Part A: NodeRED and MongoDB

MongoDB Installation

- `sudo apt update`
- `sudo apt upgrade`
- `wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -`
- `sudo apt-get install gnupg`
- `echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list`
- `sudo apt-get update`
- `sudo apt-get install -y mongodb-org` **//Remark if you cannot install mongodb-org, user this one instead** "`sudo apt-get install -y mongodb`"
- `sudo systemctl start mongod`

Exercise 1: What is the version of your mongoDB?
Example result:

```
yu@yu:/etc/apt/sources.list.d$ mongo -version
MongoDB shell version v3.6.8
git version: 8e540c0b6db93ce994cc548f000900bdc740f80a
OpenSSL version: OpenSSL 1.1.1f  31 Mar 2020
allocator: tcmalloc
modules: none
build environment:
    distarch: x86_64
    target_arch: x86_64
```

Your result:

Exercise 2: Can you check the ouput of the following command?
mongo --eval 'db.runCommand({ connectionStatus: 1 })'
Example result:

```
yu@yu:/etc/apt/sources.list.d$ mongo --eval 'db.runCommand({ connectionStatus: 1 })'
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("6a92701d-cecc-45ee-93a5-354e8f6254c3") }
MongoDB server version: 3.6.8
{
        "authInfo" : {
                "authenticatedUsers" : [ ],
                "authenticatedUserRoles" : [ ]
        },
        "ok" : 1
}
yu@yu:/etc/apt/sources.list.d$
```

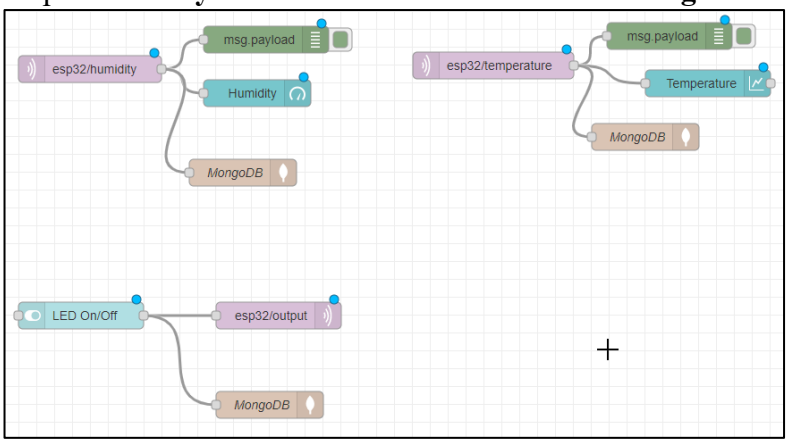
Your result:

MongoDB Node

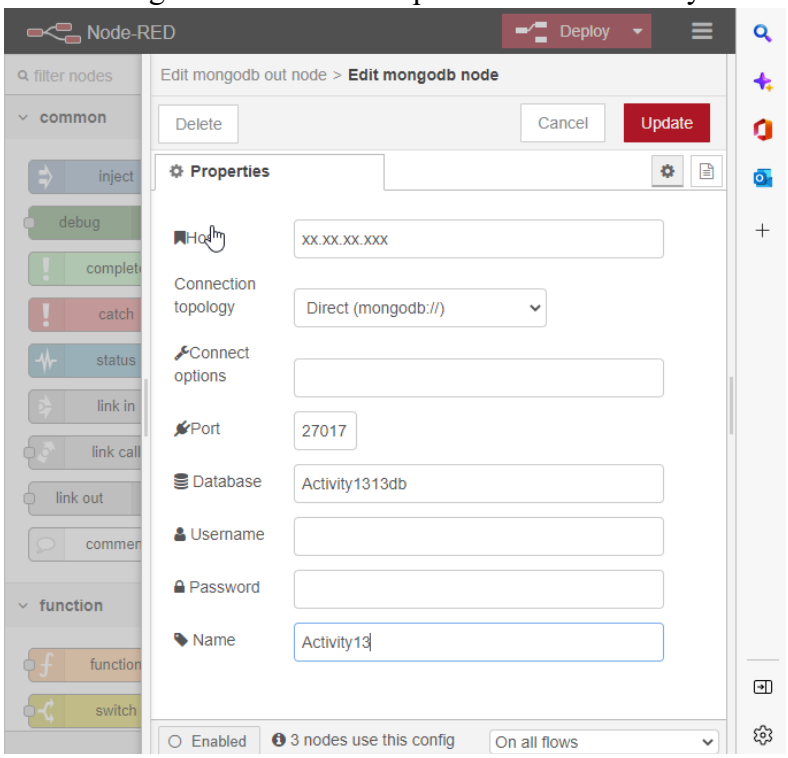
Continue the flow from Activity11 with ESP32 and AHT20.

Do the following instructions:

1. Install `node-red-contrib-mongodb` and `node-red-node-mongodb` at Manage Palette
2. import **Activity13-ESP32-AHT20-NodeRED-MongoDB.txt**



3. Add "Mongodb out" from Temperature and Humidity with the properties of



4. Find out the following results on your mongoDB.

   Go to Ubuntu terminal

   $ mongo

   > use Activity13db

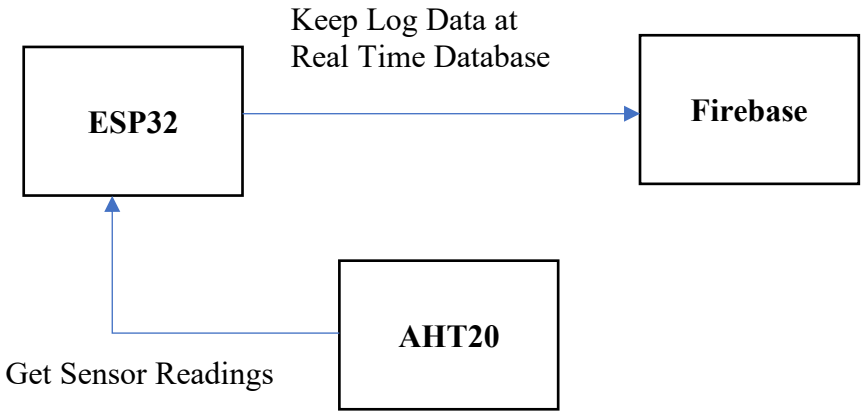
   > db.Temperature.find()

   > db.Humidity.find()

Your result:

Part B: ESP32 & Firebase

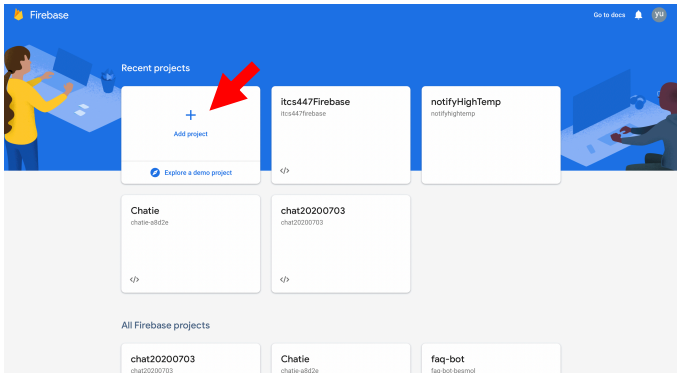
Process Flow



1. The ESP32 authenticates as a user with email and password (that user must be set on the Firebase authentication methods).
2. After authentication, the ESP gets the user UID.
3. 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.
4. The ESP32 gets temperature and humidity from AHT20 sensor.
5. The ESP32 sends temperature and humidity to the Firebase Realtime database.

Create Firebase Project

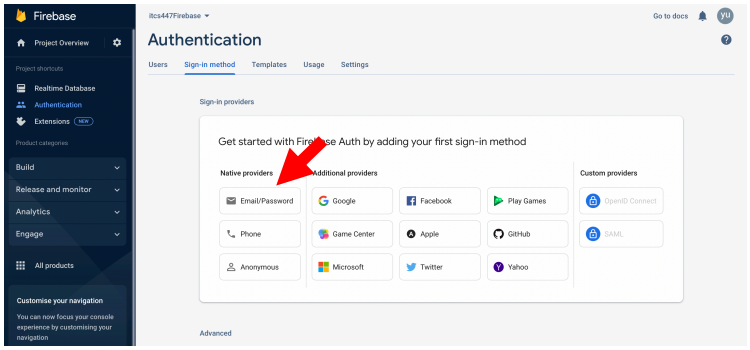
1. Go to “https://console.firebase.google.com” and sign up using Google account.
2. Click “Add project” to create a new project.

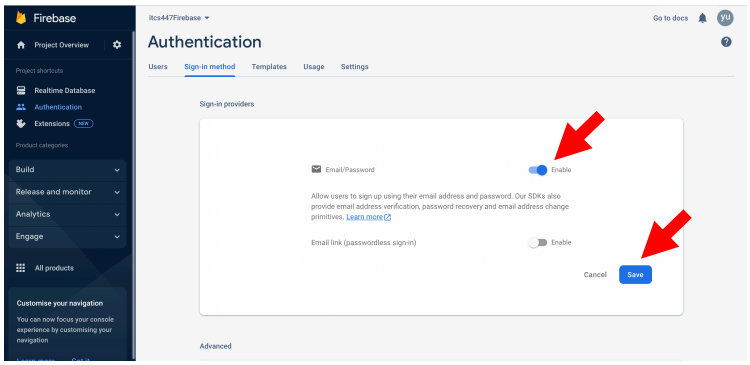


3. It will take a few seconds to create a new project. Then click “continue” when it is ready.

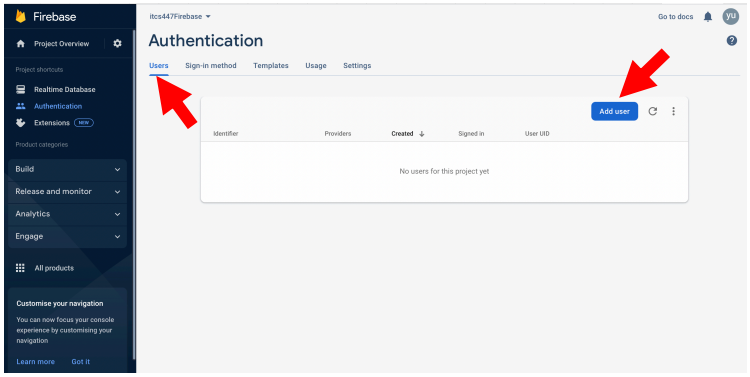
Set Authentication Method

4. On the left sidebar, click on **Authentication** and then on **Get started.** Then, select "Email/Password" option. And then "Enable" and "Save" as shown below.

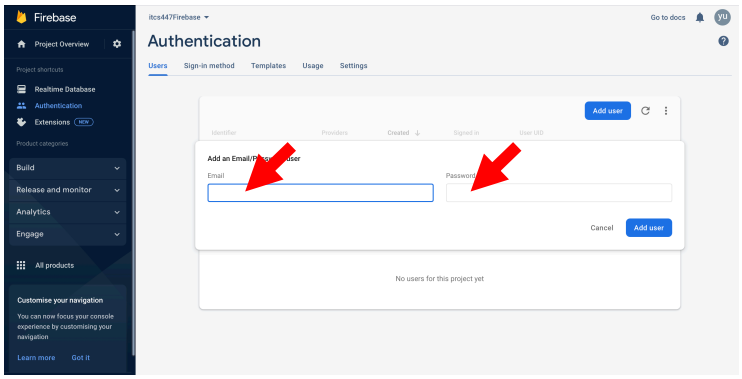




5. Then go to "Users" tab to add a new user.

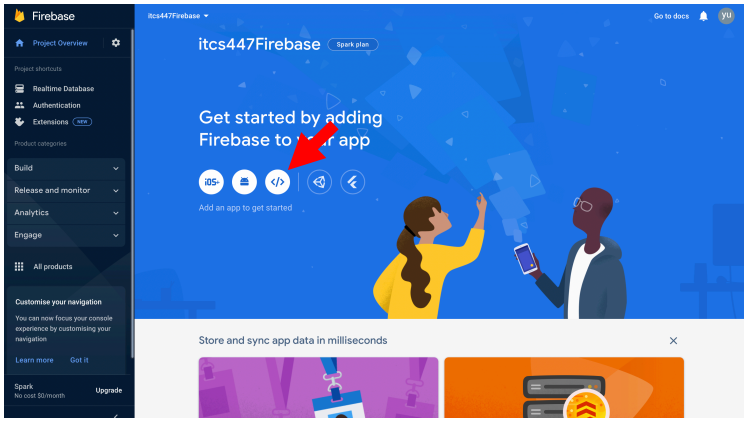


6. Add a new user with "Gmail account" and your preference password. Remark: This email address and password will be used in ESP32 Arduino code later.

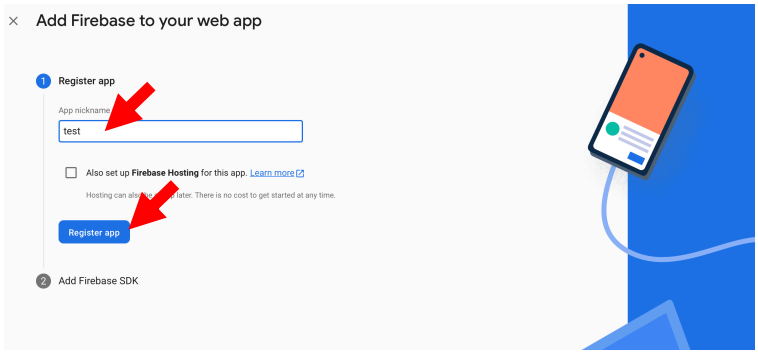


Get Project API

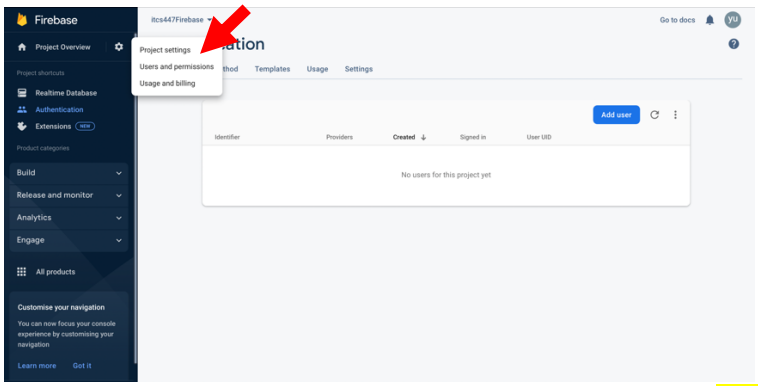
7. To get your project API, go to Firebase home page and select your project. And click on “Web” logo.



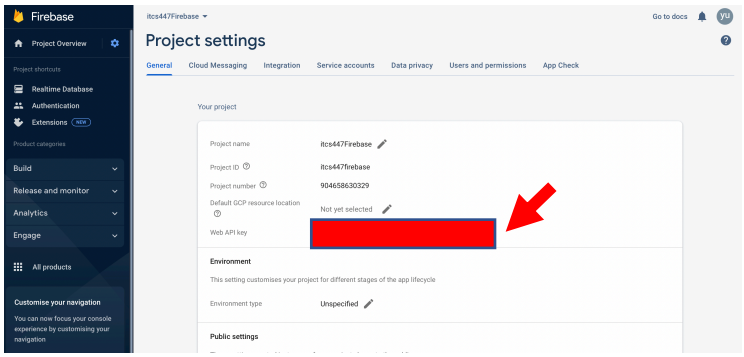
8. Give your Web API name and register your app.



9. Now, you can go to your project setting as shown below.

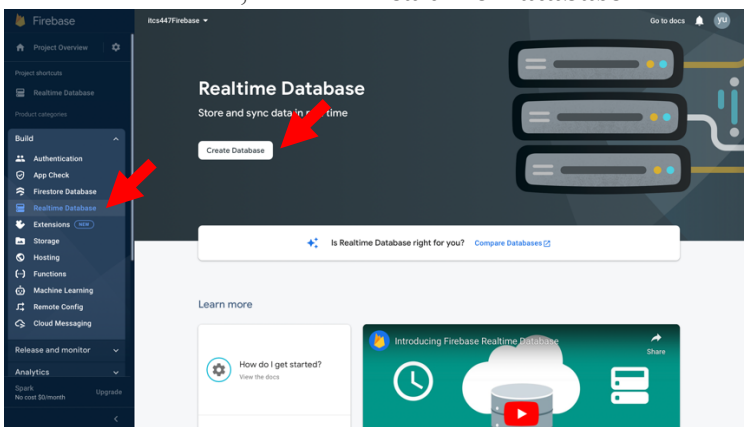


10. Go to “General” tab and you will see your project API. Remark: This API key will be used in ESP32 Arduino code later.

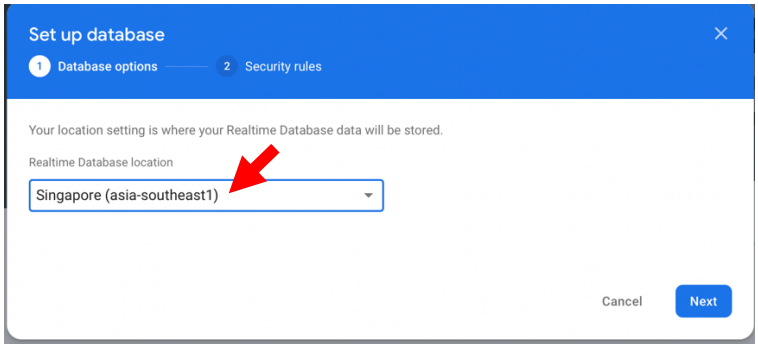


Set up Realtime Database

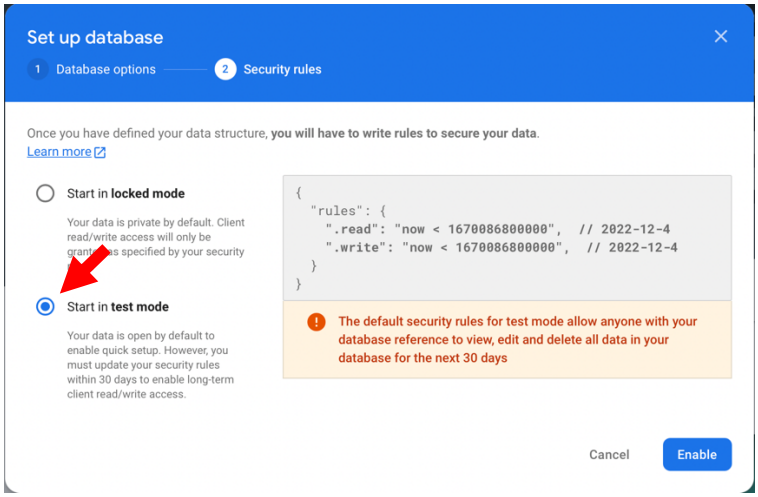
11. On the left sidebar, click on **Realtime Database** and then click on **Create Database**.



12. Select database location. It should be the closet to your location.

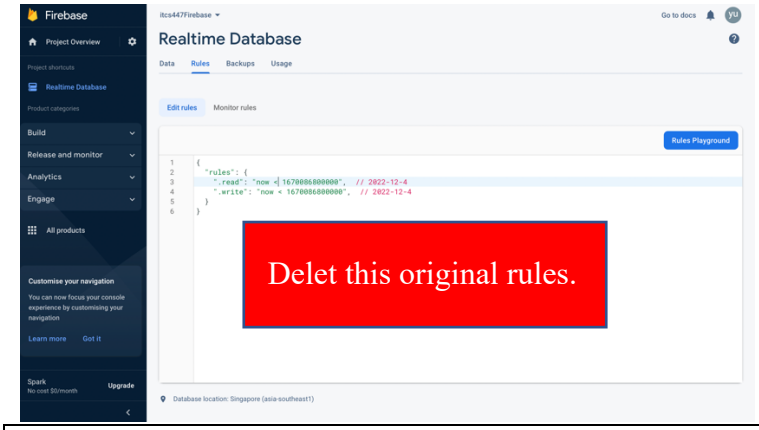


13. Select “Start in test mode”.

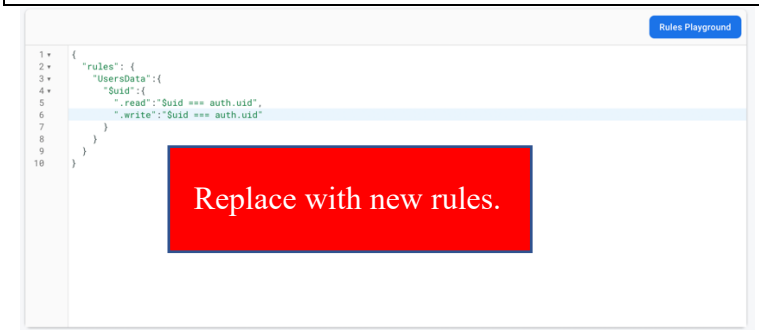


14. Now the database is created. You will get the database URL. Remark: This database URL will be used in ESP32 Arduino code later.

15. Let's set up the rules for your database.
    Delete the original rules and replace the new rules as shown in below.

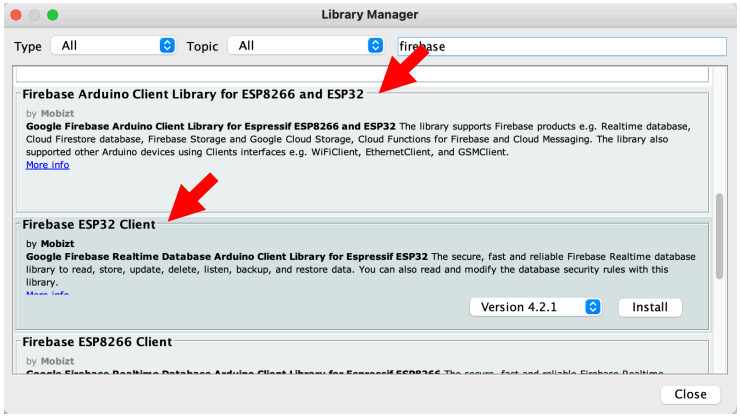



```
{
  "rules": {
    "UsersData":{
      "$uid":{
        ".read":"$uid === auth.uid",
        ".write":"$uid === auth.uid"
      }
    }
  }
}
```

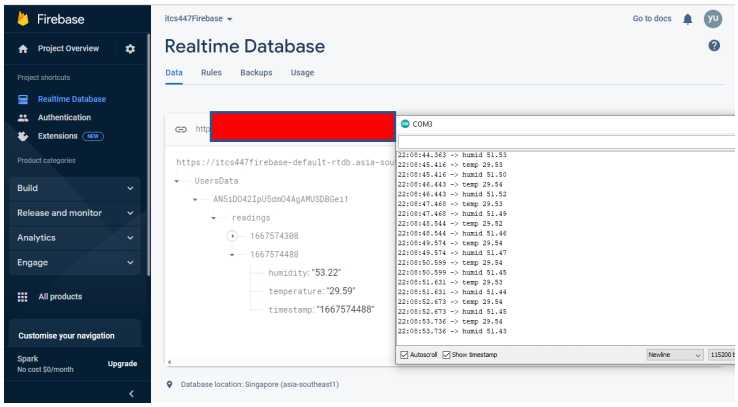


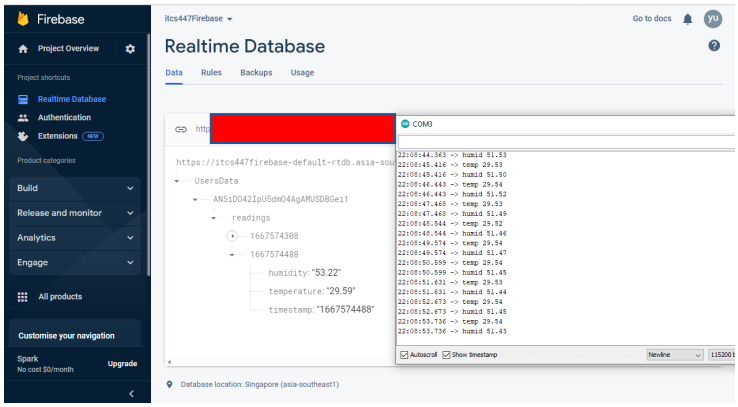
ESP32 and AHT20

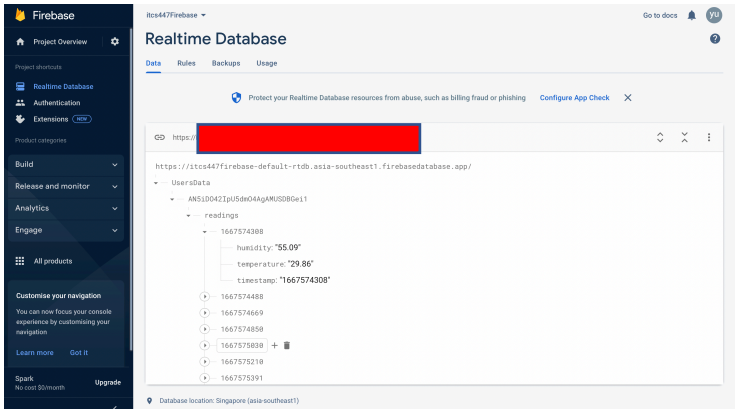
16. Let's wire up ESP32 and AHT20.
17. Install the required libraries "Firebase Arduino Client Library for ESP8266 and ESP32" and "Firebase ESP32 Client" libraries.



18. Upload “ESP32_AHT20_Firebase_Realtime_Database.ino” code to your ESP32.
    a. At “config.h”, replace your WiFi ssid and password.
    b. #define API_KEY “REPLACE_WITH_YOUR_PROJECT_API_KEY”
    c. #define USER_EMAIL “REPLACE_WITH_THE_USER_EMAIL”
    d. #define USER_PASSWORD”REPLACE_WITH_THE_USER_PASSWORD”
    e. #define DATABASE_URL “REPLACE_WITH_YOUR_DATABASE_URL”
19. Example results





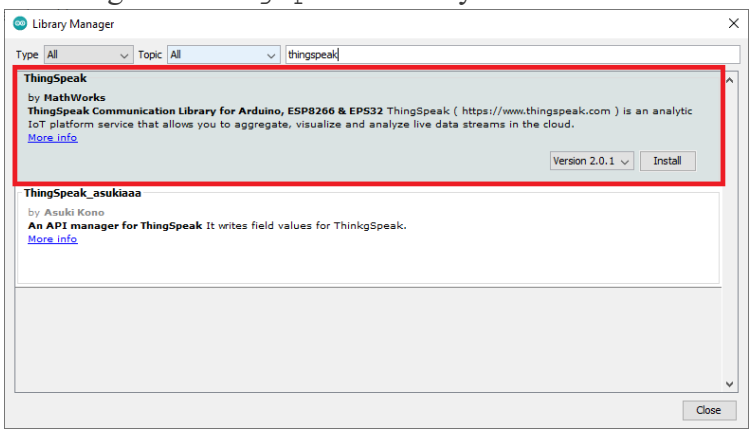


Your results

Part C: ESP32 & ThingSpeak

ThingSpeak is another IoT cloud server that allows you to publish your sensor readings to their website and plot them in charts with timestamps. Then, you can access your readings from anywhere in the world.

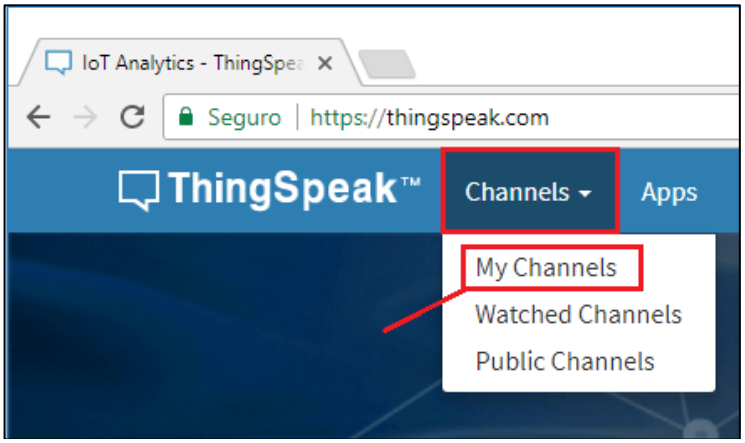
1. Installing the `ThingSpeak` Library



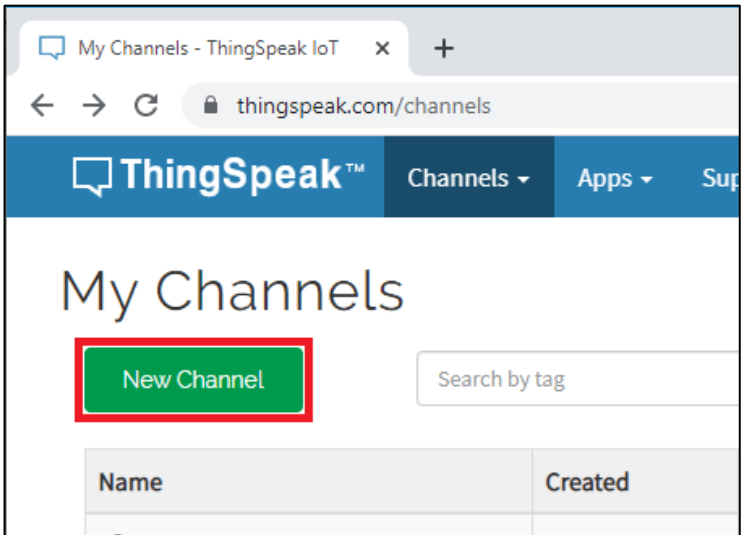
2. Register for ThingSpeak and MathWorks.

Creating New Channel

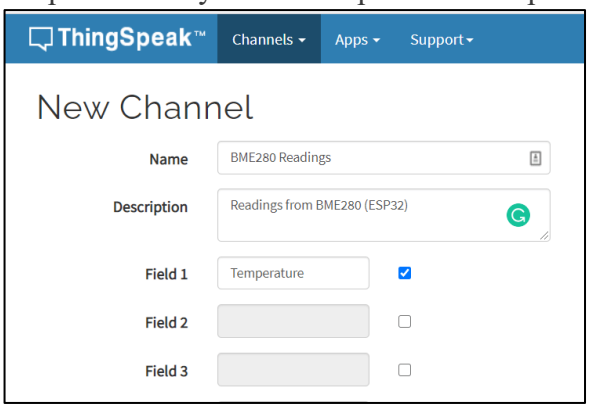
3. After your account is ready, sign in, open the "**Channels**" tab and select "**My Channels**".



4. Press the "New Channel" button to create a new channel.



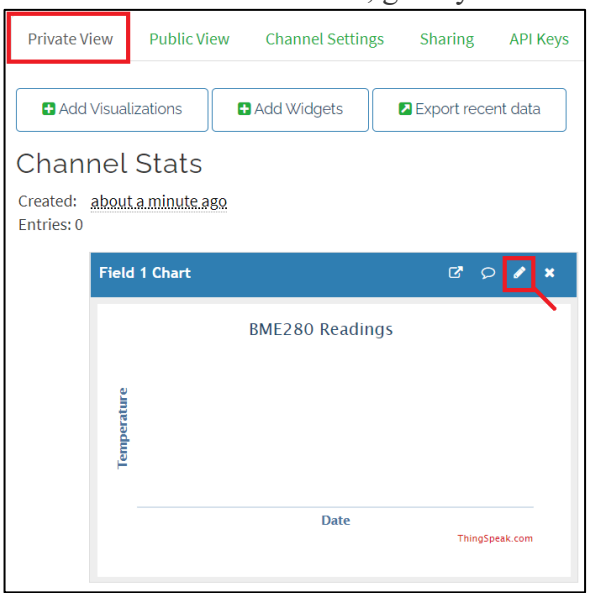
5. Type a name for your channel and add a description. In this example, we'll just publish temperature. If you want to publish multiple readings, you can enable more fields.



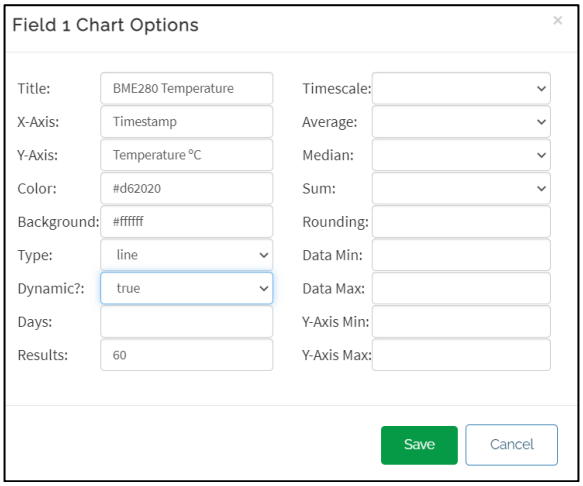
6. Click the **Save Channel** button to create and save your channel.

Customizing Chart

7. The chart can be customized, go to your **Private View** tab and click on the edit icon.

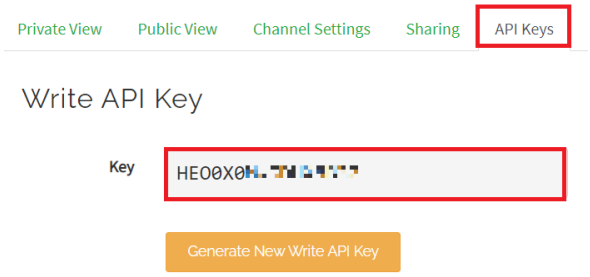


8. You can give a title to your chart, customize the background color, x and y axis, and much more. When you're done, press the "**Save**" button.

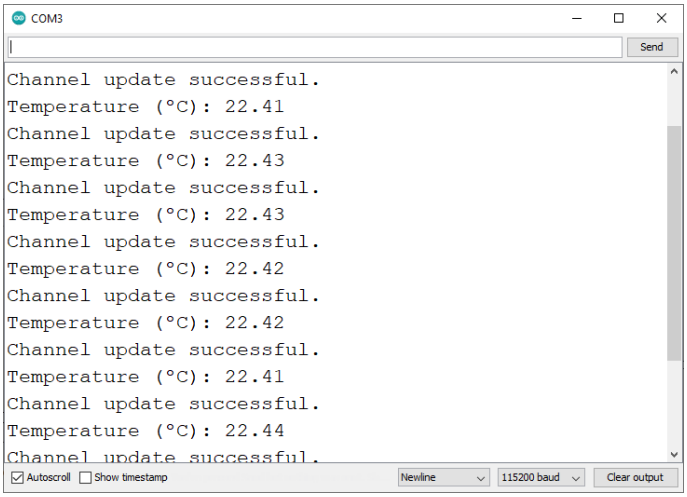


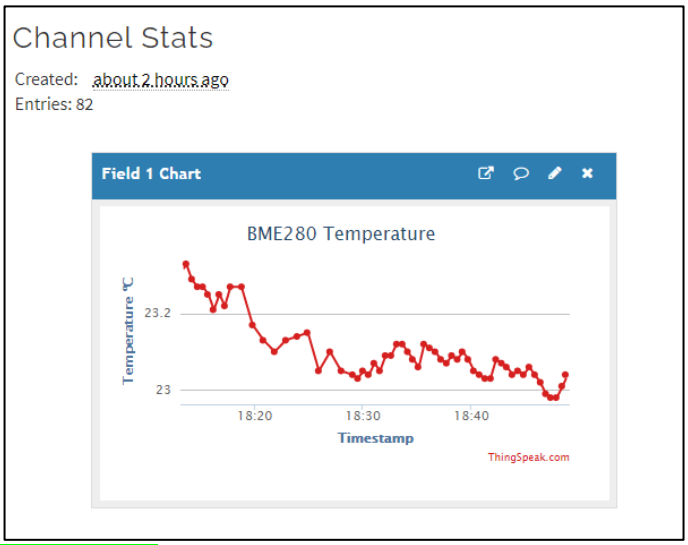
API Key

9. To send values from the ESP32 to ThingSpeak, you need the Write API Key. Open the "**API Keys**" tab and copy the Write API Key to a safe place because you'll need it in a moment.



10. Upload "ESP32_AHT20_ThingSpeak.ino" code to your ESP32.
    a. `const char* ssid = "REPLACE_WITH_YOUR_SSID";`
    b. `const char* password = "REPLACE_WITH_YOUR_PASSWORD";`
11. You need to insert the number of the channel that you're publishing to. If you only have one channel created in ThingSpeak, the channel number is 1. Otherwise, you can see the number of the channel on the **Private View** tab.
    a. `unsigned long myChannelNumber = 1;`
    b. `const char * myWriteAPIKey = "XXXXXXXXXXXXXXXX";`
12. Example results

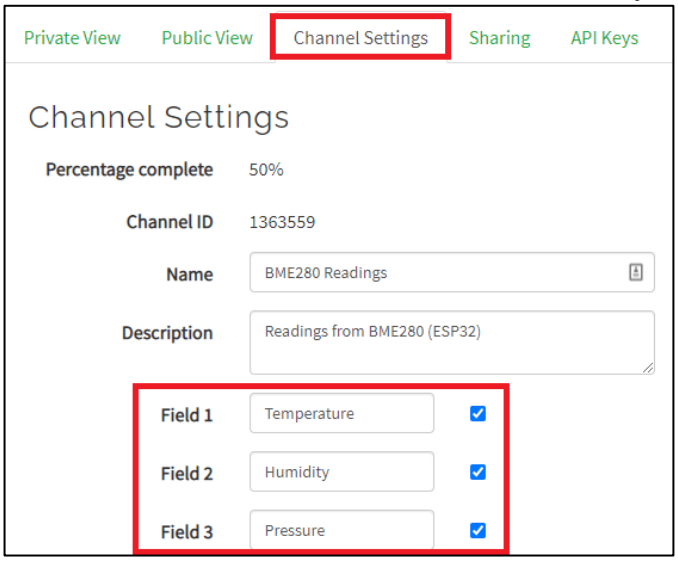




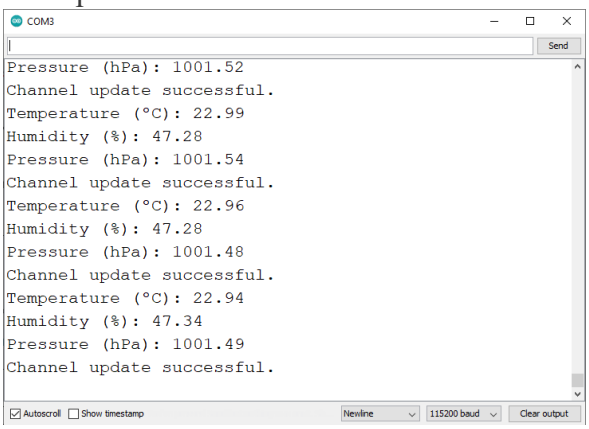
Your results

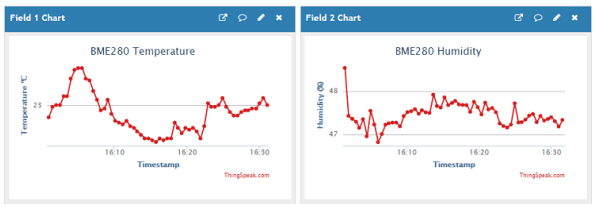
Sending Multiple Fields (Temperature, and Humidity)

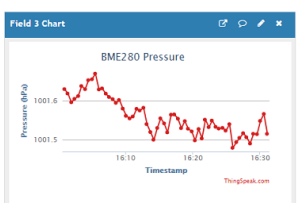
13. In this section, you'll learn how to send multiple fields this is sending more than one value at a time we'll send temperature, humidity, and pressure readings.
14. First, you need to create more fields in your ThingSpeak account. This is simple. You need to go to your **Channel Settings** and add as many fields as you want. In this case, we've added two more fields, one for the humidity and another for the pressure.



15. Now, if you go to the **Private View** tab, you should have three charts. Edit the newly created charts with a title and axis labels.
16. Edit the previous source code "ESP32_AHT20_ThingSpeak.ino" for multiple fields.
17. Example results







Your results and its source code