

A photograph showing two glowing heart-shaped objects and one glowing square object held by hands against a dark background. The heart-shaped objects are illuminated from within, showing a gradient of orange and yellow. The square object has a bright blue glow in its center. The hands are positioned to showcase the glowing objects.

Instructions

# Uploading the Arduino Code

@ NYC Resistor

**Step 1: Download Arduino IDE** Version2.3.2 here: <https://www.arduino.cc/en/software>





## Arduino IDE 2.3.2

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

[SOURCE CODE](#)

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

### DOWNLOAD OPTIONS

**Windows** Win 10 and newer, 64 bits  
**Windows** MSI installer  
**Windows** ZIP file

**Linux** AppImage 64 bits (X86-64)  
**Linux** ZIP file 64 bits (X86-64)

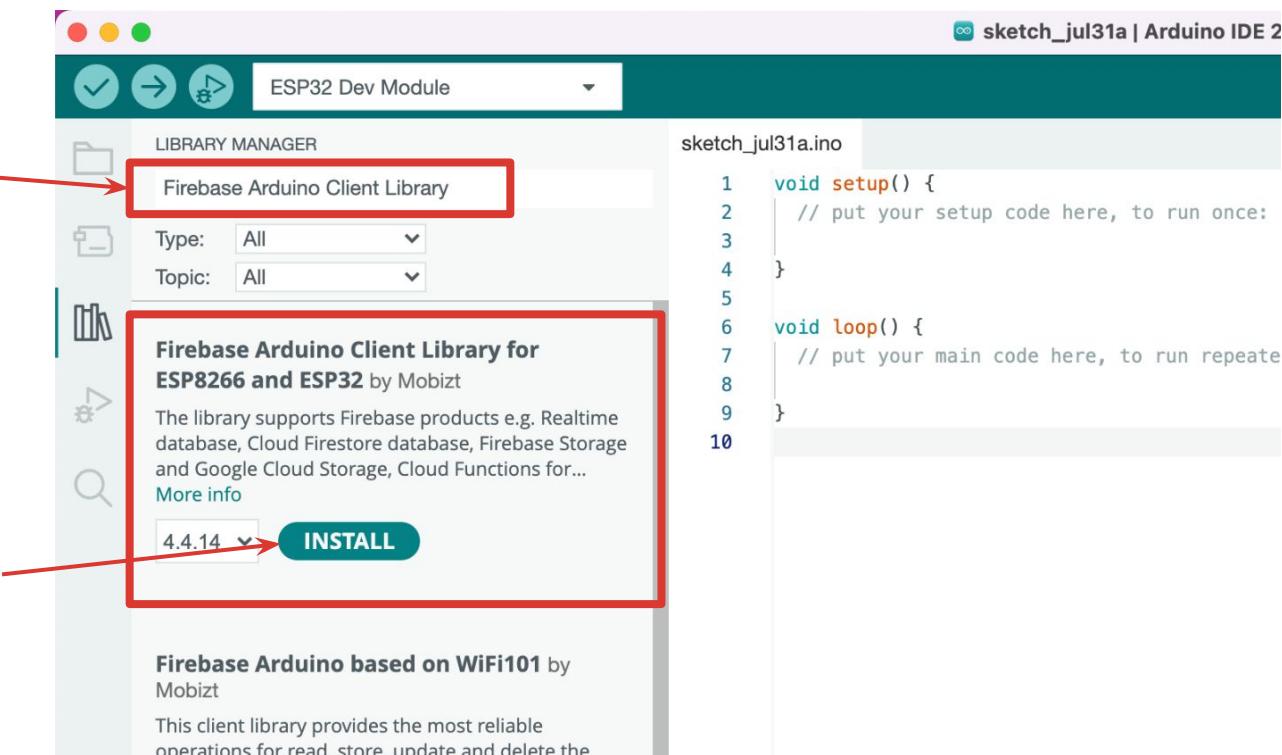
**macOS** Intel, 10.15: "Catalina" or newer, 64 bits  
**macOS** Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

Step 2: Open Arduino IDE. Click on the “**Libraries**” icon.

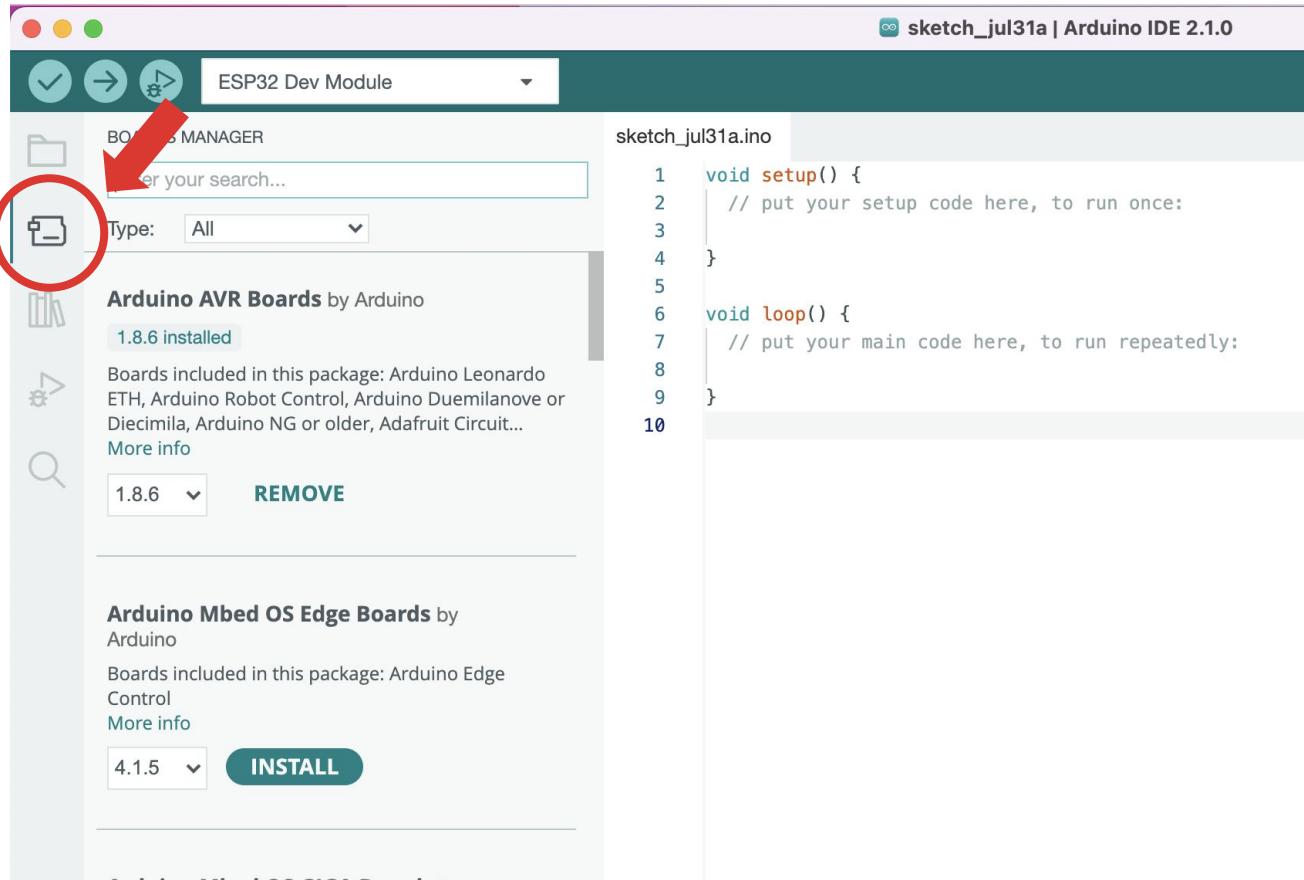


**Step 3:** In the search bar, type in: “**Firebase Arduino Client Library**”



**Step 4: Install** “**Firebase Arduino Client Library for ESP8266 and ESP32**” by **Mobitz**

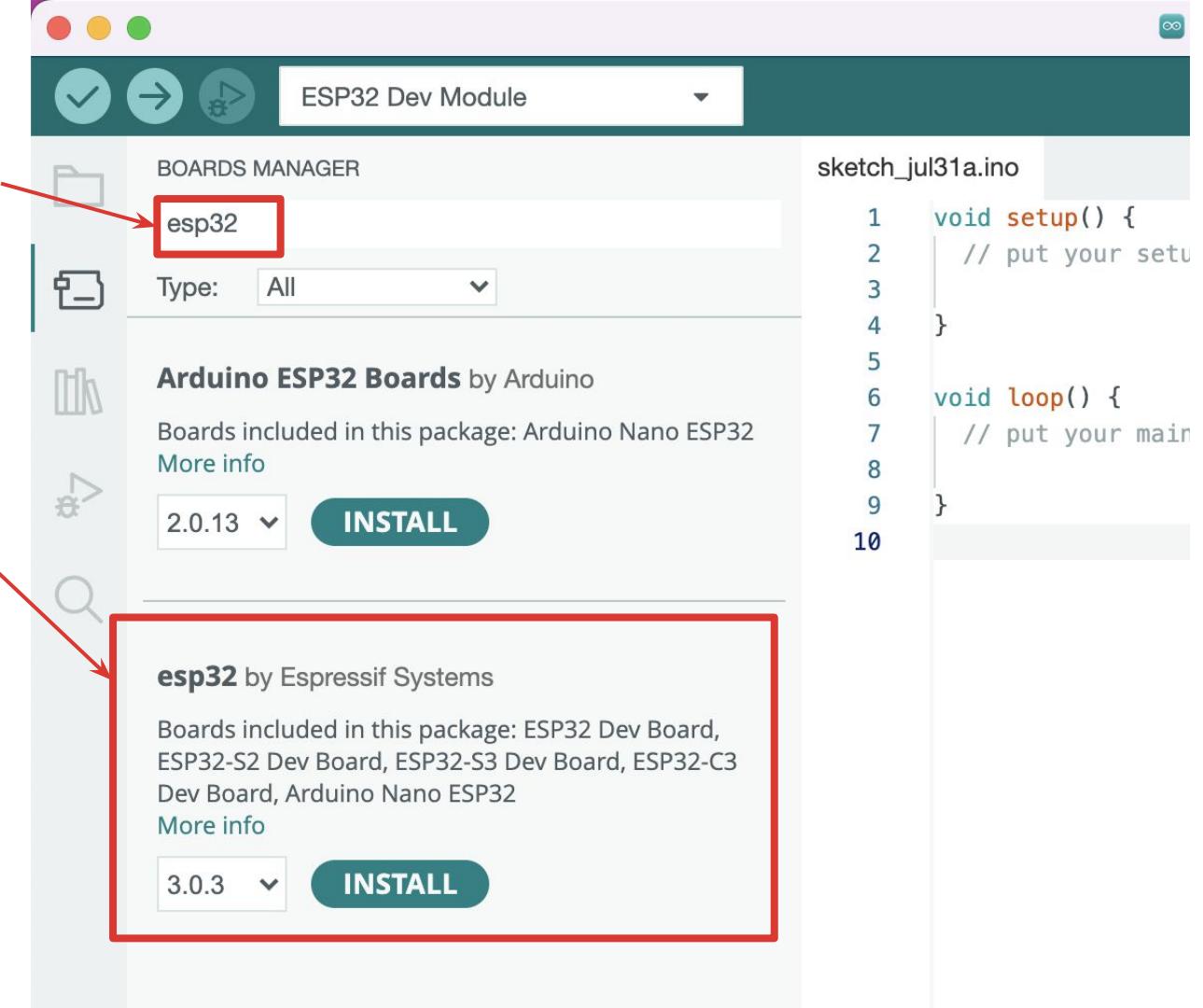
## Step 5: Next, click on the Boards Manager icon.



**Step 6:** In the search bar, search “**esp32**”

**Step 7: Install “esp32” library by Espressif Systems Version 3.0.7.** This could take some time.

Now, you finished installing all required libraries and packages.



**Step 8:** Head to our GitHub repository:

<https://github.com/juliadaser/Siggraph-Workshop.git> and go into the “**Code.ino**” file.

The screenshot shows a GitHub repository page for "NYCResistor\_LoveMessengers". The repository is public and has 1 branch and 0 tags. The main branch is selected. There are 13 commits from user "juliadaser". The commits listed are:

- Add files via upload (4039df5 · now)
- 3D Models (Add files via upload · 3 days ago)
- Media (Add files via upload · 9 hours ago)
- Slides (Add files via upload · now)
- README.md (Update README.md · 10 minutes ago)
- code.ino (Add files via upload · 11 minutes ago)

A red box highlights the "code.ino" commit, and a red arrow points to it from the bottom left.

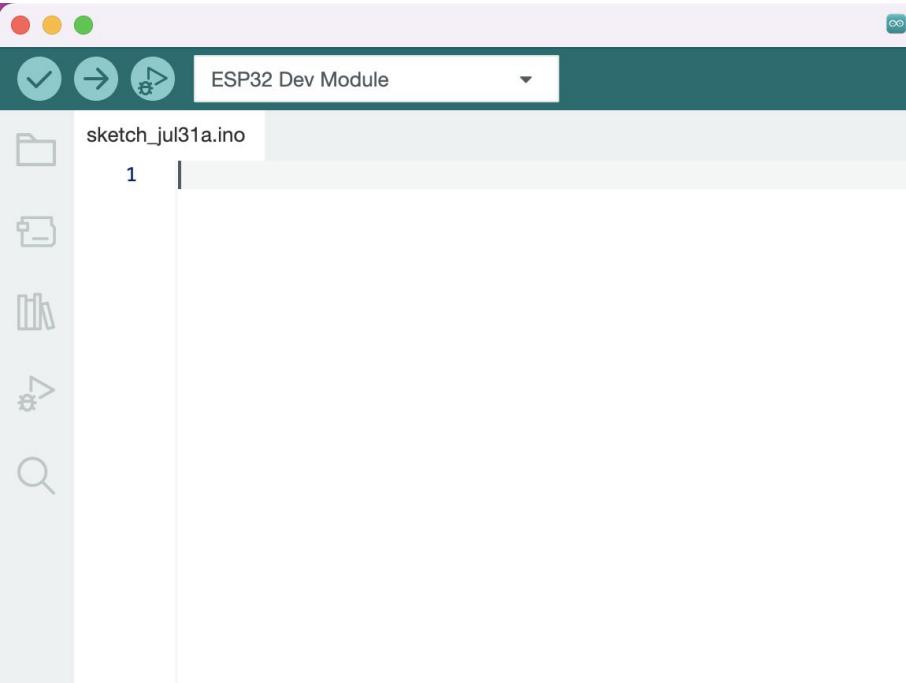
## Step 10: Hit this button to copy all the code.

The screenshot shows a code editor interface with a dark theme. At the top, there are tabs for "Code" (which is selected) and "Blame". Below the tabs, it displays "109 lines (93 loc) · 2.65 KB". The main area contains the following C++ code:

```
1 // Importing libraries
2 #include <WiFi.h>
3 #include <Firebase_ESP_Client.h>
4 #include "addons/TokenHelper.h"
5 #include "addons/RTDBHelper.h"
6
7 // Wi-Fi credentials
8 #define WIFI_SSID "NYCR24"
9 #define WIFI_PASSWORD "clubmate"
10
11 // Firebase credentials
12 #define API_KEY "insert API Key here"
13 #define DATABASE_URL "insert API Url here"
14
15 // LED & button connections to the ESP
16 const int ledPin = 4;
17 const int buttonPin = 14;
18 const int buttonLedPin = 25;
```

A red arrow points to the copy icon in the toolbar, which is circled in red. The toolbar also includes icons for Raw, download, edit, and refresh.

**Step 11:** Delete all the code that is currently in your Arduino IDE.



**Step 12:** Paste all the code you have copied from the GitHub repository into here.

A screenshot of the Arduino IDE showing a sketch named "nycresistor.ino". The code is as follows:

```
1 // Importing libraries
2 #include <WiFi.h>
3 #include <Firebase_ESP_Client.h>
4
5 // We need this for the firebase library to work
6 #include "addons/TokenHelper.h"
7 #include "addons/RTDBHelper.h"
8
9 // Wi-Fi credentials
10 #define WIFI_SSID "Verizon_7YV47L"
11 #define WIFI_PASSWORD "hotly-oak7-sock"
12
13 // Firebase credentials
14 #define API_KEY "AIzaSyCA1pmnFY3z8p134UPZ07wd7SWBVFzS9ek"
15 #define DATABASE_URL "https://newmicrocontroller-test-default.firebaseio.com/"
16
17 // Hardware components (LED & button)
18 const int ledPin = 4;
19 const int buttonPin = 14;
20 const int buttonLedPin = 25;
21
22 // Global variables
23 bool firebaseData = false;
24 int buttonState = 1;
```

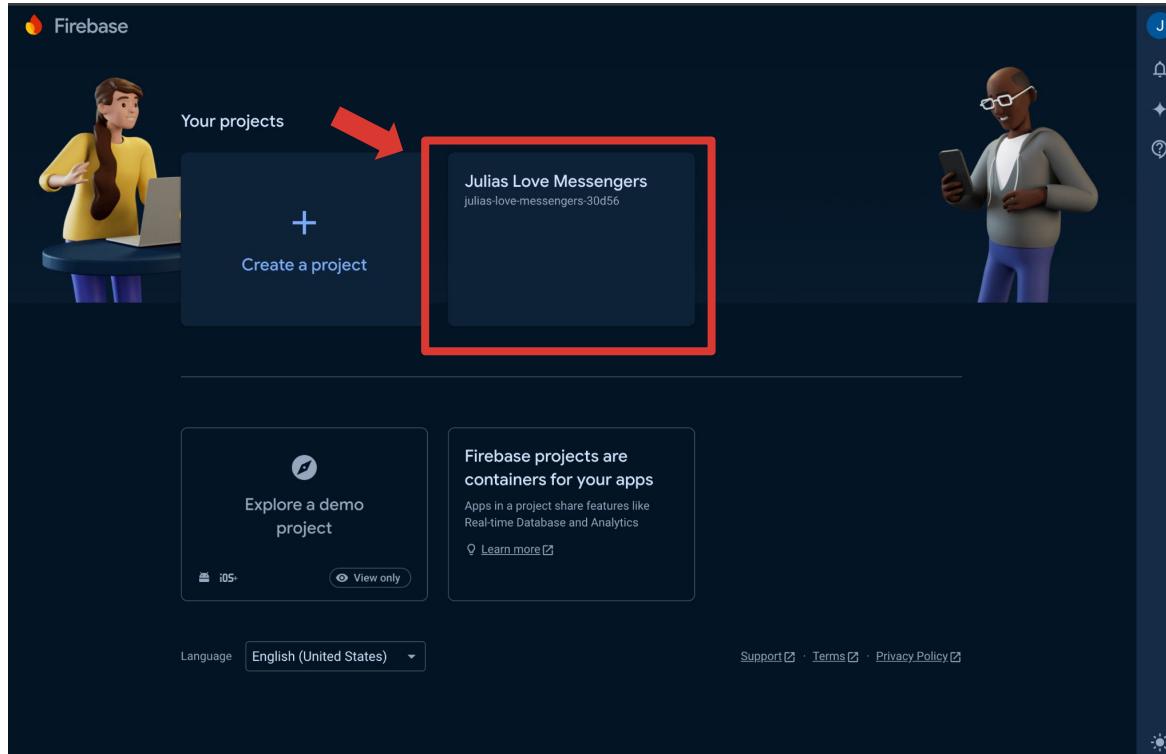
## Step 12: Insert your home WIFI Name and Password.

```
sketch_jul31a.ino
1 // Importing libraries
2 #include <WiFi.h>           // to Connect the ESP32 to Wifi
3 #include <Firebase_ESP_Client.h> // to interface the ESP32 with Firebase
4
5 // We need this for the firebase library to work
6 #include "addons	TokenNameHelper.h" //Provide the token generation process info.
7 #include "addons/RTDBHelper.h"   //Provide the RTDB payload printing info and other helper functions.
8
9 // Wi-Fi credentials
10 #define WIFI_SSID "insert Wifi Name here"
11 #define WIFI_PASSWORD "insert Wifi Password here"
12
13 // Firebase credentials
14 #define API_KEY "insert API Key here"
15 #define DATABASE_URL "insert Database URL here"
16
17 // Hardware components (LED & button)
18 const int ledPin = 4;
19 const int buttonPin = 14;
20
21 // Global variables
22 bool firebaseData = false;
23 int buttonState = 1;
24
25 // Define Firebase Data object and other necessary objects
26 FirebaseData fbdo;
27 FirebaseAuth auth;
28 FirebaseConfig config;
29 unsigned long lastFirebaseUpdate = 0;
30 int count = 0;
31 bool signupOK = false;
32
33 void setup() {
34     Serial.begin(115200);
35
36     pinMode(ledPin, OUTPUT);
37     pinMode(buttonPin, INPUT_PULLUP);
38
39     connectWiFi();
40 }
```

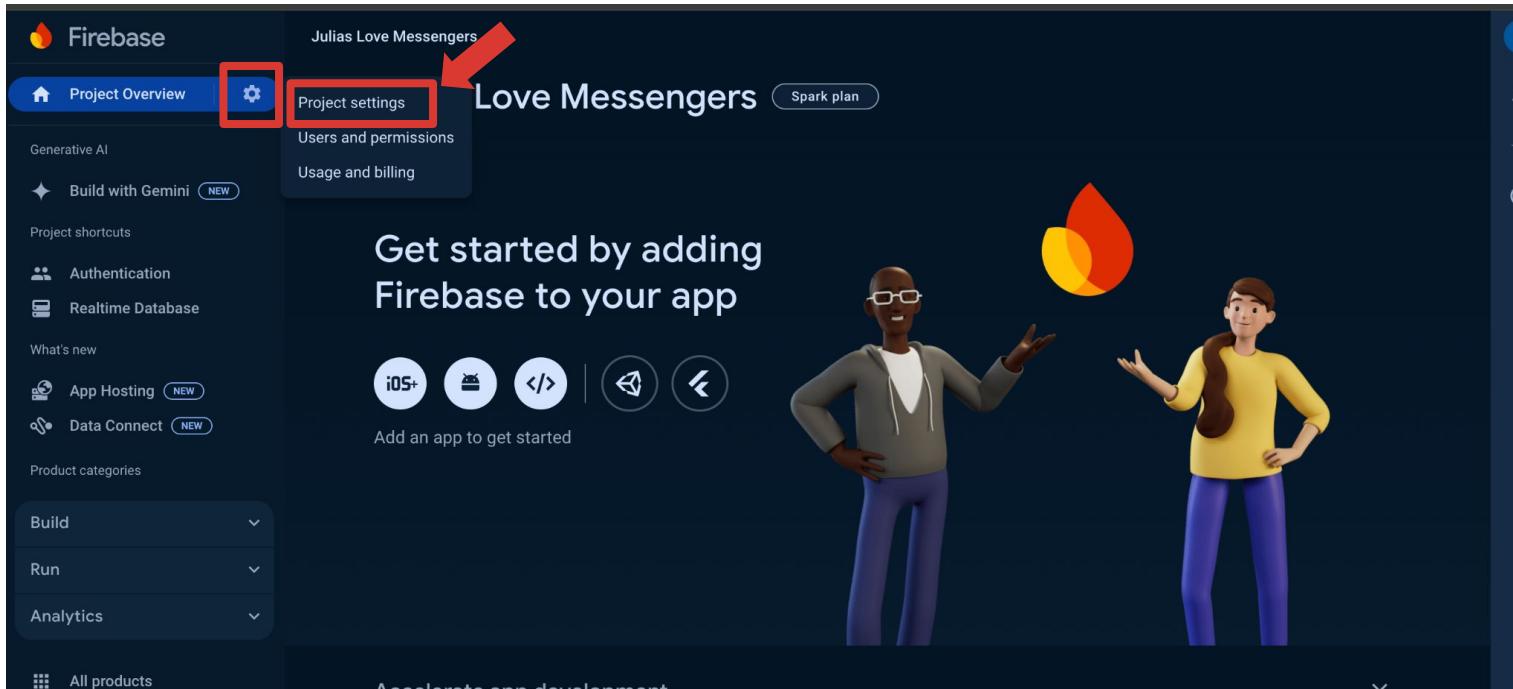
For example:

```
// Wi-Fi credentials
#define WIFI_SSID "Verizon-R500L6"
#define WIFI_PASSWORD "juliawifipassword"
```

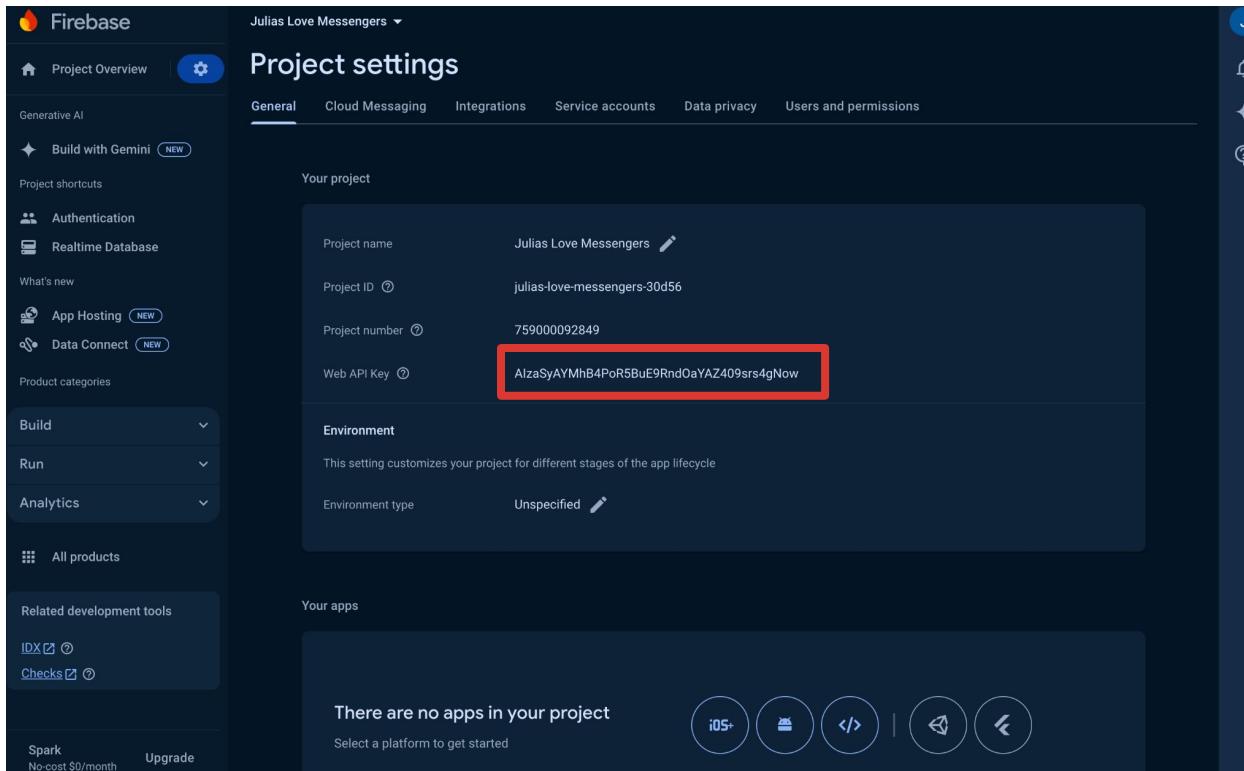
**Step 13:** Next, you insert the unique API Key of your Database in the code. To find it, head to [console.firebaseio.google.com](https://console.firebaseio.google.com), and select your Firebase Project. If you have not yet created a Firebase Project, head to our “CreateDatabase.pdf” file in the Slides folder of our Github.



**Step 14:** Inside your Firebase Console, select the little **gear icon**, and go to **“Project Settings”**



## Step 15: Inside your Project Settings, you will find your API Key. Copy the API Key.

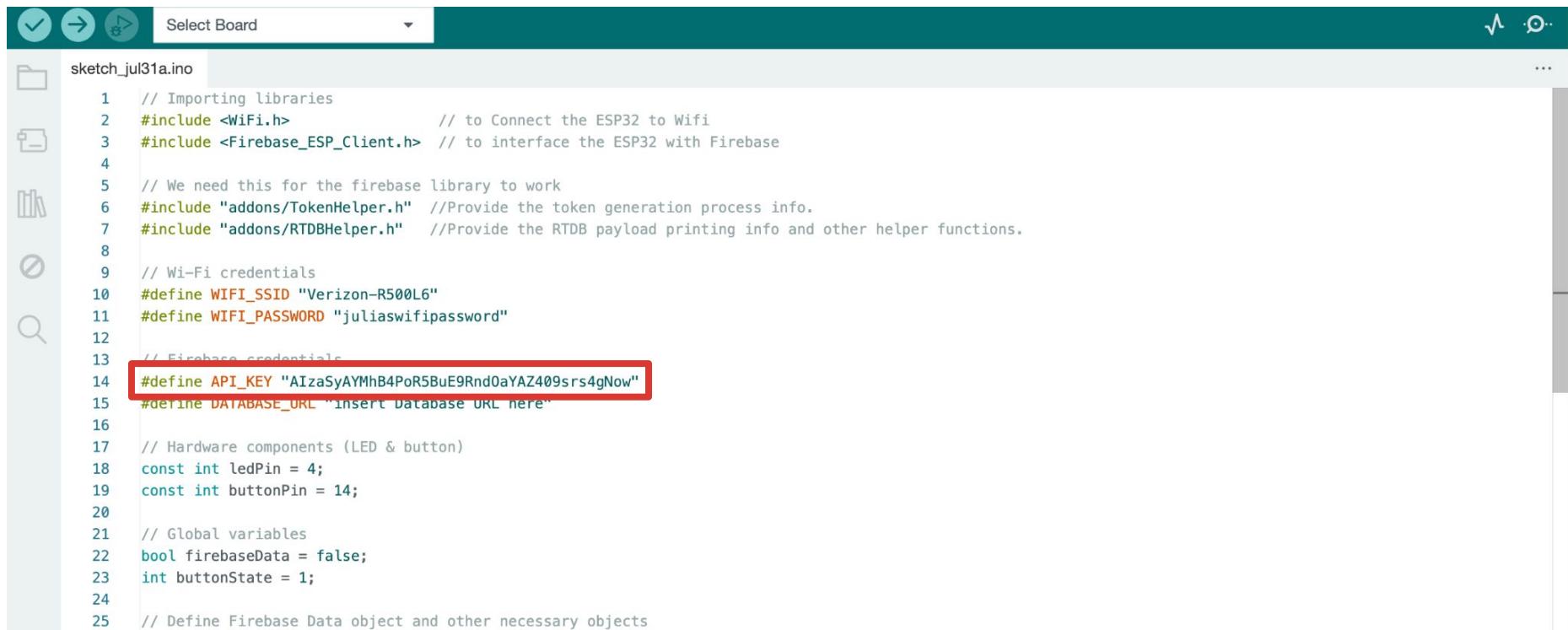


The screenshot shows the Firebase Project Overview page with the "Project settings" tab selected. The "General" tab is active. In the "Your project" section, there is a table with the following data:

Project name	Julias Love Messengers
Project ID	julias-love-messengers-30d56
Project number	759000092849
Web API Key	AlzaSyAYMhb4PoR5BuE9RndOaYAZ409srs4gNow

The "Web API Key" row is highlighted with a red box. Below the table, under the "Environment" section, it says: "This setting customizes your project for different stages of the app lifecycle". Under "Your apps", it says: "There are no apps in your project" and "Select a platform to get started".

## Step 16: Head back to the Arduino code, and insert the API Key inside the double quotes after “#define API\_KEY”



```
sketch_jul31a.ino
1 // Importing libraries
2 #include <WiFi.h>          // to Connect the ESP32 to Wifi
3 #include <Firebase_ESP_Client.h> // to interface the ESP32 with Firebase
4
5 // We need this for the firebase library to work
6 #include "addons	TokenNameHelper.h" //Provide the token generation process info.
7 #include "addons/RTDBHelper.h"   //Provide the RTDB payload printing info and other helper functions.
8
9 // Wi-Fi credentials
10 #define WIFI_SSID "Verizon-R500L6"
11 #define WIFI_PASSWORD "juliaswifipassword"
12
13 // Firebase credentials
14 #define API_KEY "AIzaSyAYMhB4PoR5BuE9Rnd0aYAZ409rs4gNow"
15 #define DATABASE_URL "insert database URL here"
16
17 // Hardware components (LED & button)
18 const int ledPin = 4;
19 const int buttonPin = 14;
20
21 // Global variables
22 bool firebaseData = false;
23 int buttonState = 1;
24
25 // Define Firebase Data object and other necessary objects
```

**Step 17:** Next, we will find your Database- URL. On Firebase, head to “**Build**” and select “**Realtime Database**”

The screenshot shows the Firebase Project settings interface for the project "Julias Love Messengers". The left sidebar has a red box around the "Build" category, which contains "App Check", "App Hosting (NEW)", "Authentication", "Data Connect (NEW)", "Extensions", "Firestore Database", "Functions", "Hosting", "Machine Learning", and "Realtime Database". A red arrow points from the text in Step 17 to the "Realtime Database" item in the sidebar. The main content area has tabs for General, Cloud Messaging, Integrations, Service accounts, Data privacy, and Users and permissions. Under "General", there are sections for "Your project" (Project name: Julias Love Messengers, Project ID: julias-love-messengers-30d56, Project number: 759000092849, Web API Key: AlzaSyAYMhB4PoR5BuE9RnD0aYAZ409srs4gNow) and "Environment" (Environment type: Unspecified). The "Your apps" section states "There are no apps in your project" and "Select a platform to get started".

## Step 18: Copy your Database URL.

The screenshot shows the Firebase Realtime Database console for the project "Julia's Love Messengers". The left sidebar lists various Firebase products: Generative AI, Build with Gemini, Project shortcuts, Authentication (selected), Realtime Database (highlighted with a blue border), What's new, App Hosting, Data Connect, Product categories, Build, Run, Analytics, and All products. Below these are Related development tools: IDEs (with a question mark icon) and Checks (with a question mark icon). At the bottom, there are links for Spark (No-cost \$0/month) and Upgrade. The main area is titled "Realtime Database" and includes tabs for Data, Rules, Backups, Usage, and Extensions. A banner at the top right says "Protect your Realtime Database resources from abuse, such as billing fraud or phishing" with a "Configure App Check" button and a close button. The Data tab is selected, showing the database URL: <https://julia's-love-messengers-30d56-default-rtbd.firebaseio.com>. Below the URL, the database structure is visible with nodes for user\_1 and user\_2.

Julia's Love Messengers ▾

## Realtime Database

Data    Rules    Backups    Usage    Extensions

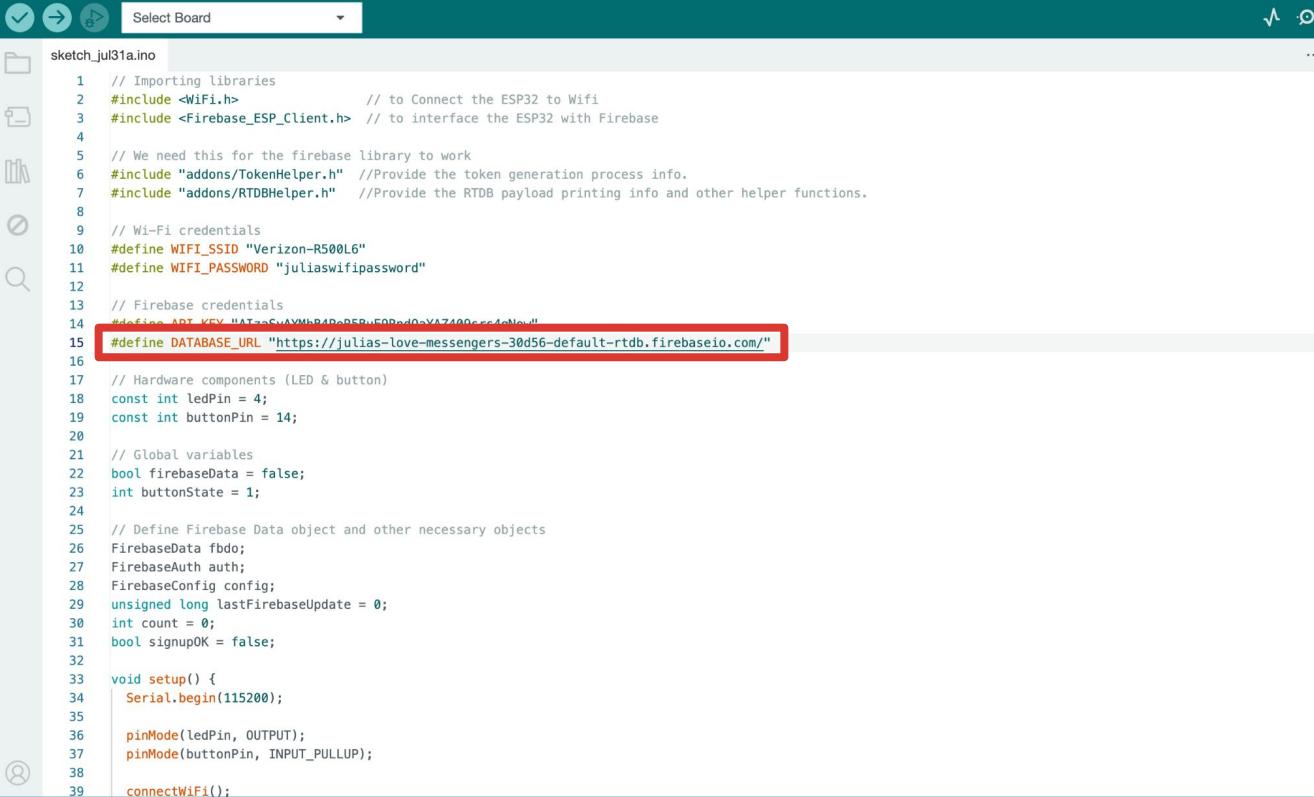
Protect your Realtime Database resources from abuse, such as billing fraud or phishing    Configure App Check    X

🔗 <https://julia's-love-messengers-30d56-default-rtbd.firebaseio.com>

```
https://julia's-love-messengers-30d56-default-rtbd.firebaseio.com/
  └── user_1
      └── 0
  └── user_2
      └── 0
```

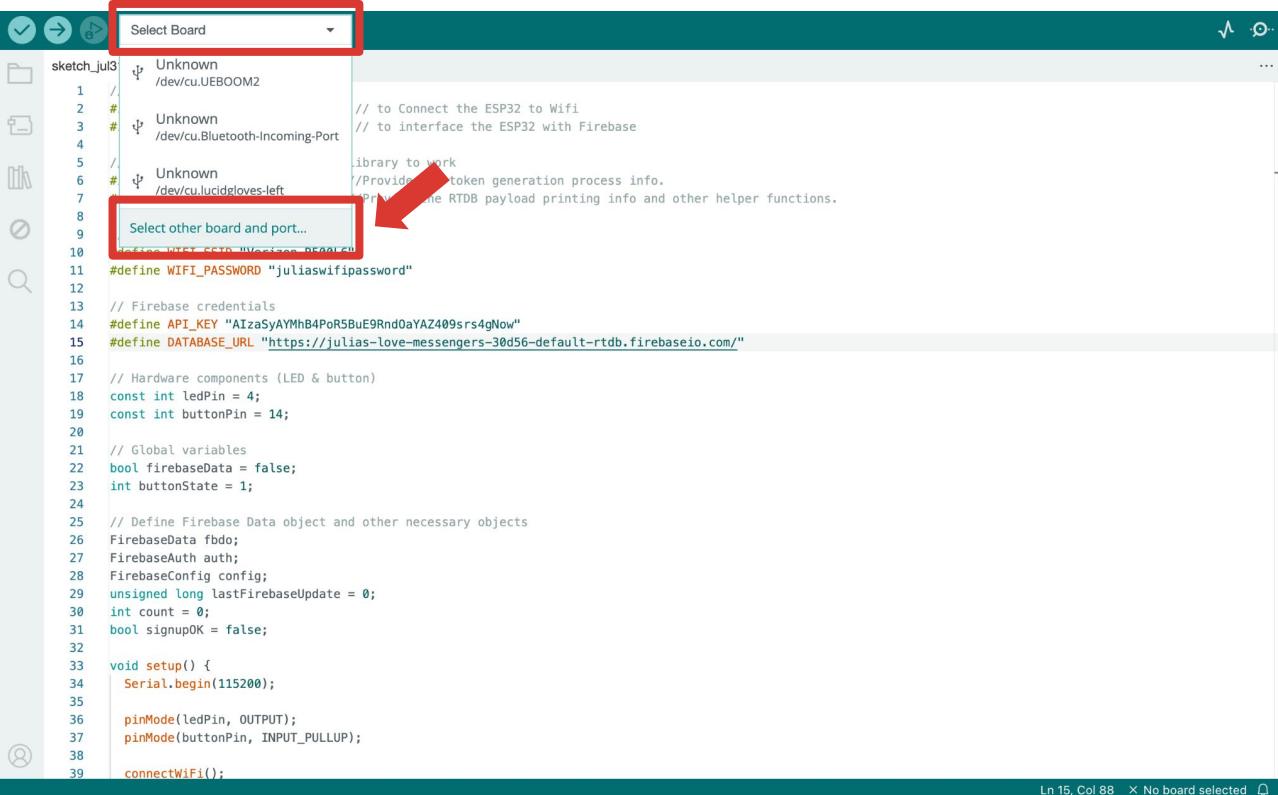
Database location: United States (us-central1)

## Step 19: Paste the Database URL into the Arduino Code into the double quotes after "#define DATABASE\_URL"



```
sketch_jul31a.ino
1 // Importing libraries
2 #include <WiFi.h>           // to Connect the ESP32 to Wifi
3 #include <firebase_ESP_Client.h> // to interface the ESP32 with Firebase
4
5 // We need this for the firebase library to work
6 #include "addons/TokenHelper.h" //Provide the token generation process info.
7 #include "addons/RTDBHelper.h"  //Provide the RTDB payload printing info and other helper functions.
8
9 // Wi-Fi credentials
10 #define WIFI_SSID "Verizon-R500L6"
11 #define WIFI_PASSWORD "juliaswifipassword"
12
13 // Firebase credentials
14 #define ADT_KEY "AIzaSyAVMbR4DnDFBvF0Dad0aVAZ400cc4aMowd"
15 #define DATABASE_URL "https://julias-love-messengers-30d56-default.firebaseio.com/"
16
17 // Hardware components (LED & button)
18 const int ledPin = 4;
19 const int buttonPin = 14;
20
21 // Global variables
22 bool firebaseData = false;
23 int buttonState = 1;
24
25 // Define Firebase Data object and other necessary objects
26 FirebaseDatabase fbdo;
27 FirebaseAuth auth;
28 FirebaseConfig config;
29 unsigned long lastFirebaseUpdate = 0;
30 int count = 0;
31 bool signupOK = false;
32
33 void setup() {
34   Serial.begin(115200);
35
36   pinMode(ledPin, OUTPUT);
37   pinMode(buttonPin, INPUT_PULLUP);
38
39   connectWiFi();
```

**Step 20:** The code is now ready to be uploaded to the first Love Messenger.  
Go to “**Select Board**”, and go to “**Select other board and port...**”



The screenshot shows the Arduino IDE interface. At the top, there's a toolbar with icons for file operations like Open, Save, and Print. Below the toolbar is a menu bar with 'File', 'Edit', 'Tools', 'Sketch', 'Help', and a 'Board' dropdown menu. The 'Board' menu is currently open, displaying a list of available boards and ports. The first item in the list is 'Unknown /dev/cu.UEBOOM2'. Below it are several other entries, including 'Unknown /dev/cu.Blueooth-Incoming-Port' and 'Unknown /dev/cu.lucidgloves-left'. At the bottom of the list, there's an option 'Select other board and port...'. This option is highlighted with a red box and has a red arrow pointing towards it from the left side of the image.

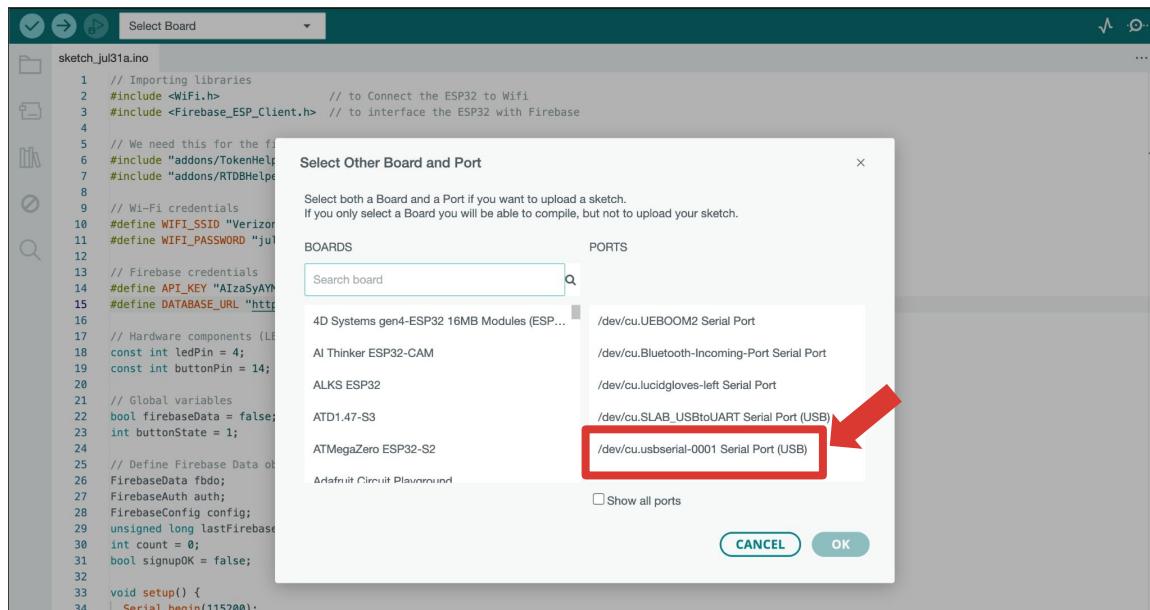
```
sketch_jul3 // to Connect the ESP32 to Wifi
1 // to interface the ESP32 with Firebase
2 # Unknown
3 # Unknown
4 //library to work
5 //Provide token generation process info.
6 # Unknown
7 //Print the RTDB payload printing info and other helper functions.
8 // Select WiFi and FirebaseDatabase DEFAULT
9 //Firebase credentials
10 #define WIFI_PASSWORD "juliaswifipassword"
11
12 // Hardware components (LED & button)
13 const int ledPin = 4;
14 const int buttonPin = 14;
15
16 // Global variables
17 bool firebaseData = false;
18 int buttonState = 1;
19
20 // Define Firebase Data object and other necessary objects
21 FirebaseData fbd;
22 FirebaseAuth auth;
23 FirebaseConfig config;
24 unsigned long lastFirebaseUpdate = 0;
25 int count = 0;
26 bool signupOK = false;
27
28 void setup() {
29   Serial.begin(115200);
30
31   pinMode(ledPin, OUTPUT);
32   pinMode(buttonPin, INPUT_PULLUP);
33
34   connectWiFi();
35
36
37
38
39 }
```

Ln 15, Col 88 X No board selected

## Step 21: Plug in your first Love Messenger to your Computer using a data-transfer USB cable.

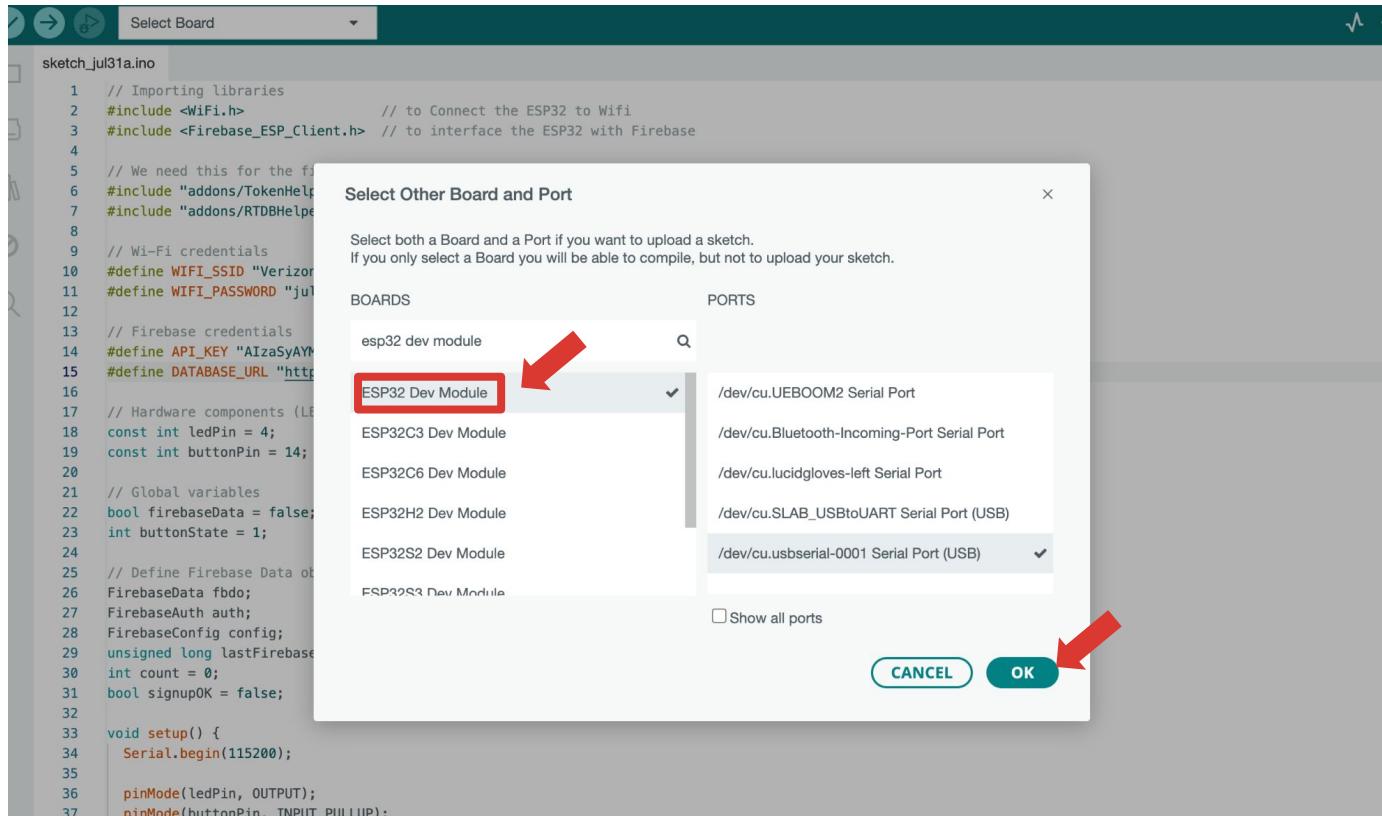
Once the Love Messenger is plugged in, a new port should appear in the “Ports” section. Depending on if you are using a Mac or PC, the port names can vary (“COM” for PC, and “usbserial” for Mac).

Select the new port.

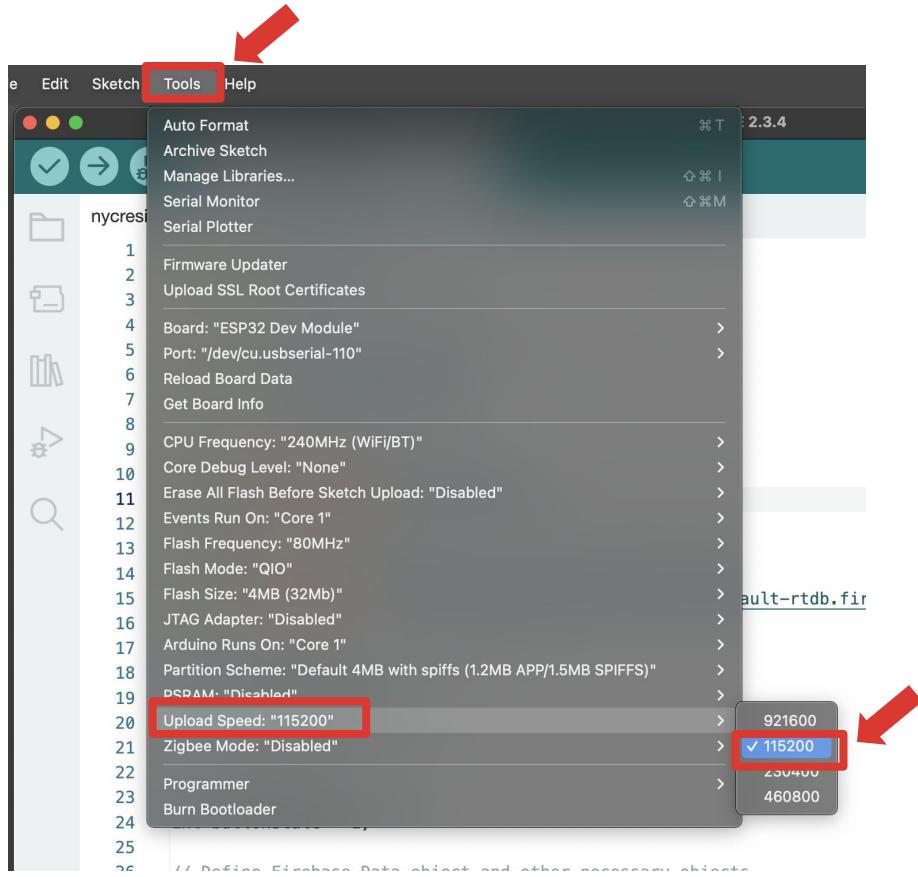


*\*If your port does not show up, you might be using a charge-only USB wire that doesn't transmit data. You need to use a USB wire that also can do data-transfer. We have the ones we used in the workshop linked on Github.*

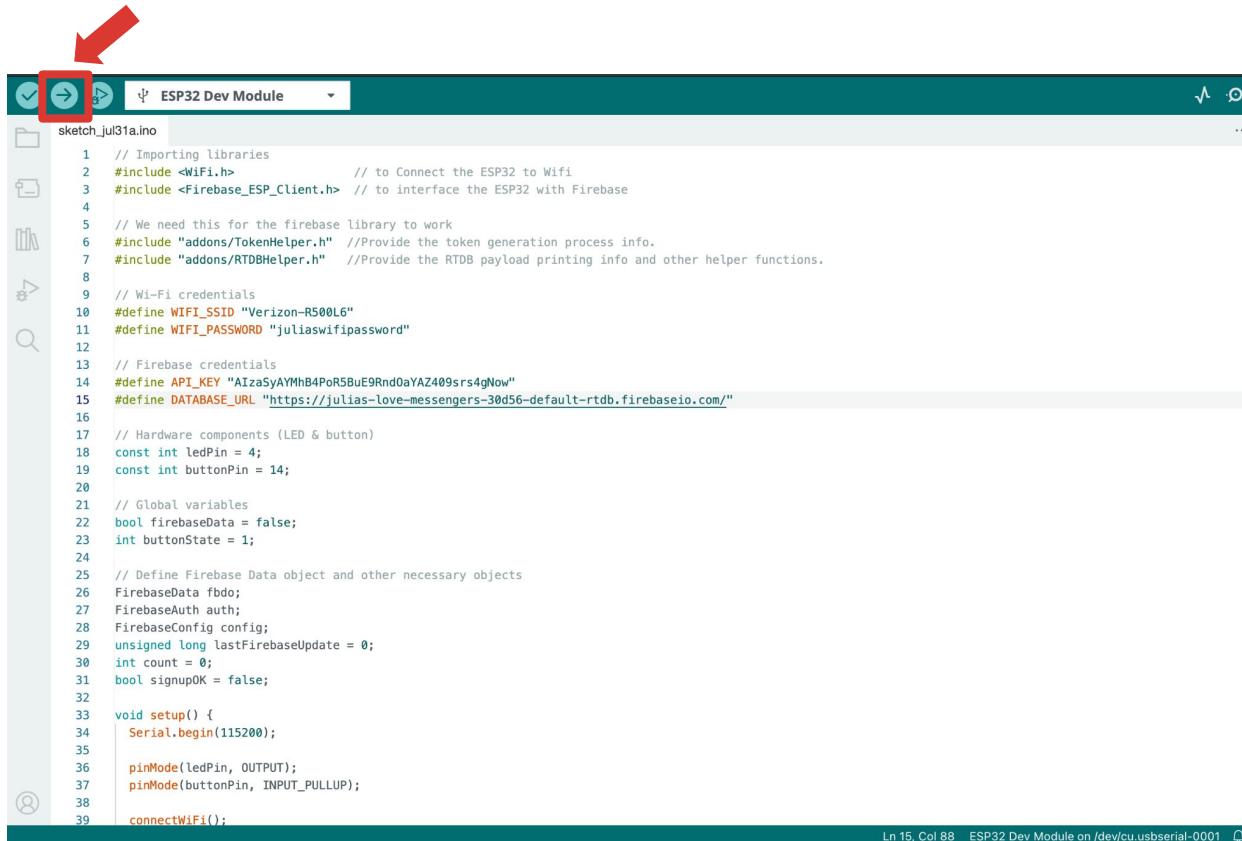
**Step 22:** In the “**Boards**” section, search for “**ESP32 Dev Module**”, and select it. Then Press OK



## Step 23: Change the upload speed: Tools - Upload Speed - 115200



**Step 24:** The code is now ready to be uploaded. Hit the little arrow icon. This will compile the code (takes a few minutes), and will upload it to your Love Messenger.

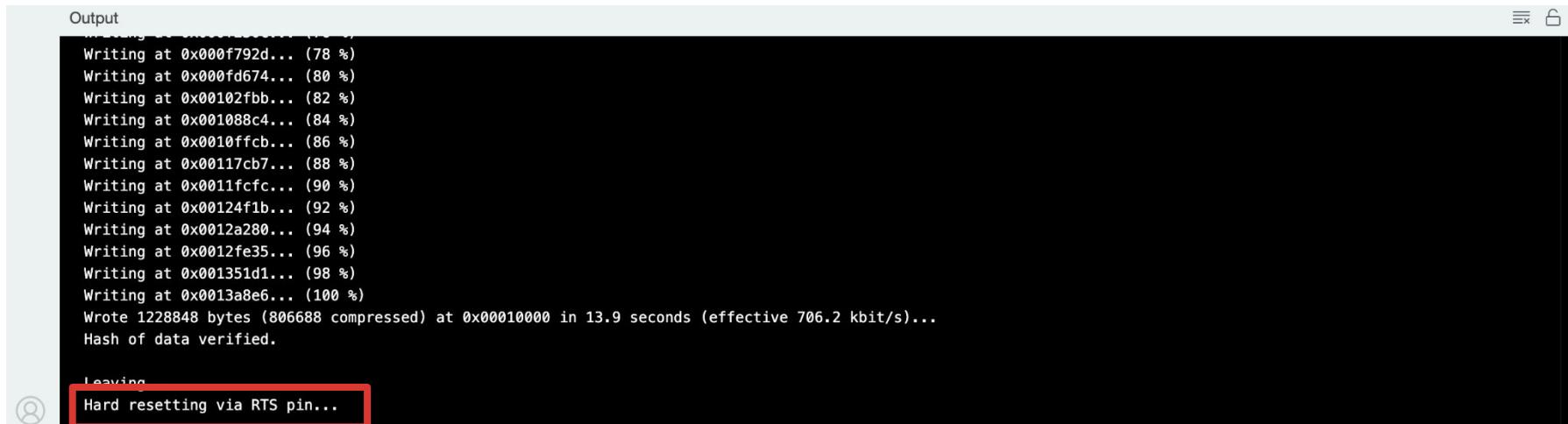


```
sketch_jul31a.ino
1 // Importing libraries
2 #include <WiFi.h>           // to Connect the ESP32 to Wifi
3 #include <Firebase_ESP_Client.h> // to interface the ESP32 with Firebase
4
5 // We need this for the firebase library to work
6 #include "addons/TokenHelper.h" //Provide the token generation process info.
7 #include "addons/RTDBHelper.h" //Provide the RTDB payload printing info and other helper functions.
8
9 // Wi-Fi credentials
10 #define WIFI_SSID "Verizon-R500L6"
11 #define WIFI_PASSWORD "juliaswifipassword"
12
13 // Firebase credentials
14 #define API_KEY "AIzaSyAYMhB4PoR5BuE9RnDoaYAZ409srs4gNow"
15 #define DATABASE_URL "https://julias-love-messengers-30d56-default-rtdb.firebaseio.com/"
16
17 // Hardware components (LED & button)
18 const int ledPin = 4;
19 const int buttonPin = 14;
20
21 // Global variables
22 bool firebaseData = false;
23 int buttonState = 1;
24
25 // Define Firebase Data object and other necessary objects
26 FirebaseData fbdo;
27 FirebaseAuth auth;
28 FirebaseConfig config;
29 unsigned long lastFirebaseUpdate = 0;
30 int count = 0;
31 bool signupOK = false;
32
33 void setup() {
34   Serial.begin(115200);
35
36   pinMode(ledPin, OUTPUT);
37   pinMode(buttonPin, INPUT_PULLUP);
38
39   connectWiFi();
```

Ln 15, Col 88 ESP32 Dev Module on /dev/cu.usbserial-0001

**Step 25:** The Sketch will first compile and then upload.

The code is uploaded once the “**Output**” window reads “**Hard resetting via RTS pin...**”

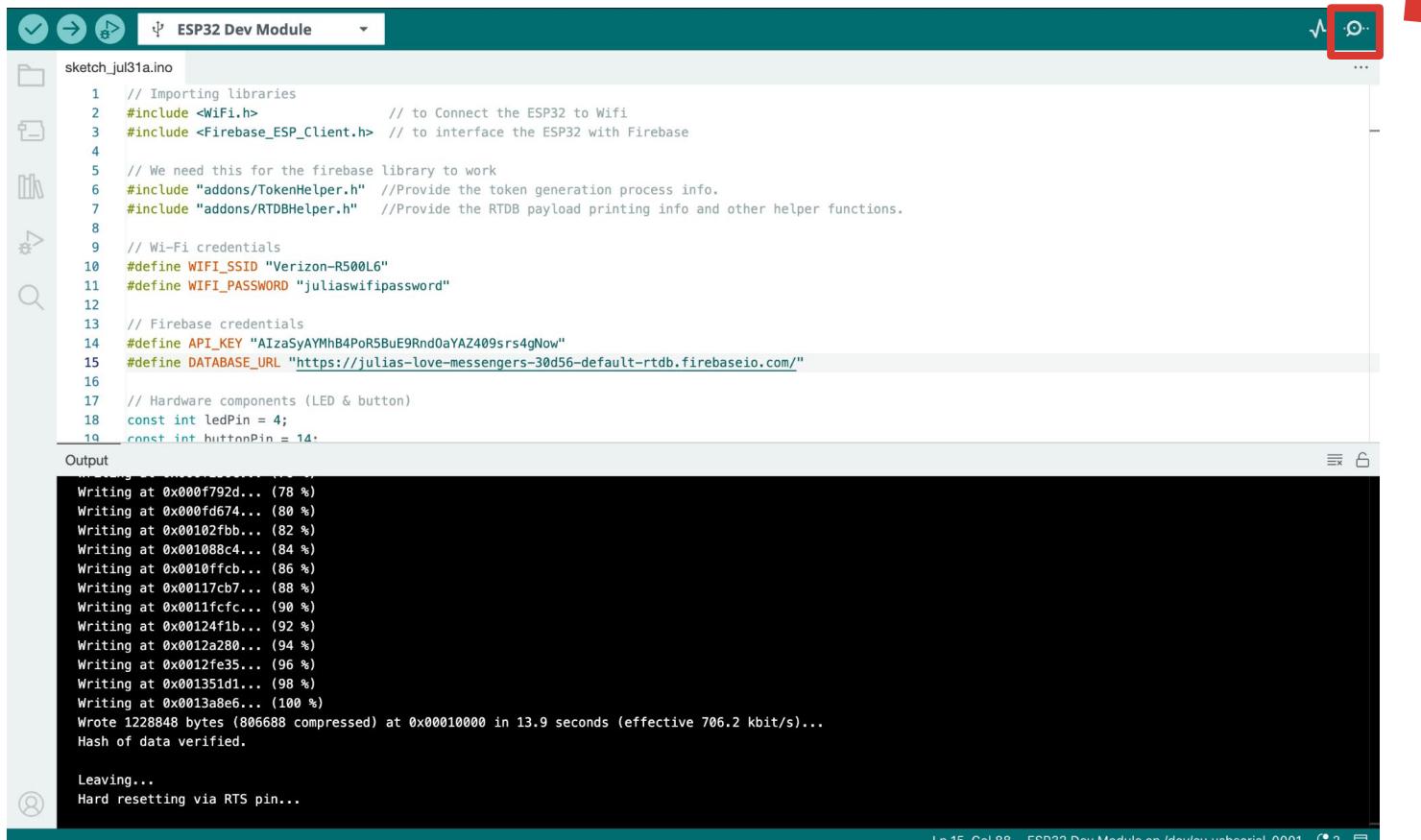


Output

```
Writing at 0x000f792d... (78 %)
Writing at 0x000fd674... (80 %)
Writing at 0x00102fbb... (82 %)
Writing at 0x001088c4... (84 %)
Writing at 0x0010ffcb... (86 %)
Writing at 0x00117cb7... (88 %)
Writing at 0x0011fcfc... (90 %)
Writing at 0x00124f1b... (92 %)
Writing at 0x0012a280... (94 %)
Writing at 0x0012fe35... (96 %)
Writing at 0x001351d1... (98 %)
Writing at 0x0013a8e6... (100 %)
Wrote 1228848 bytes (806688 compressed) at 0x00010000 in 13.9 seconds (effective 706.2 kbit/s)...
Hash of data verified.

Leaving
Hard resetting via RTS pin...
```

## Step 26: To check if our code works, open the Serial Monitor.



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** ESP32 Dev Module
- File Explorer:** Shows the file sketch\_jul31a.ino
- Code Editor:** Displays the C++ code for the ESP32. The code includes imports for WiFi and Firebase libraries, defines WiFi\_SSID and WiFi\_PASSWORD, and defines API\_KEY and DATABASE\_URL. It also defines hardware components like ledPin and buttonPin.
- Serial Monitor:** A large black text area showing the output of the code execution. The output shows the progress of writing data to memory (Writing at 0x000f792d...), the completion of writing (Wrote 1228848 bytes (806688 compressed) at 0x00010000 in 13.9 seconds (effective 706.2 kbit/s)...), a hash verification message (Hash of data verified.), and the final message (Leaving... Hard resetting via RTS pin...).
- Top Right Icons:** Includes icons for upload, refresh, and serial monitor. The serial monitor icon is highlighted with a red box and an arrow pointing to it from the left.
- Bottom Status Bar:** Ln 15, Col 88 ESP32 Dev Module on /dev/cu.usbserial-0001

## Step 27: In the serial Monitor, you should see these messages;

The Love Messenger will first connect to Wifi. (If it doesn't connect, double check your wifi strength, and if you correctly entered your name and password.)

The Wifi is connected

Once the Serial Monitor prints this, we can move on to the second Love Messenger. Everything is set up.

```
Output Serial Monitor X  
Message (Enter to send message to 'ESP32 Dev Module' on '/dev/cu.usbserial-0001')  
-----  
18:10:33.172 -> Connecting to Wifi...  
18:10:33.470 -> Connecting to Wifi...  
  
18:10:33.774 -> Connected with IP: 172.22.52.167  
18:10:34.894 -> Firebase successful  
18:10:35.518 -> Data UPLOAD successful, Data DOWNLOAD failed  
18:10:40.008 -> Data UPLOAD successful, Data DOWNLOAD failed  
18:10:43.701 -> Data UPLOAD successful, Data DOWNLOAD failed  
18:10:47.495 -> Data UPLOAD successful, Data DOWNLOAD failed  
18:10:50.266 -> Data UPLOAD successful, Data DOWNLOAD failed  
18:10:53.112 -> Data UPLOAD successful, Data DOWNLOAD failed  
18:10:56.809 -> Data UPLOAD successful, Data DOWNLOAD failed  
18:11:00.107 -> Data UPLOAD successful, Data DOWNLOAD failed
```

**Step 28:** Now, unplug your first Love Messenger, and plug in your second Love Messenger.

**Step 29:** In your code, in the **uploadData** function, change “/user\_1” to “/user\_2”

```
88 ~ void uploadData(int buttonstate) {  
89 ~ | if (Firebase.RTDB.setInt(&fbdo, "/user_1", buttonstate)) {  
90 | | Serial.printf("Data UPLOAD successful, ");  
91 ~ | } else {  
92 | | Serial.println("Data UPLOAD failed, ");  
93 }  
94 }
```

Change to

```
88 void uploadData(int buttonstate) {  
89 | if (Firebase.RTDB.setInt(&fbdo, "/user_2", buttonstate)) {  
90 | | Serial.printf("Data UPLOAD successful, ");  
91 } else {  
92 | | Serial.println("Data UPLOAD failed, ");  
93 }  
94 }
```

**Step 30:** Similarly, in your code, in the **download data** function, change “/user\_2” to “/user\_1”

```
96 void downloadData() {  
97     if (Firebase.RTDB.getInt(&fbdo, "/user_2")) {  
98         firebaseData = fbdo.intData();  
99         Serial.println("Data DOWNLOAD successful");  
100    } else {  
101        Serial.println("Data DOWNLOAD failed");  
102    }  
103 }
```

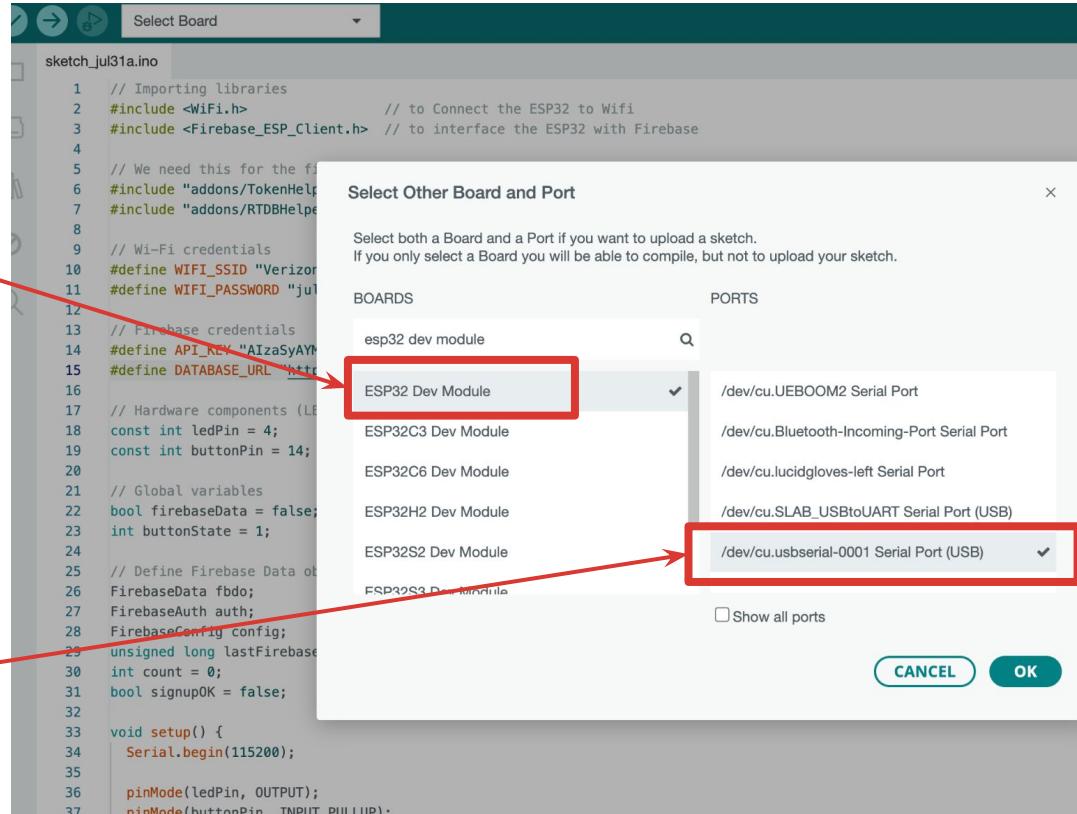
Change to

```
96 void downloadData() {  
97     if (Firebase.RTDB.getInt(&fbdo, "/user_1")) {  
98         firebaseData = fbdo.intData();  
99         Serial.println("Data DOWNLOAD successful");  
100    } else {  
101        Serial.println("Data DOWNLOAD failed");  
102    }  
103 }
```

## Step 31:

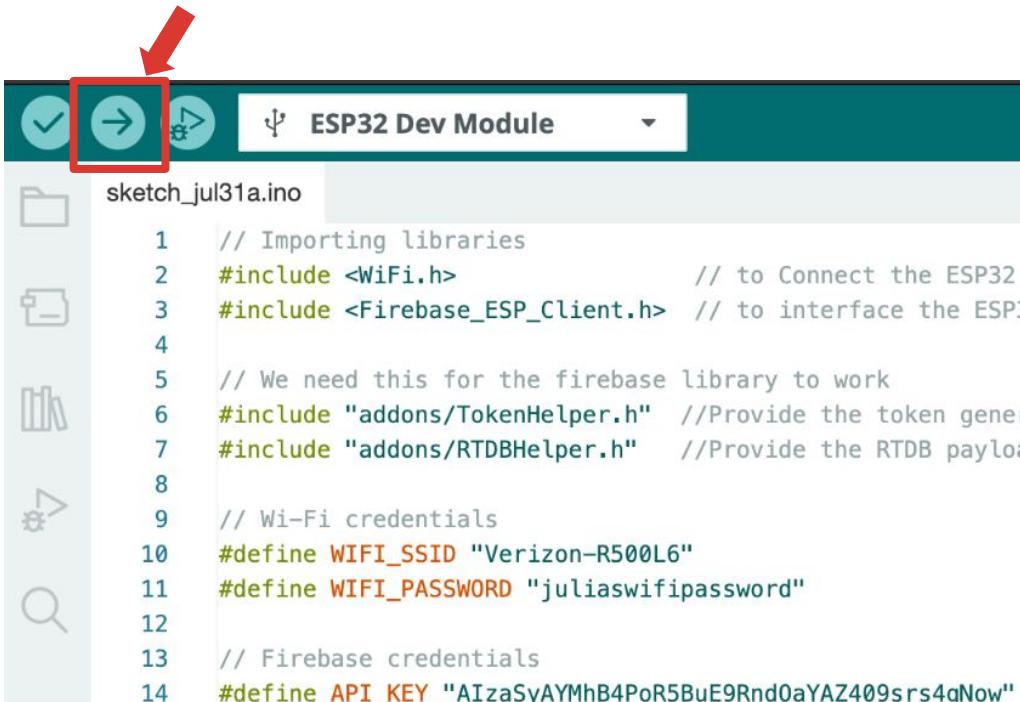
Once again, under “Select Board”, you should have **“ESP32 Dev Module”** selected under BOARDS.

Check that your second Love Messenger is properly connected to a port.



\*If your Love Messenger is not showing up in the PORTS, try disconnecting the wire from your Love Messenger and connecting it again.

## Step 32: Now, upload the code to the second Love Messenger!



```
sketch_jul31a.ino
1 // Importing libraries
2 #include <WiFi.h> // to Connect the ESP32
3 #include <Firebase_ESP_Client.h> // to interface the ESP32
4
5 // We need this for the firebase library to work
6 #include "addons	TokenNameHelper.h" //Provide the token generation
7 #include "addons/RTDBHelper.h" //Provide the RTDB payload
8
9 // Wi-Fi credentials
10 #define WIFI_SSID "Verizon-R500L6"
11 #define WIFI_PASSWORD "juliaswifipassword"
12
13 // Firebase credentials
14 #define API_KEY "AIzaSyAYMhB4PoR5BuE9Rnd0aYAZ409srs4gNow"
```

Once again, be patient! Your code might take awhile to compile and upload.

## Step 33: Once it has successfully uploaded, check the serial monitor again.

Once again, it needs to go through these steps in order to successfully connect to Firebase

**Step 34: Once you see “Data UPLOAD successful, Data DOWNLOAD successful”, this means that your second Love Messenger has successfully connected to Firebase, together with your first Love Messenger!**

```
18:20:03.343 -> Connecting to Wifi...
18:20:03.638 -> Connected with IP: 172.22.52.101
18:20:04.963 -> Firebase successful
18:20:05.490 -> Data UPLOAD successful, Data DOWNLOAD successful
18:20:07.597 -> Data UPLOAD successful, Data DOWNLOAD successful
18:20:09.801 -> Data UPLOAD successful, Data DOWNLOAD successful
18:20:11.912 -> Data UPLOAD successful, Data DOWNLOAD successful
18:20:14.156 -> Data UPLOAD successful, Data DOWNLOAD successful
18:20:16.420 -> Data UPLOAD successful, Data DOWNLOAD successful
18:20:18.511 -> Data UPLOAD successful, Data DOWNLOAD successful
18:20:20.758 -> Data UPLOAD successful, Data DOWNLOAD successful
```

**Step 35:** Now, plug both Love Messengers into your laptop/ any power supply.

Wait a few moments for both Love Messengers to connect to Wifi.

If everything is successful, you should see both Love Messengers working!



### Tips:

1. Be patient: press the button for a few seconds until you see your Love Messenger light up
2. Unplugging and replugging the Love Messenger can help too!

**Step 36:** While the Love Messengers are working, you should be able to see Firebase updating in real time.

Default Firebase states

```
🔗 https://newmicrocontroller-test-default-rtdb.firebaseio.com/  
  
https://newmicrocontroller-test-default-rtdb.firebaseio.com/  
└── user_1: 0  
└── user_2: 0
```

When user\_1's button is successfully pressed- both Love Messengers should light up!

```
🔗 https://newmicrocontroller-test-default-rtdb.firebaseio.com/  
  
https://newmicrocontroller-test-default-rtdb.firebaseio.com/  
└── user_1: 1  
└── user_2: 0
```

# Troubleshooting!

## 1. If only one Love Messenger is working...

Unplug both Love Messengers, and **plug only one** into your laptop again.

**Check the Serial Monitor:** Make sure that its connection to Firebase is successful.

Then, **plug in your second Love Messenger again**, and make sure it is also successfully connected to Wifi and Firebase.

```
Output Serial Monitor X

Message (Enter to send message to 'ESP32 Dev Module' on '/dev/cu.usbserial-0001')

-----
18:10:33.172 -> Connecting to Wifi...
18:10:33.470 -> Connecting to Wifi...
18:10:33.774 -> Connected with IP: 172.22.52.167
18:10:34.894 -> Firebase successful
18:10:35.518 -> Data UPLOAD successful, Data DOWNLOAD failed
18:10:40.008 -> Data UPLOAD successful, Data DOWNLOAD failed
18:10:43.701 -> Data UPLOAD successful, Data DOWNLOAD failed
18:10:47.495 -> Data UPLOAD successful, Data DOWNLOAD failed
```

# Troubleshooting!

## 2. If you are getting a “Token error” in your serial monitor:

Upload the code again to the respective Love Messenger.

## Troubleshooting!

**There are many other things that can go wrong with delicate code**

Reach out to us if you have any more questions:

Email: [yiqing.ng@gmail.com](mailto:yiqing.ng@gmail.com)

Instagram: [@julia.daser](https://www.instagram.com/@julia.daser)

We are more than happy to help! ❤️