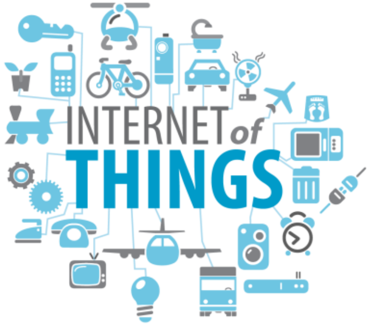
UNIVERSITY OF SCIENCE - VNUHCM

Faculty of Information Technology

INTERNET OF THINGS

3.4

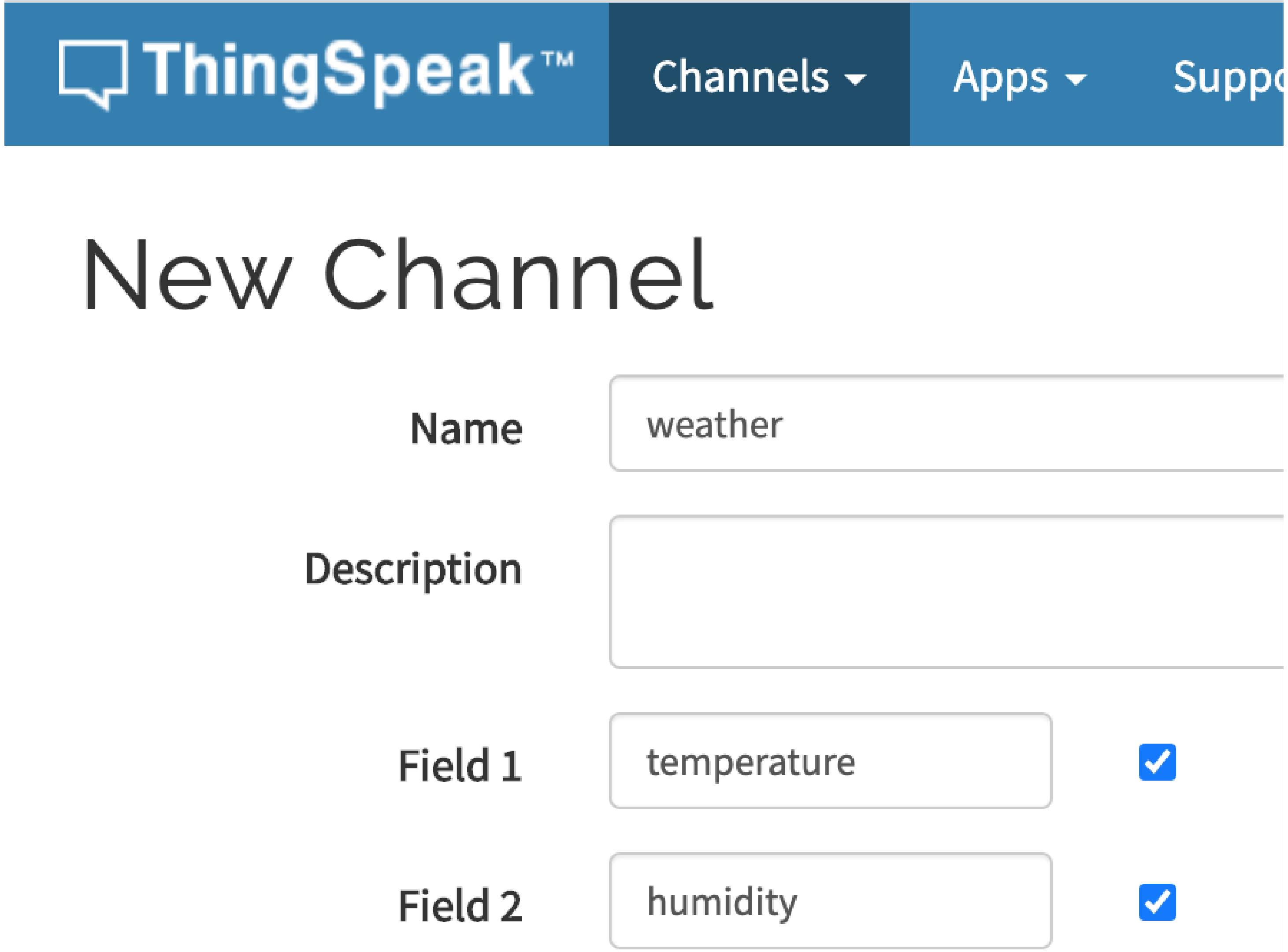
ESP8266 STORE DATA IN CLOUD



LECTURER: CAO XUÂN NAM

THINGSPEAK

using **ThingSpeak Library**



New Channel

Name: weather

Description:

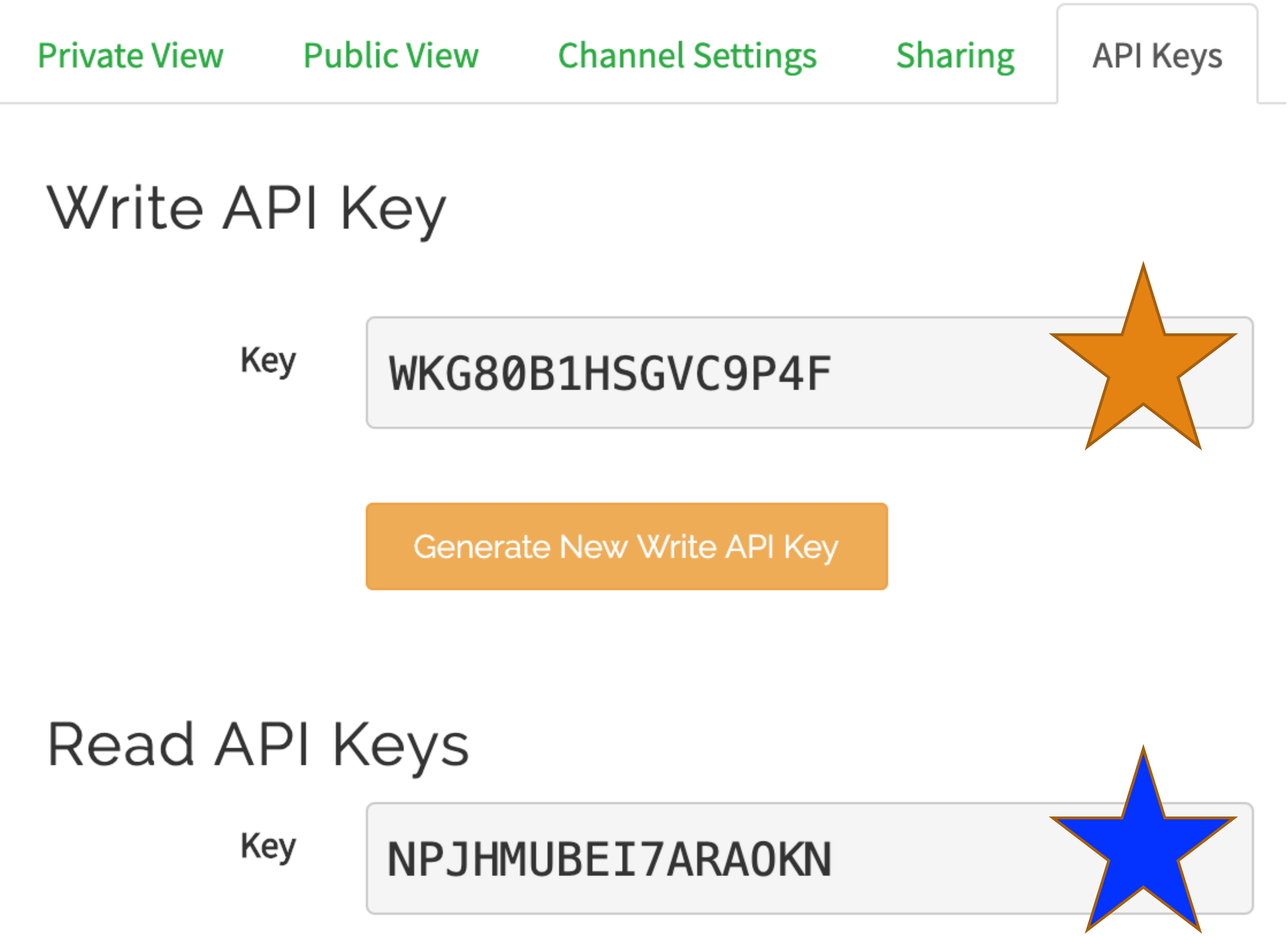
Field 1: temperature ☑

Field 2: humidity ☑

Step 1: Sign up new account in **mathworks.com**

Step 2: Sign in with mathworks account in **thingspeak.com**

Step 3: Go to **Channels > New Channel > Input Fields > Save Channel**



Write API Key

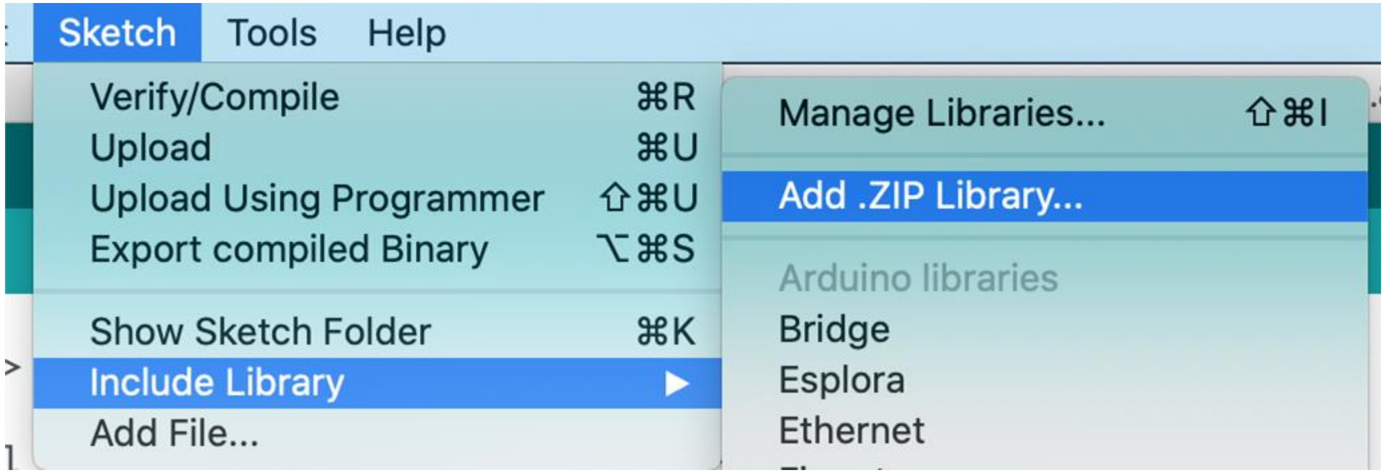
Key WKG80B1HSGVC9P4F

Read API Keys

Key NPJHMUBEI7ARAOKN

Step 4: Select ***API Keys*** Tab.

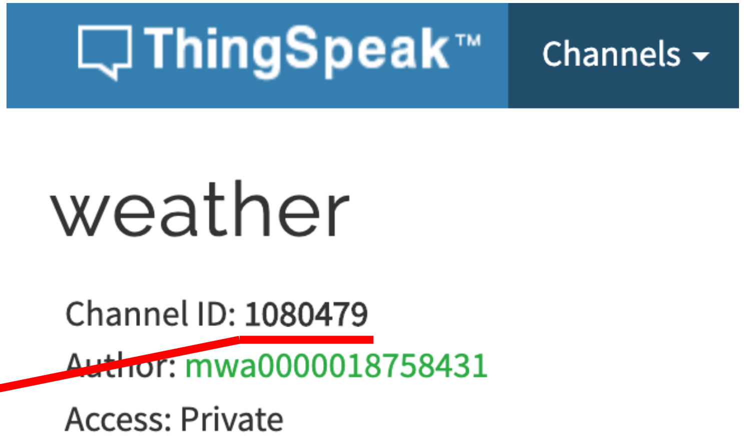
Remember ***Write API Key***



Step 5: Download Zip file to your computer

https://github.com/mathworks/thingspeak-arduino

Step 6: In Arduino IDE, choose ***Sketch > Include Library > Add Zip Library > Select Zip file***



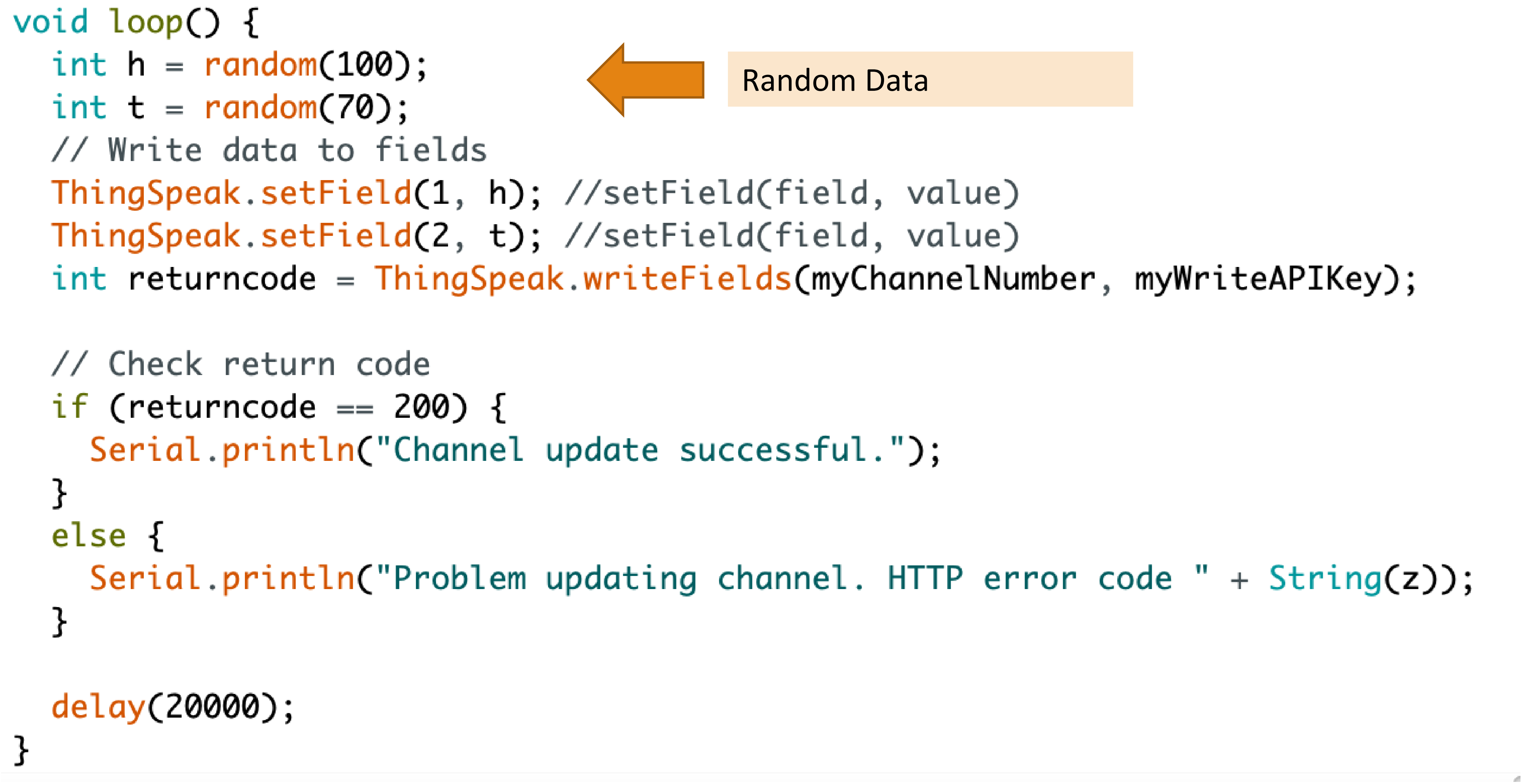
weather

Channel ID: 1080479

Author: mwa0000018758431

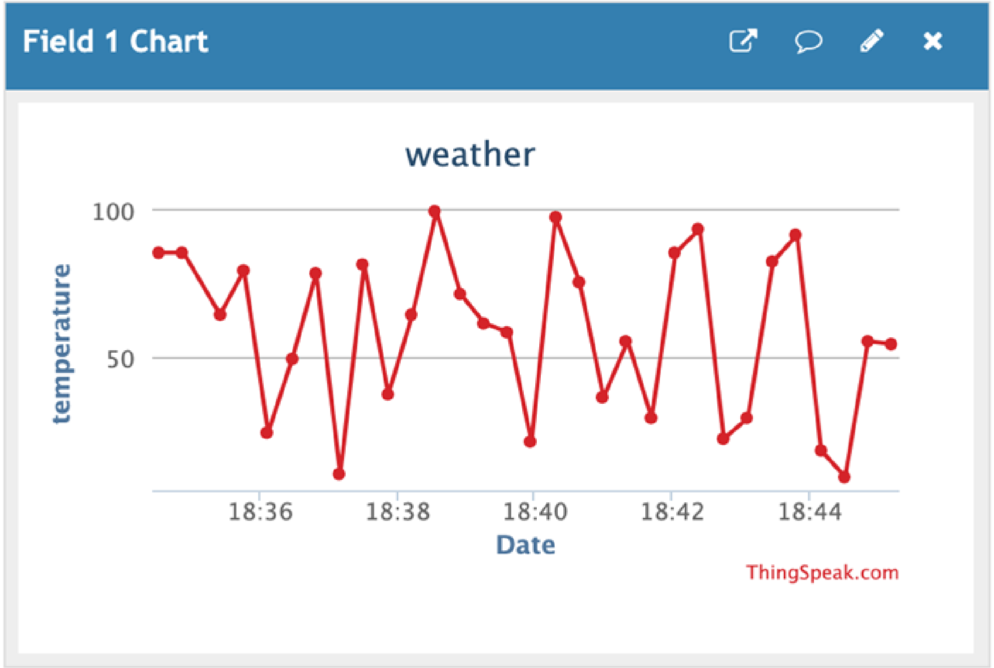
Access: Private

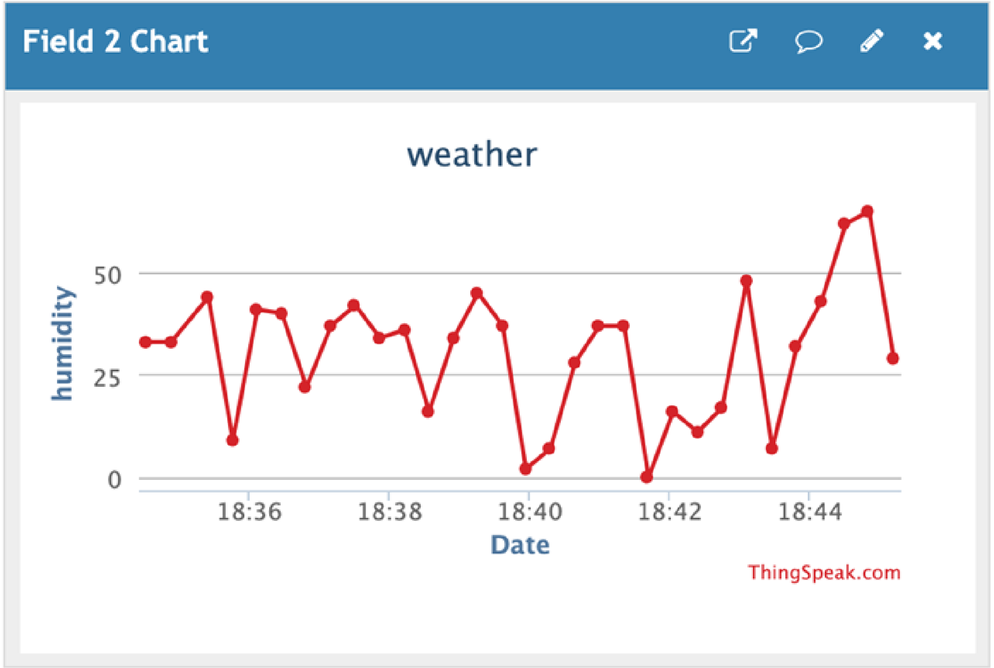
```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>

char ssid[] = "your wifi name";
char pass[] = "your wifi password";

unsigned long myChannelNumber = 0;  // Replace the 0 with your channel number
const char * myWriteAPIKey = "your Write API";    // Paste your ThingSpeak Write API Key

WiFiClient  client;

void setup() {
  //Connect to WiFi Network. DIY

  ThingSpeak.begin(client);
}
```

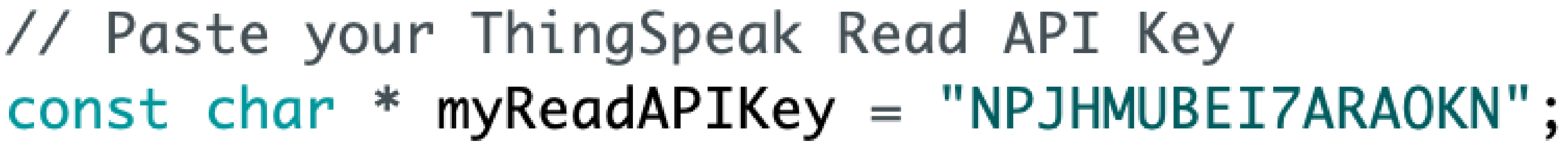


```
void loop() {
  int h = random(100);
  int t = random(70);
  // Write data to fields
  ThingSpeak.setField(1, h); //setField(field, value)
  ThingSpeak.setField(2, t); //setField(field, value)
  int returncode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

  // Check return code
  if (returncode == 200) {
    Serial.println("Channel update successful.");
  }
  else {
    Serial.println("Problem updating channel. HTTP error code " + String(z));
  }

  delay(20000);
}
```

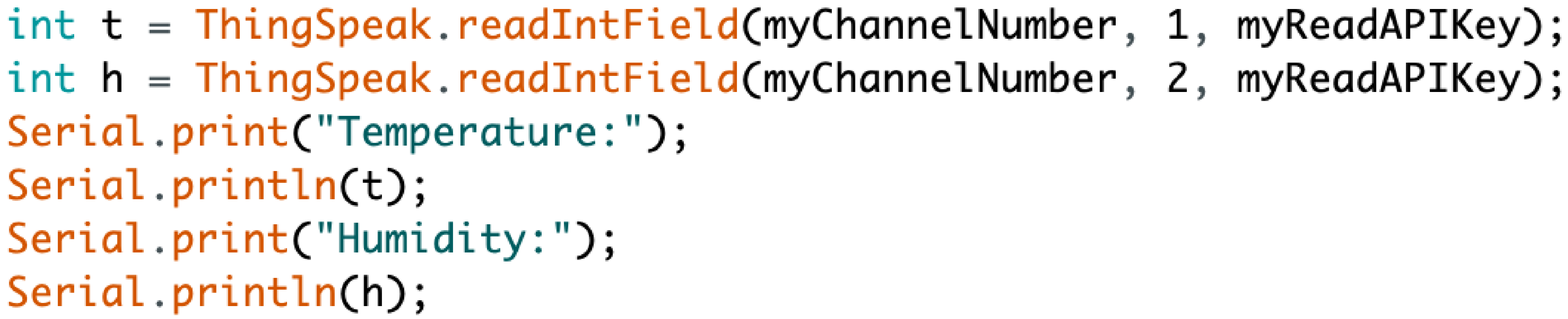
Random Data


Field 1 Chart
weather
100
50
temperature
18:36
18:38
18:40
18:42
18:44
Date
ThingSpeak.com


Field 2 Chart
weather
50
25
0
humidity
18:36
18:38
18:40
18:42
18:44
Date
ThingSpeak.com



```
// Paste your ThingSpeak Read API Key
const char * myReadAPIKey = "NPJHMUBEI7ARAOKN";
```



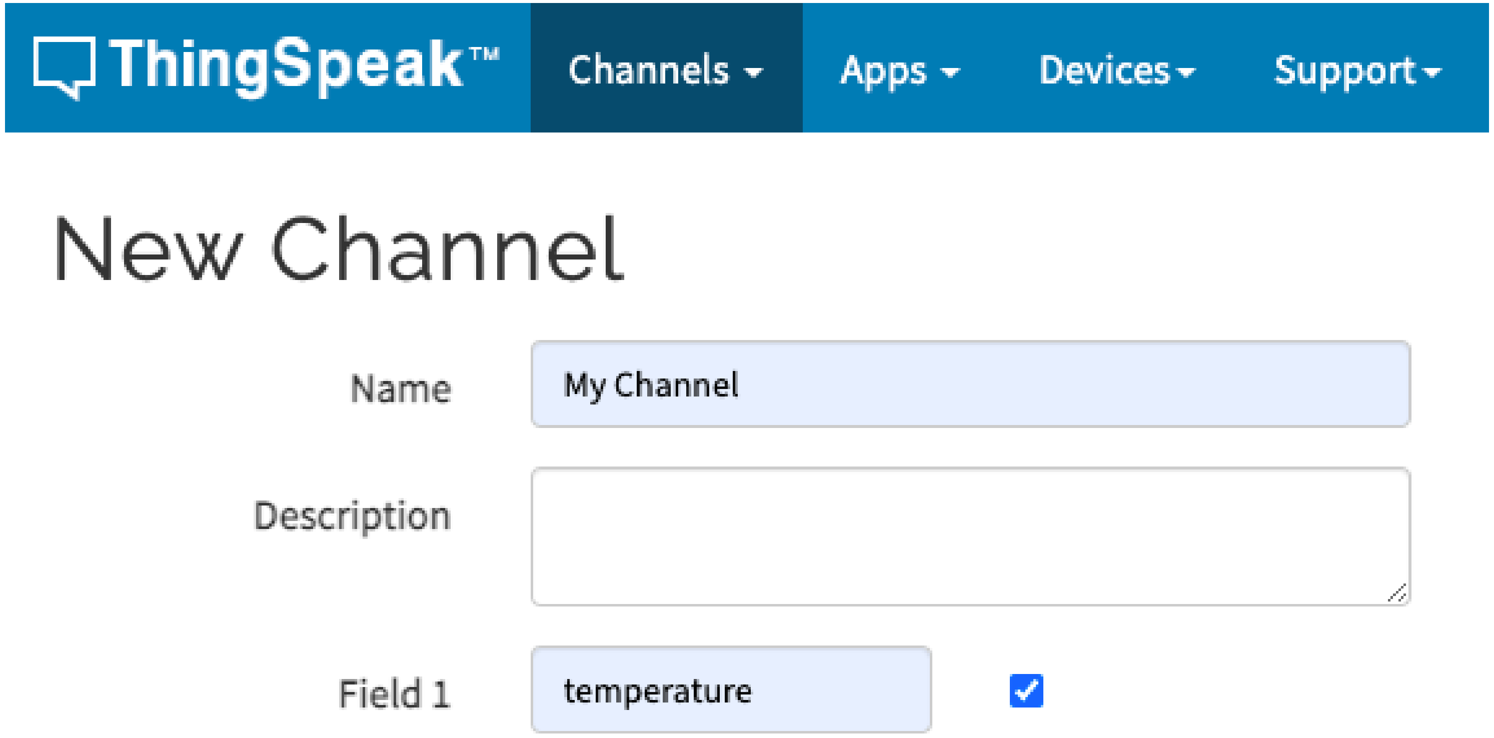
```
int t = ThingSpeak.readIntField(myChannelNumber, 1, myReadAPIKey);
int h = ThingSpeak.readIntField(myChannelNumber, 2, myReadAPIKey);
Serial.print("Temperature:");
Serial.println(t);
Serial.print("Humidity:");
Serial.println(h);
```

Read data from ThingSpeak

THINGSPEAK

ThingSpeak™

using **Http Request**



New Channel

Name: My Channel

Description:

Field 1: temperature ☑

Step 1: Sign up new account in **mathworks.com**

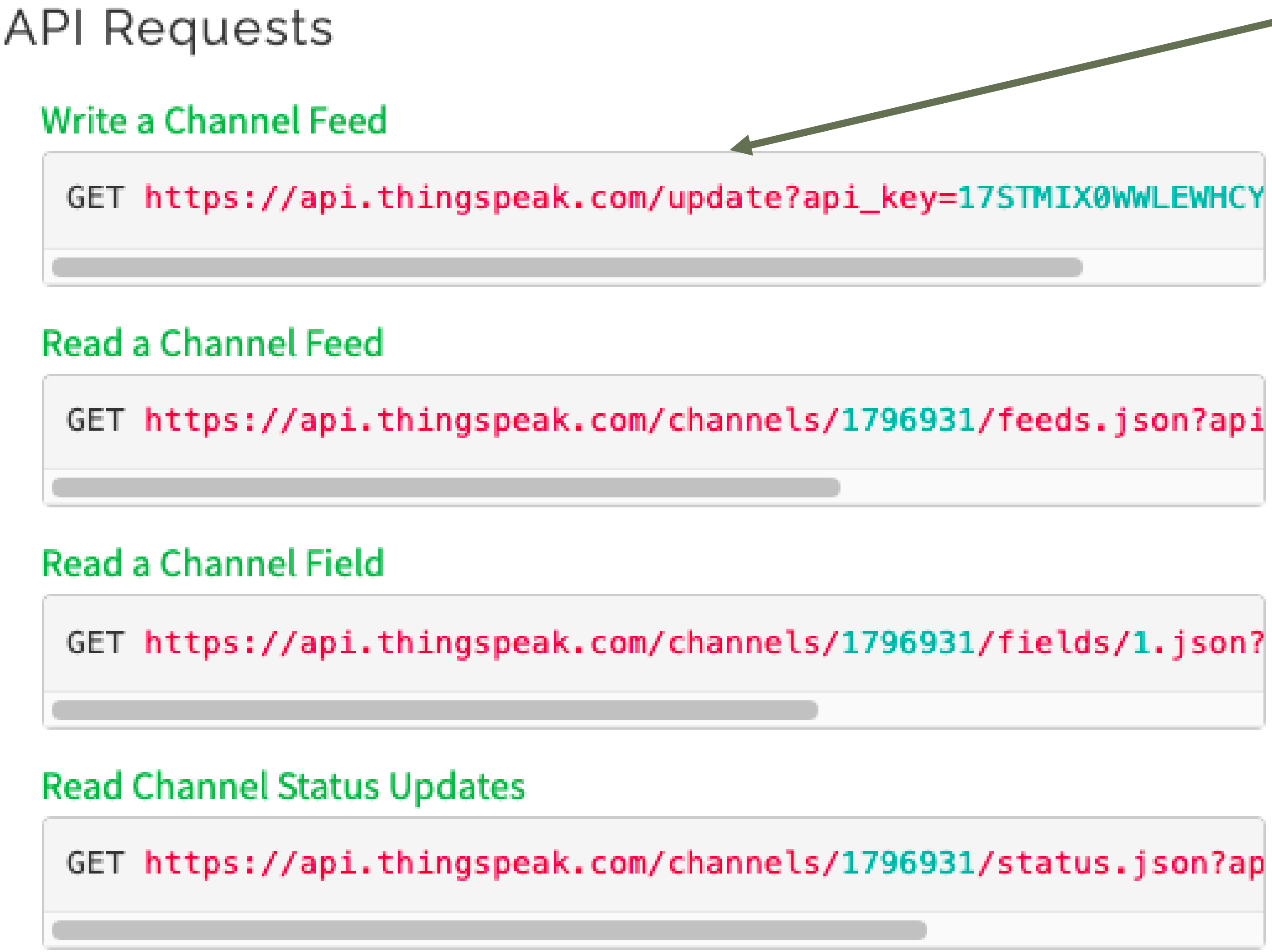
Step 2: Sign in with mathworks account in **thingspeak.com**

Step 3: Go to **Channels > New Channel > Input Fields > Save Channel**

Host

Request

```
GET https://api.thingspeak.com/update?api_key=17STMIX0WWLEWHCY&field1=0
```



API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=17STMIX0WWLEWHCY
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/1796931/feeds.json?api
```

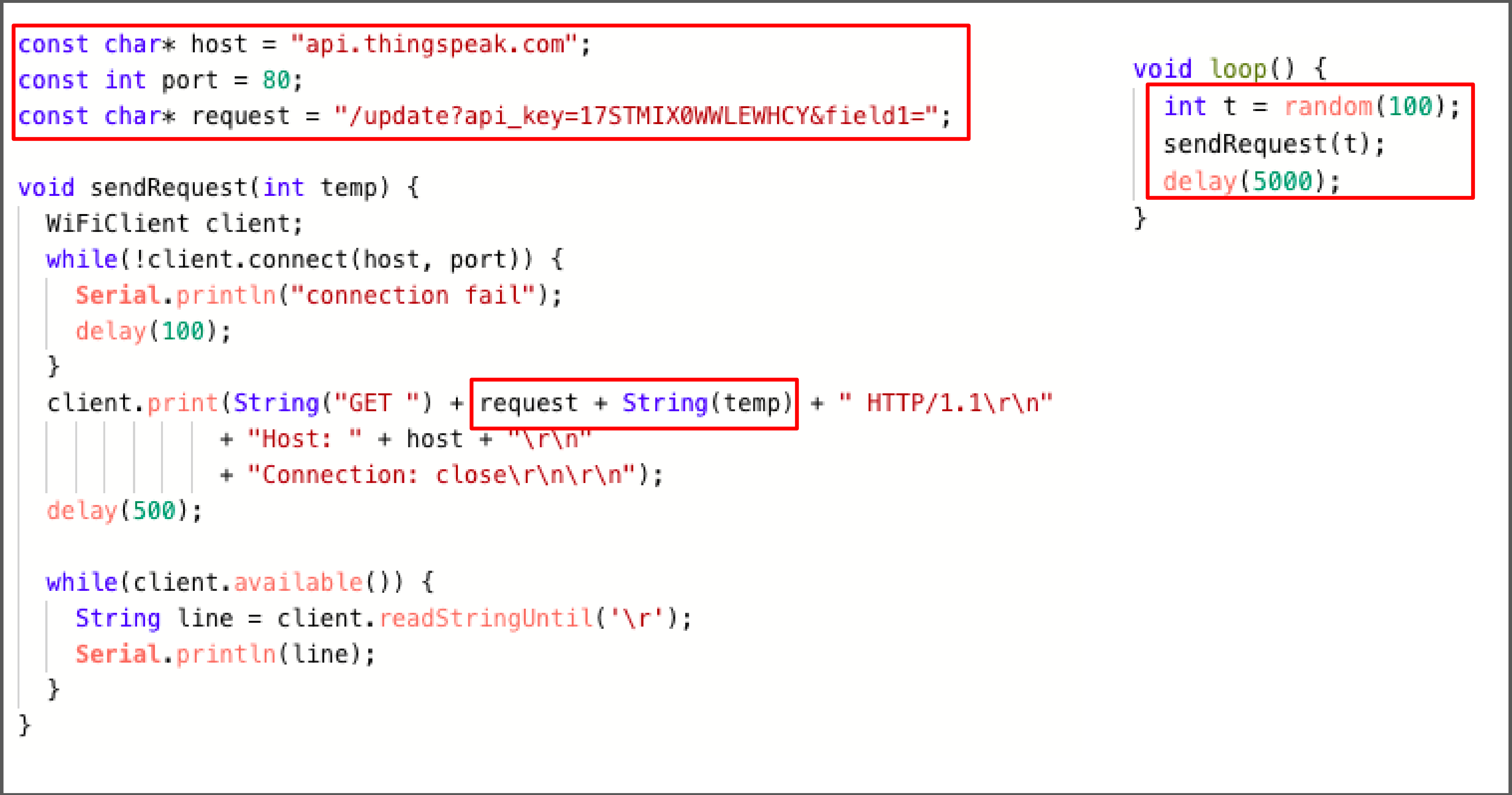
Read a Channel Field

```
GET https://api.thingspeak.com/channels/1796931/fields/1.json?
```

Read Channel Status Updates

```
GET https://api.thingspeak.com/channels/1796931/status.json?ap
```

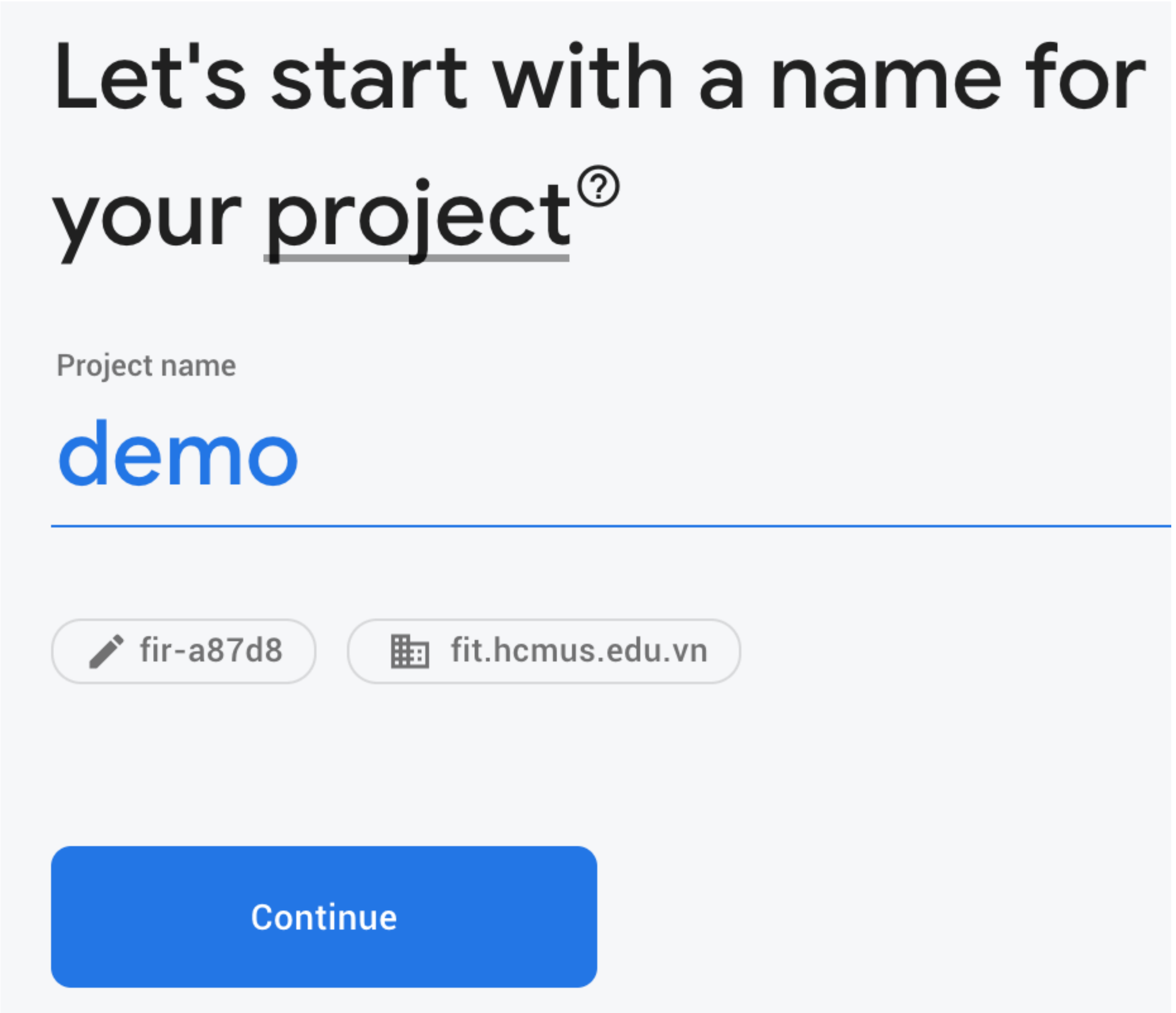
Step 4: Select ***API Keys*** Tab.



```
const char* host = "api.thingspeak.com";
const int port = 80;
const char* request = "/update?api_key=17STMIX0WWLEWHCY&field1=";

void sendRequest(int temp) {
  WiFiClient client;
  while(!client.connect(host, port)) {
    Serial.println("connection fail");
    delay(100);
  }
  client.print(String("GET ") + request + String(temp) + " HTTP/1.1\r\n"
               + "Host: " + host + "\r\n"
               + "Connection: close\r\n\r\n");
  delay(500);

  while(client.available()) {
    String line = client.readStringUntil('\r');
    Serial.println(line);
  }
}
```

```
void loop() {
  int t = random(100);
  sendRequest(t);
  delay(5000);
}
```

FIREBASE



Let's start with a name for your project

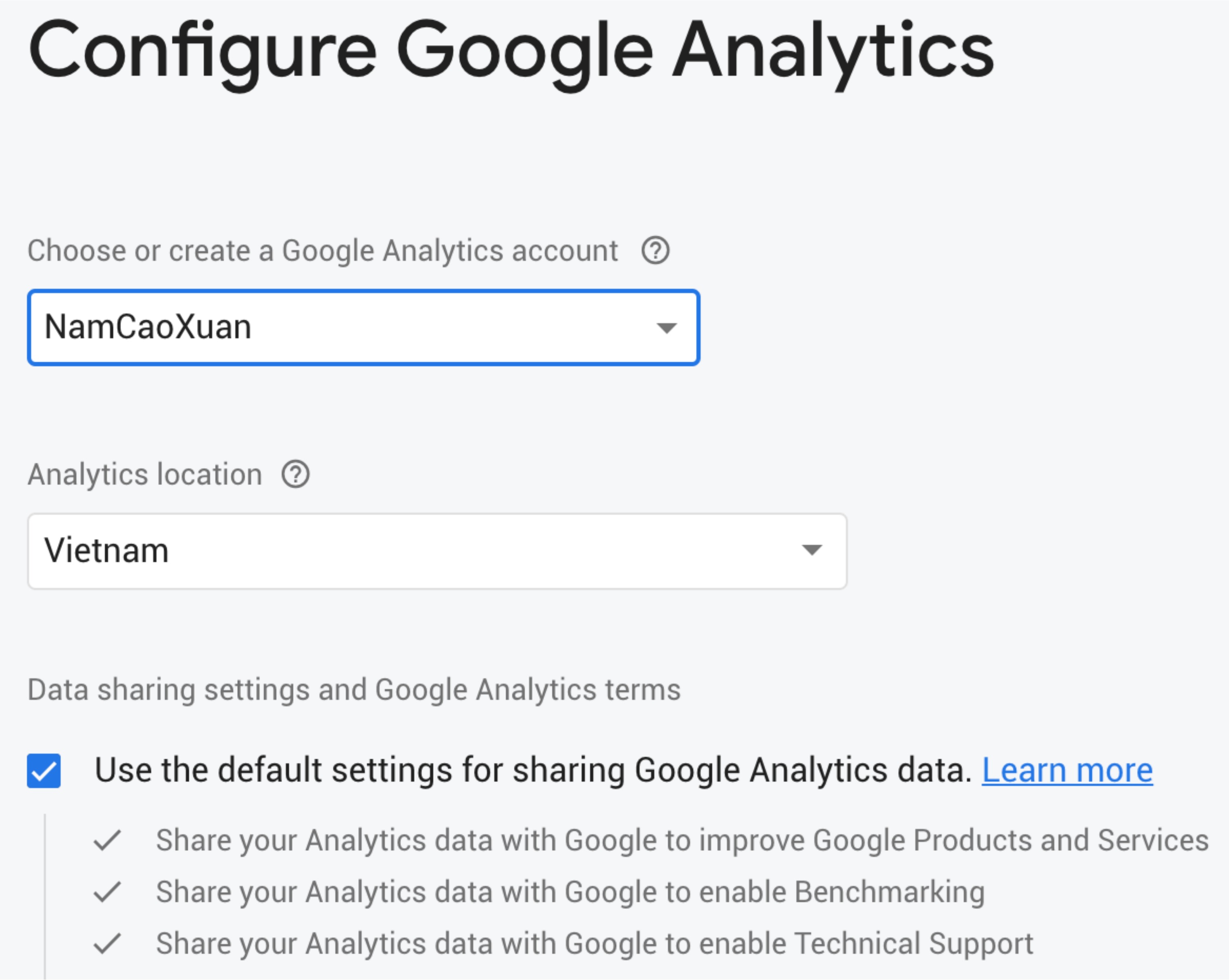
Project name

demo

Step 1: Sign up/Sign in **firebase.google.com** using Google account

Step 2: Add new project

Step 3: Named for your project



Configure Google Analytics

Choose or create a Google Analytics account ⓘ

NamCaoXuan

Analytics location ⓘ

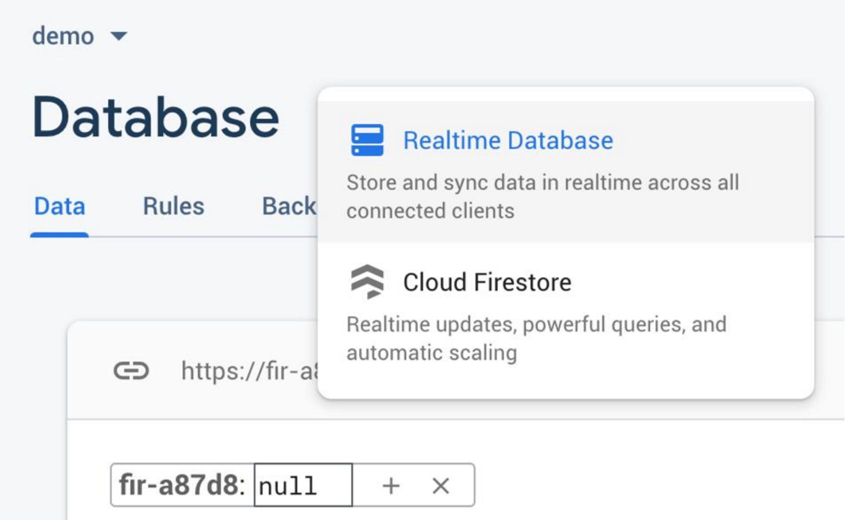
Vietnam

Data sharing settings and Google Analytics terms

☑ Use the default settings for sharing Google Analytics data. Learn more

- ✓ Share your Analytics data with Google to improve Google Products and Services
- ✓ Share your Analytics data with Google to enable Benchmarking
- ✓ Share your Analytics data with Google to enable Technical Support

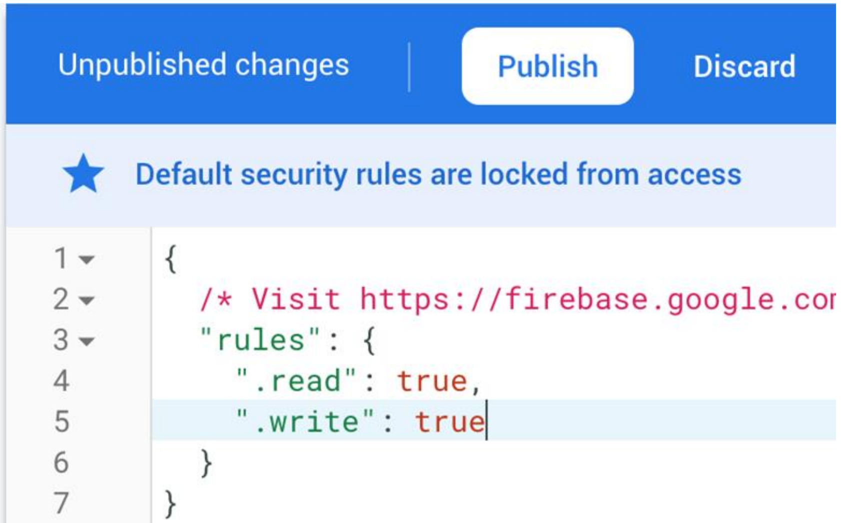
Step 4: Configure Google Analytics (optional)



Realtime Database: Store and sync data in realtime across all connected clients

Cloud Firestore: Realtime updates, powerful queries, and automatic scaling

fir-a87d8: null



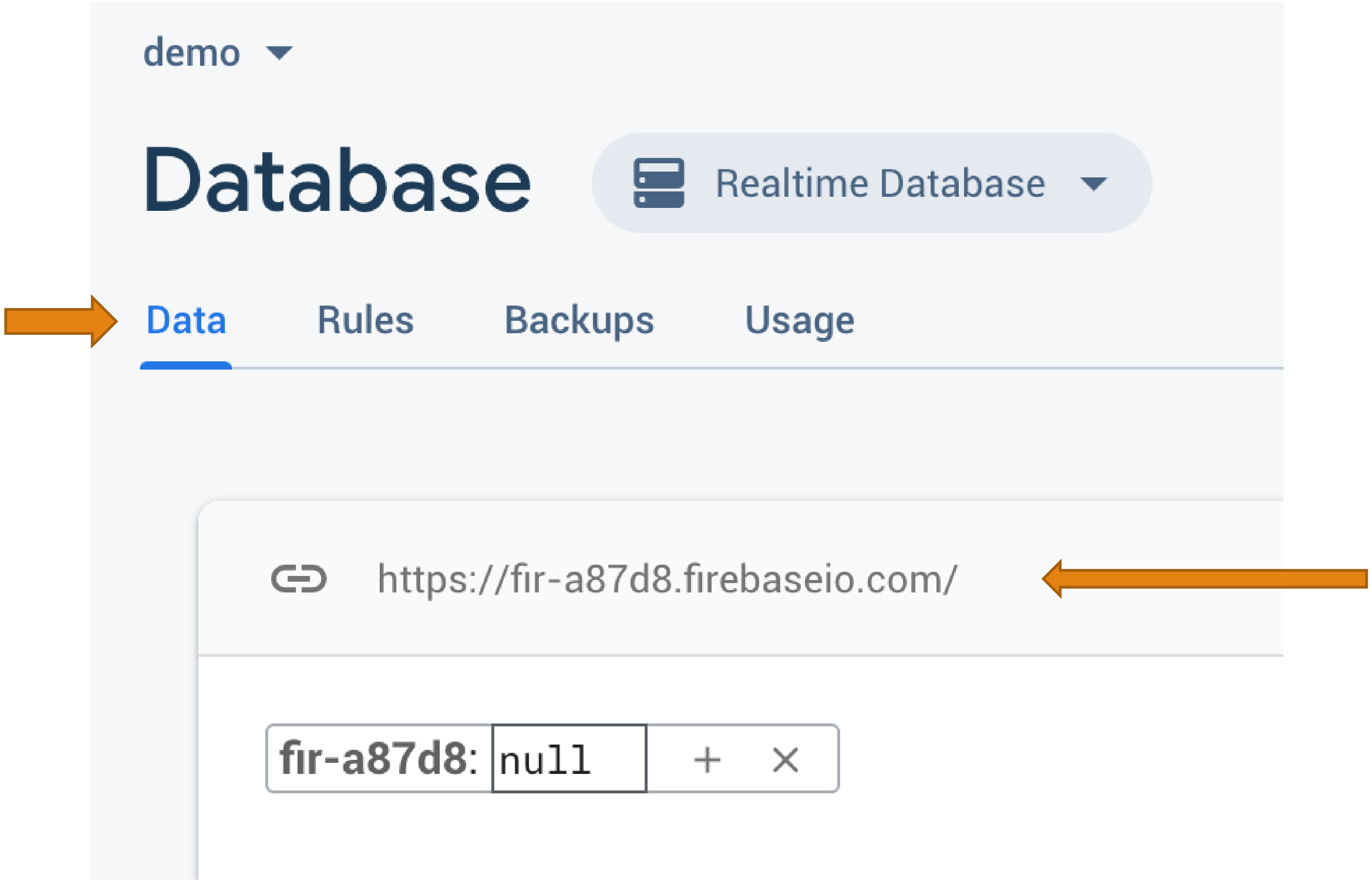
Default security rules are locked from access

```
{
  /* Visit https://firebase.google.co
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Step 5: Developer Menu > Database > Create database

Step 6: Change Database mode is **Realtime Database**

Step 7: Choose Rules tab > Edit rules > set “**true**” for read and write rule > Publish



demo

Database

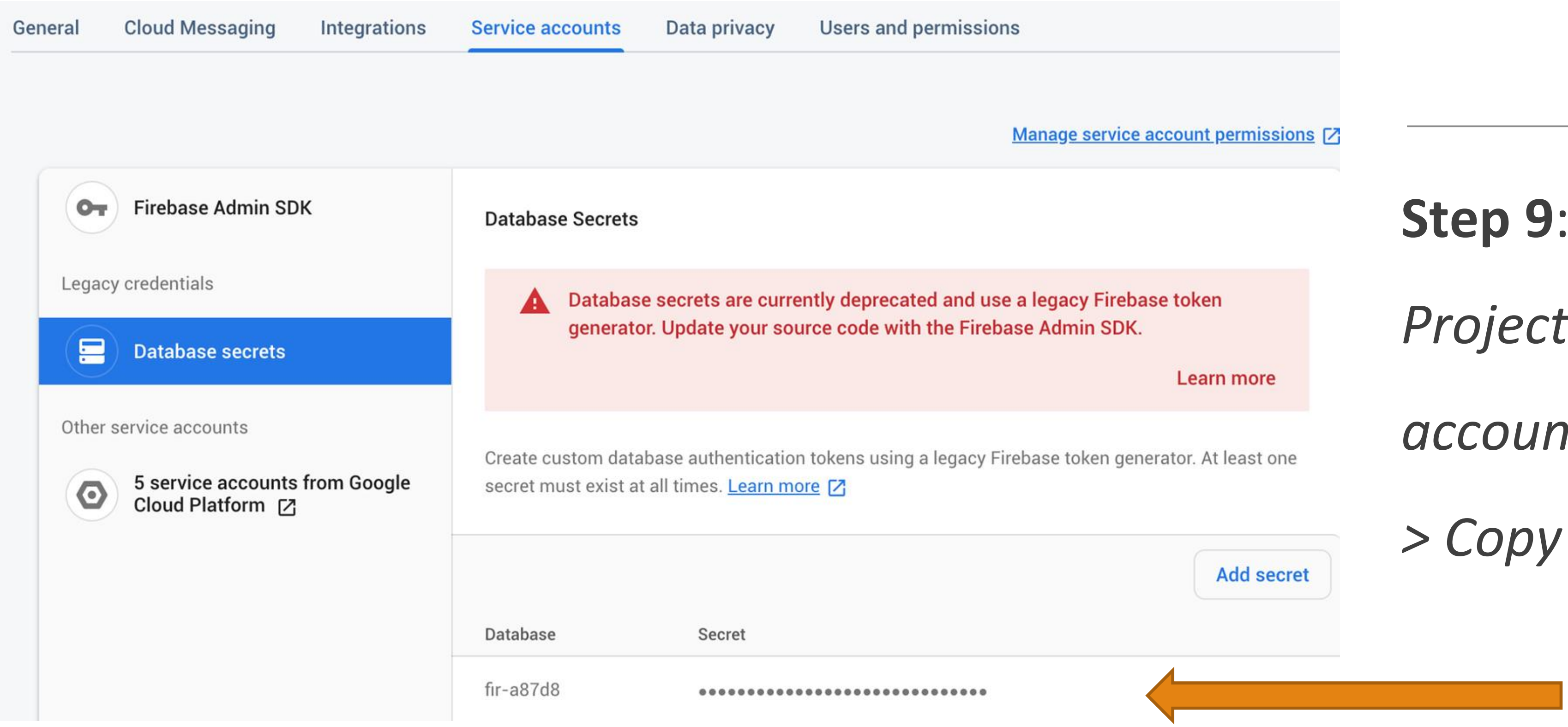
Realtime Database

Data Rules Backups Usage

https://fir-a87d8.firebaseio.com/

fir-a87d8: null

Step 8: Copy your firebase host



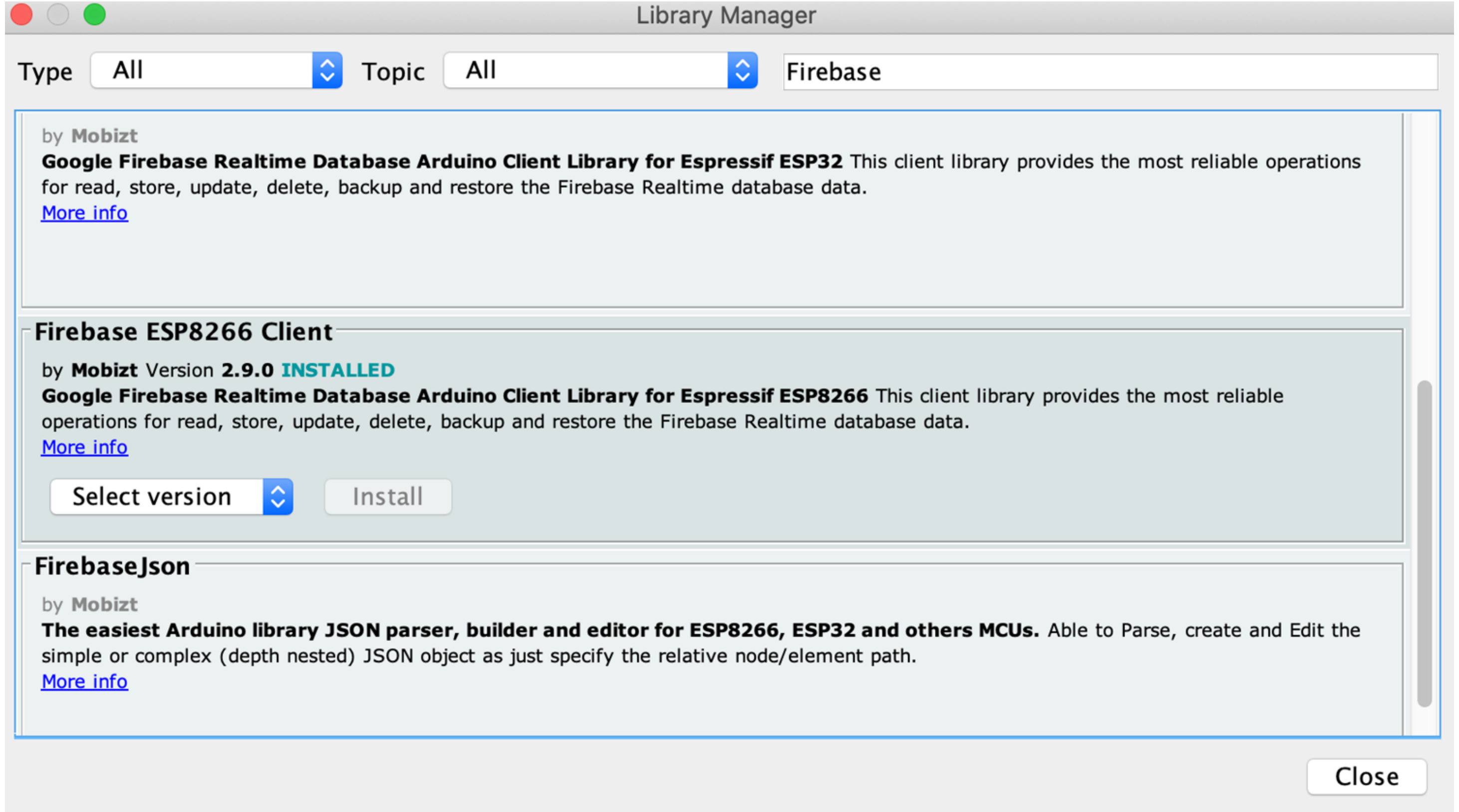
Database Secrets

Database secrets are currently deprecated and use a legacy Firebase token generator. Update your source code with the Firebase Admin SDK.

Create custom database authentication tokens using a legacy Firebase token generator. At least one secret must exist at all times. Learn more

Database	Secret
fir-a87d8	••••••••••••••••••••••••••••••

Step 9: Select *Gear icon > Project Settings > Service accounts > Database secrets > Copy Secret key*



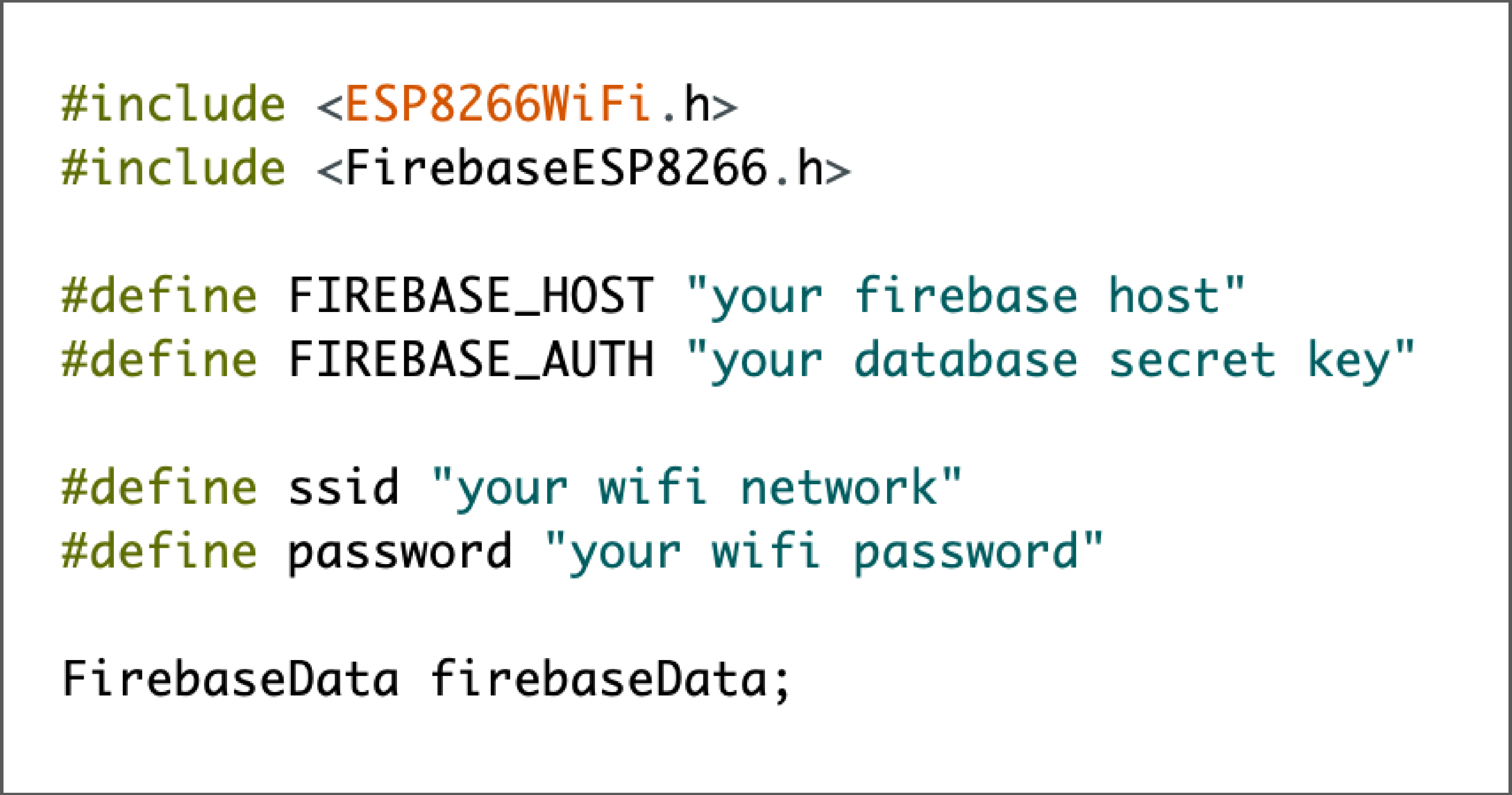
by **Mobizt**
Google Firebase Realtime Database Arduino Client Library for Espressif ESP32 This client library provides the most reliable operations for read, store, update, delete, backup and restore the Firebase Realtime database data.
More info

Firebase ESP8266 Client
by **Mobizt** Version **2.9.0** **INSTALLED**
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.
More info

FirebaseJson
by **Mobizt**
The easiest Arduino library JSON parser, builder and editor for ESP8266, ESP32 and others MCUs. Able to Parse, create and Edit the simple or complex (depth nested) JSON object as just specify the relative node/element path.
More info

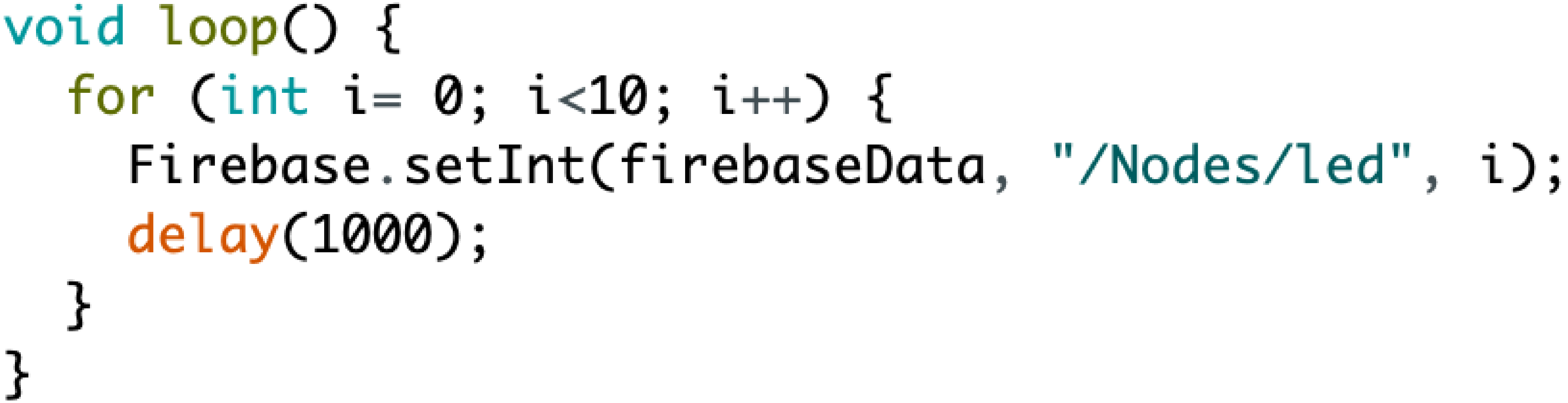
Step 10: In Arduino IDE, Sketch > Include Library > Manage Libraries…

Step 11: Search “**Firebase**” and install the latest version of “**Firebase ESP8266 Client**”



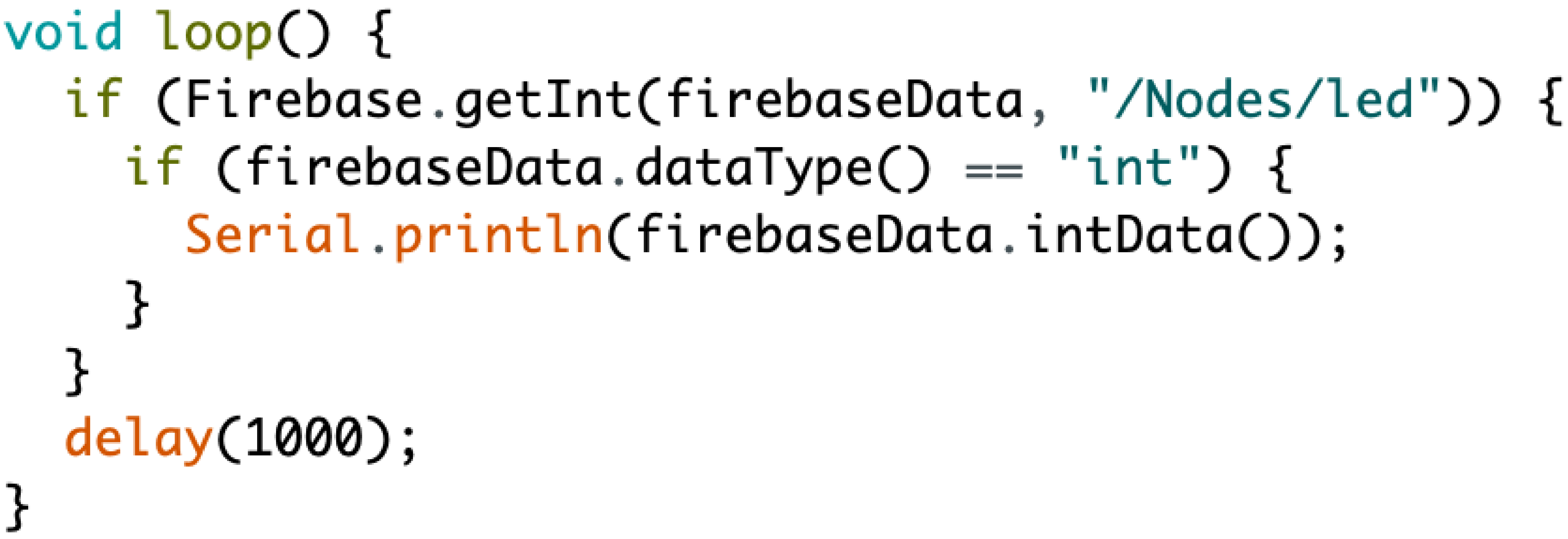
```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

#define FIREBASE_HOST "your firebase host"
#define FIREBASE_AUTH "your database secret key"

#define ssid "your wifi network"
#define password "your wifi password"

FirebaseData firebaseData;
```

```
void setup() {
  //Connect to wifi network. DIY

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);

  if (!Firebase.beginStream(firebaseData, "/Nodes/led"))
  {
      Serial.println("Could not begin stream");
      Serial.println("REASON: " + firebaseData.errorReason());
      Serial.println();
  }
}
```



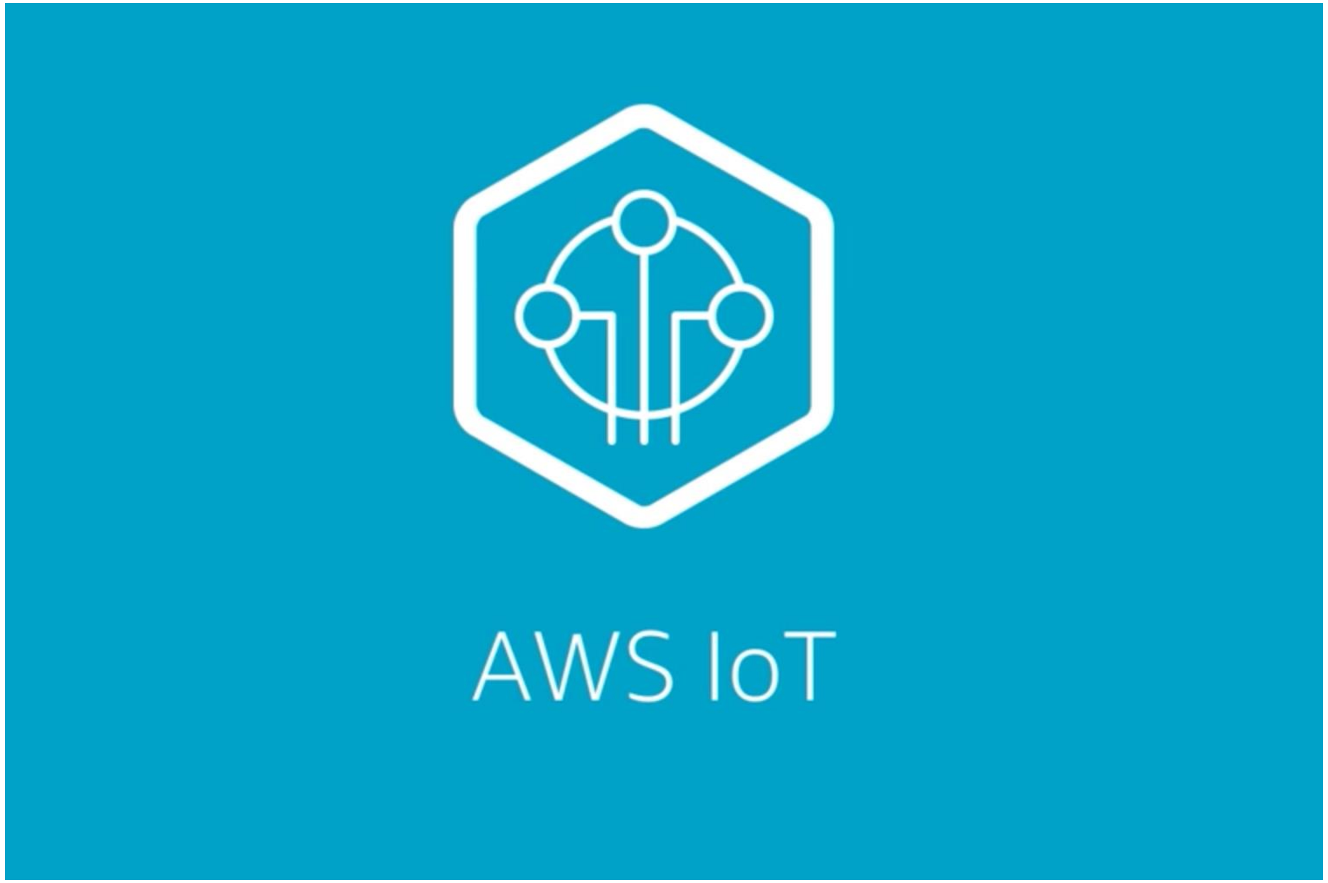
```
void loop() {
  for (int i= 0; i<10; i++) {
    Firebase.setInt(firebaseData, "/Nodes/led", i);
    delay(1000);
  }
}
```

Write data to Firebase



```
void loop() {
  if (Firebase.getInt(firebaseData, "/Nodes/led")) {
    if (firebaseData.dataType() == "int") {
      Serial.println(firebaseData.intData());
    }
  }
  delay(1000);
}
```

Read data from Firebase



AWS IoT