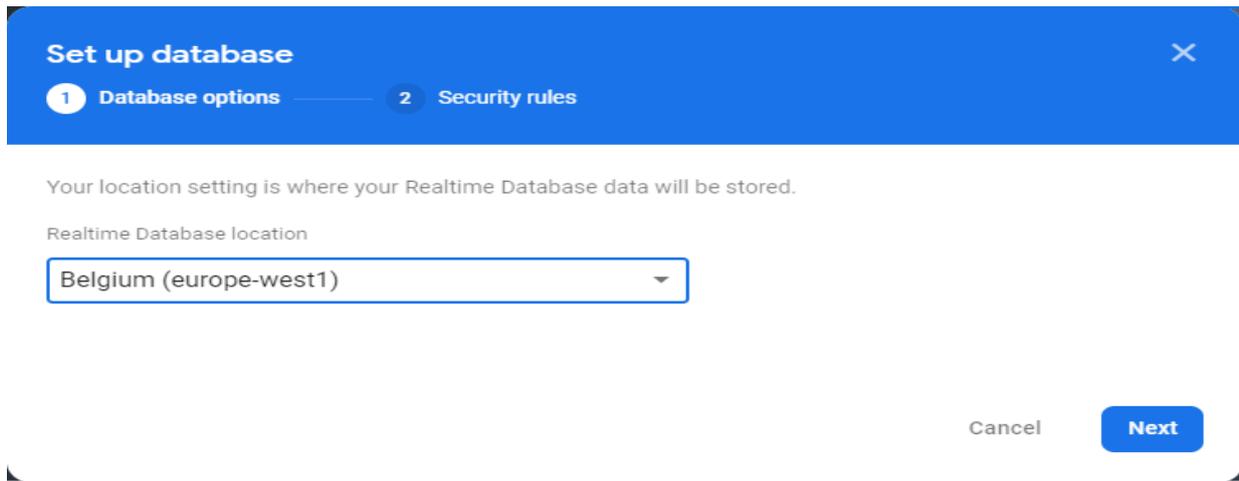
Cancel

3. Creating a Realtime Database: The next step is creating a Realtime Database for your project. Follow the next steps to create the database.

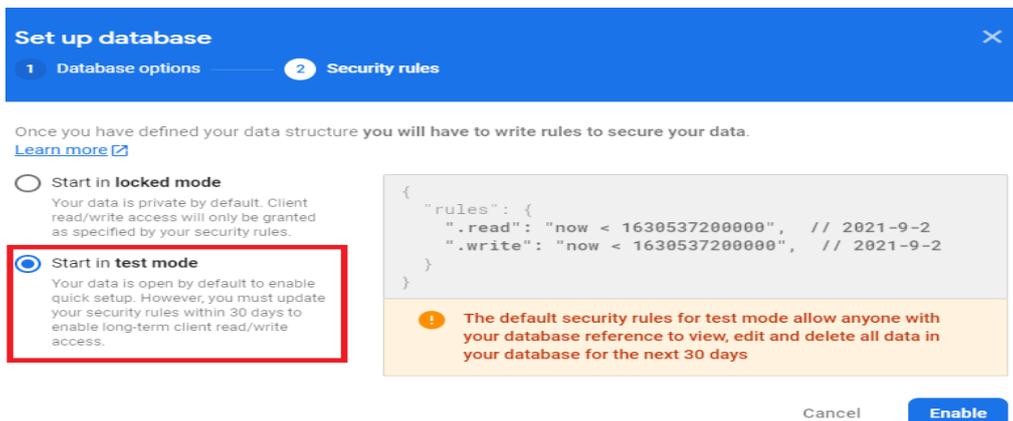
I. On the left sidebar click on *Realtime Database* and then, click on *Create Database*.



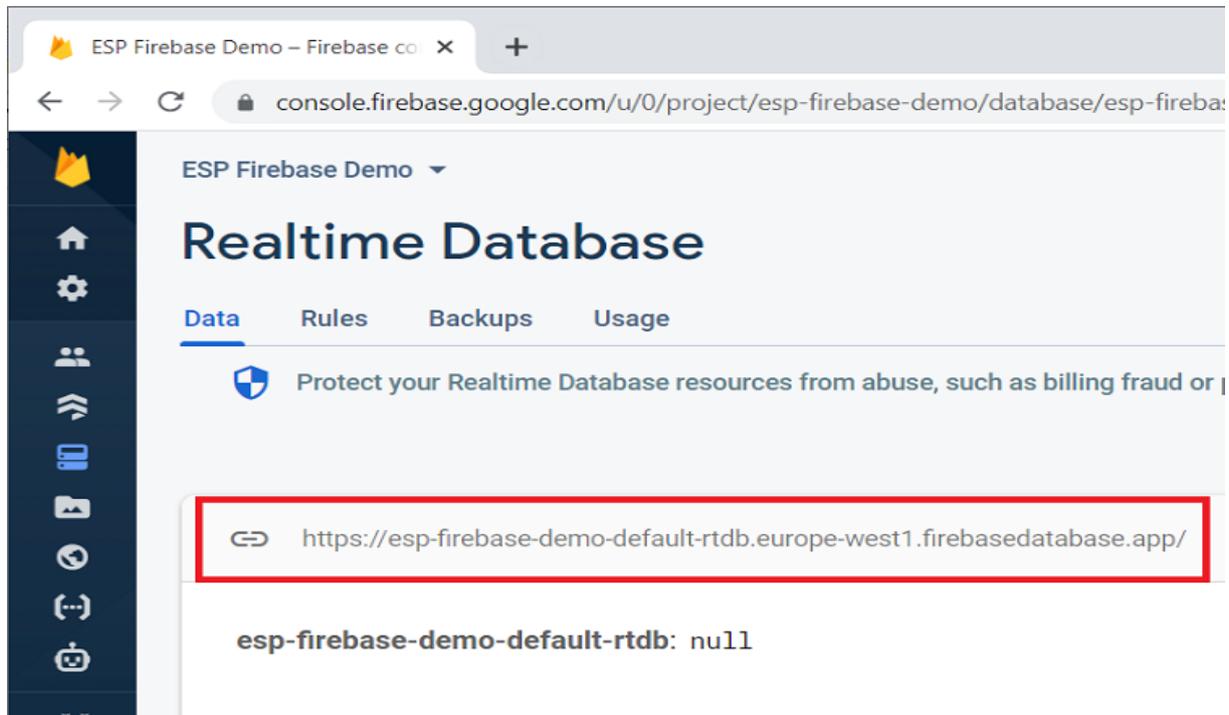
II. Select your database location. It should be the closest to your location.



III. Set up security rules for your database. For testing purposes, select *Start in test mode*. In later tutorials you'll learn how to secure your database using database rules.



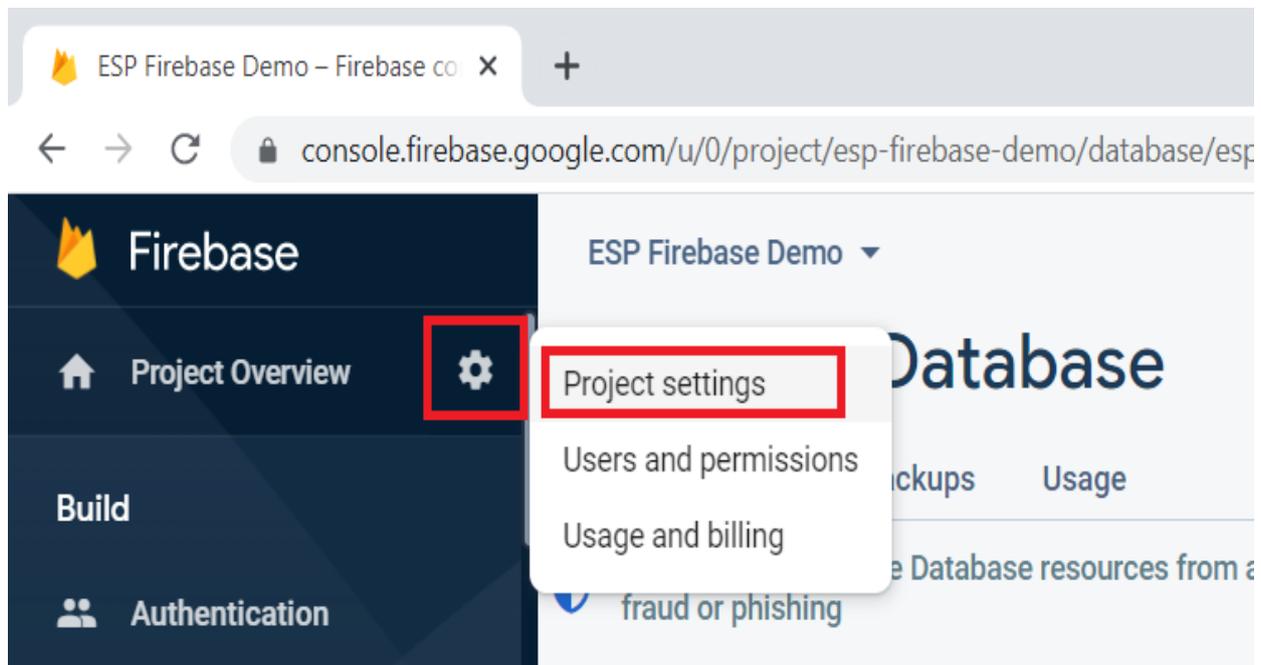
IV. Your database is now created. You need to copy and save the database URL—highlighted in the following image—because you’ll need it later in your ESP8266 code.



The Realtime Database is all set. Now, you also need to get your project API key.

4. Get Project API Key:

I. To get your project’s API key, on the left sidebar click on *Project Settings*.



II. Copy the API Key to a safe place because you’ll need it later.

The screenshot shows the Firebase console interface. On the left is a dark sidebar with navigation options: Project Overview, Build (Authentication, Firestore Database, Realtime Database, Storage, Hosting, Extensions), Spark (Free \$0/month), and Upgrade. The main content area is titled 'Project settings' for 'ESP Firebase Demo'. Below the title are tabs for General, Cloud Messaging, Integrations, Service accounts, and Data privacy. The 'General' tab is active, displaying 'Your project' information: Project name (ESP Firebase Demo), Project ID (esp-firebase-demo), Project number (1086300631316), and Default GCP resource location (Not yet selected). The 'Web API Key' field is highlighted with a red rectangular box.

Program the ESP8266 to Interface with Firebase:

Installation – Arduino IDE

If you're using Arduino IDE, follow the next steps to install the library.

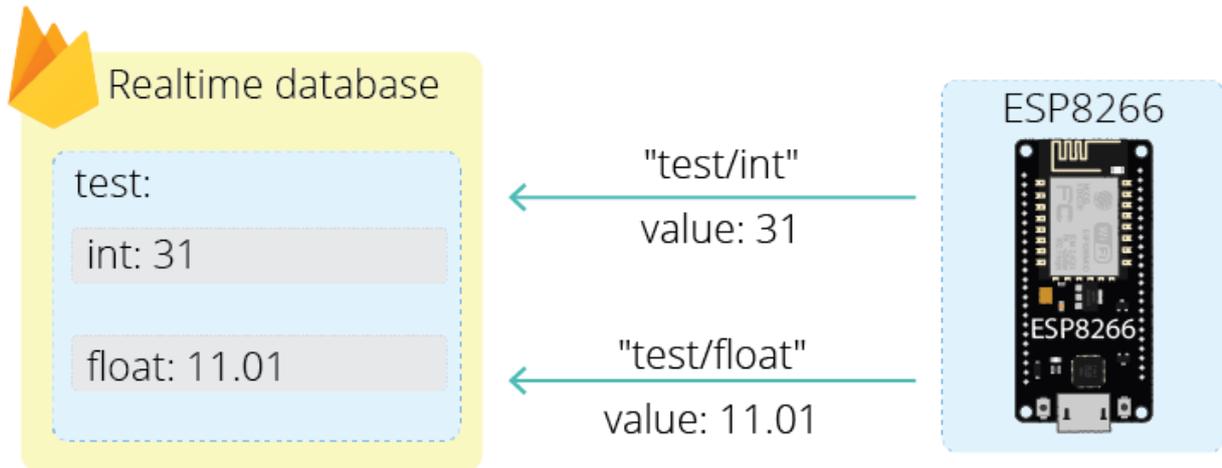
1. Go to **Sketch > Include Library > Manage Libraries**
2. Search for *Firebase ESP Client* and install the *Firebase Arduino Client Library for ESP8266 and ESP32* by Mobitz.

The screenshot shows the 'Library Manager' window in Arduino IDE. The search filter is set to 'Firebase ESP Client'. Three search results are visible:

- Highlighted (red box):** **Firebase Arduino Client Library for ESP8266 and ESP32** by Mobitz. Description: Google Firebase Arduino Client Library for Espressif ESP8266 and ESP32. This client library provides the functions to work with Firebase Realtime database, Firestore, Storage and Cloud messaging. Version 2.3.7, Install button.
- Second result:** **Firebase ESP32 Client** by Mobitz. Description: Google Firebase Realtime Database Arduino Client Library for Espressif ESP32. This client library provides the most reliable operations for read, store, update, delete, backup and restore the Firebase Realtime database data.
- Third result:** **Firebase ESP8266 Client** by Mobitz. Description: Google Firebase Realtime Database Arduino Client Library for Espressif ESP8266. This client library provides the most reliable operations for read, store, update, delete, backup and restore the Firebase Realtime database data.

 A 'Close' button is located at the bottom right of the window.

ESP8266 Store Data to Firebase Database:



```
#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.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 "REPLACE_WITH_YOUR_SSID"
#define WIFI_PASSWORD "REPLACE_WITH_YOUR_PASSWORD"

// Insert Firebase project API Key
#define API_KEY "REPLACE_WITH_YOUR_FIREBASE_PROJECT_API_KEY"

// Insert RTDB URLdefine the RTDB URL */
#define DATABASE_URL "REPLACE_WITH_YOUR_FIREBASE_DATABASE_URL"

//Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;
int count = 0;
bool signupOK = false;

void setup(){
```

```

Serial.begin(115200);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to Wi-Fi");
while (WiFi.status() != WL_CONNECTED){
  Serial.print(".");
  delay(300);
}
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();
/* Assign the api key (required) */
config.api_key = API_KEY;
/* Assign the RTDB URL (required) */
config.database_url = DATABASE_URL;
/* Sign up */
if (Firebase.signUp(&config, &auth, "", "")){
  Serial.println("ok");
  signupOK = true;
}
else{
  Serial.printf("%s\n", config.signer.signupError.message.c_str());
}
/* Assign the callback function for the long running token generation task */
config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);
}

void loop(){
if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 15000 || sendDataPrevMillis == 0)){
  sendDataPrevMillis = millis();
  // Write an Int number on the database path test/int
  if (Firebase.RTDB.setInt(&fbdo, "test/int", count)){
    Serial.println("PASSED");
    Serial.println("PATH: " + fbdo.dataPath());
    Serial.println("TYPE: " + fbdo.dataType());
  }
  else {
    Serial.println("FAILED");
    Serial.println("REASON: " + fbdo.errorReason());
  }
  count++;
  // Write an Float number on the database path test/float
  if (Firebase.RTDB.setFloat(&fbdo, "test/float", 0.01 + random(0,100))){
    Serial.println("PASSED");
  }
}
}}

```

COM16



Send

```
PASSED  
PATH: test/int  
TYPE: int  
PASSED  
PATH: test/float  
TYPE: float
```



Autoscroll Show timestamp

Newline ▾

115200 baud ▾

Clear output