A photograph showing two hands against a black background. The hand on the left holds a glowing orange-red heart-shaped object, which appears to be a 3D-printed model of a heart. The hand on the right holds a glowing green rectangular object with a central square cutout, also appearing to be a 3D-printed model. Both objects emit a bright light, suggesting they are illuminated from within.

Instructions

Uploading the Arduino Code

SIGGRAPH Experience Labs

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)

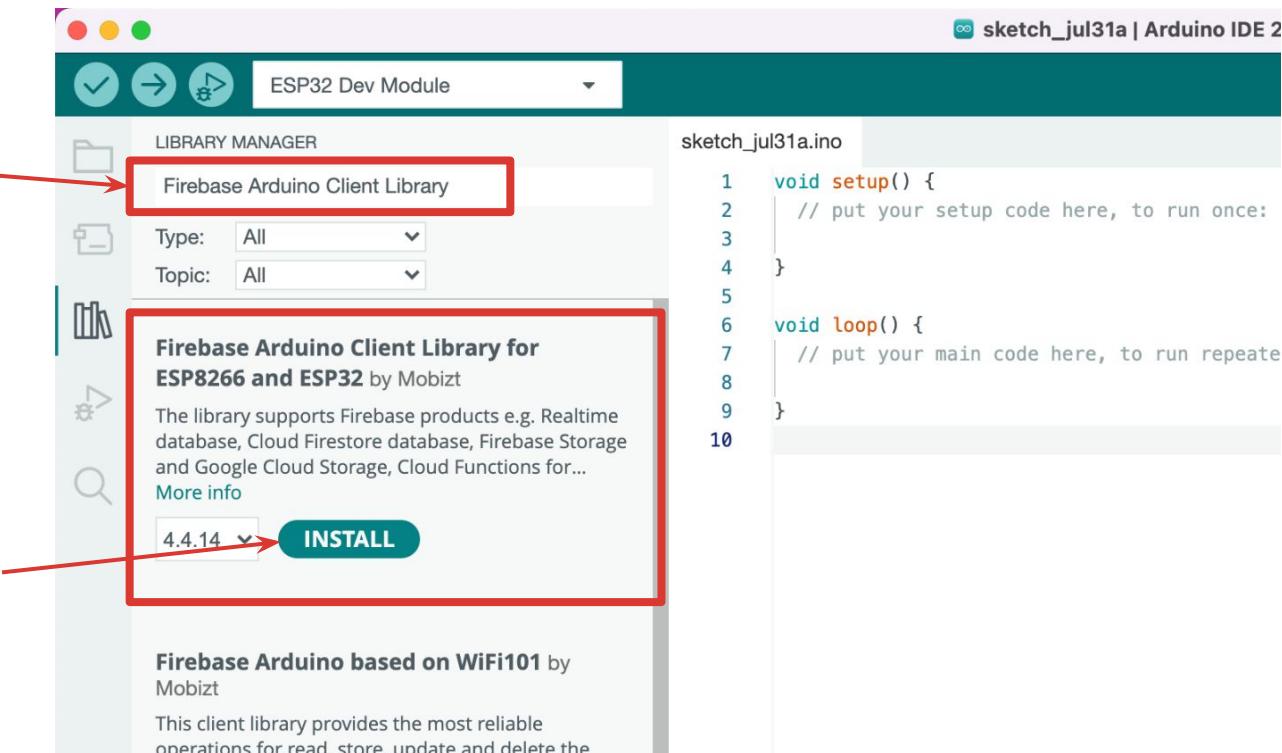
macOS Intel, 10.15: "Catalina" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

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

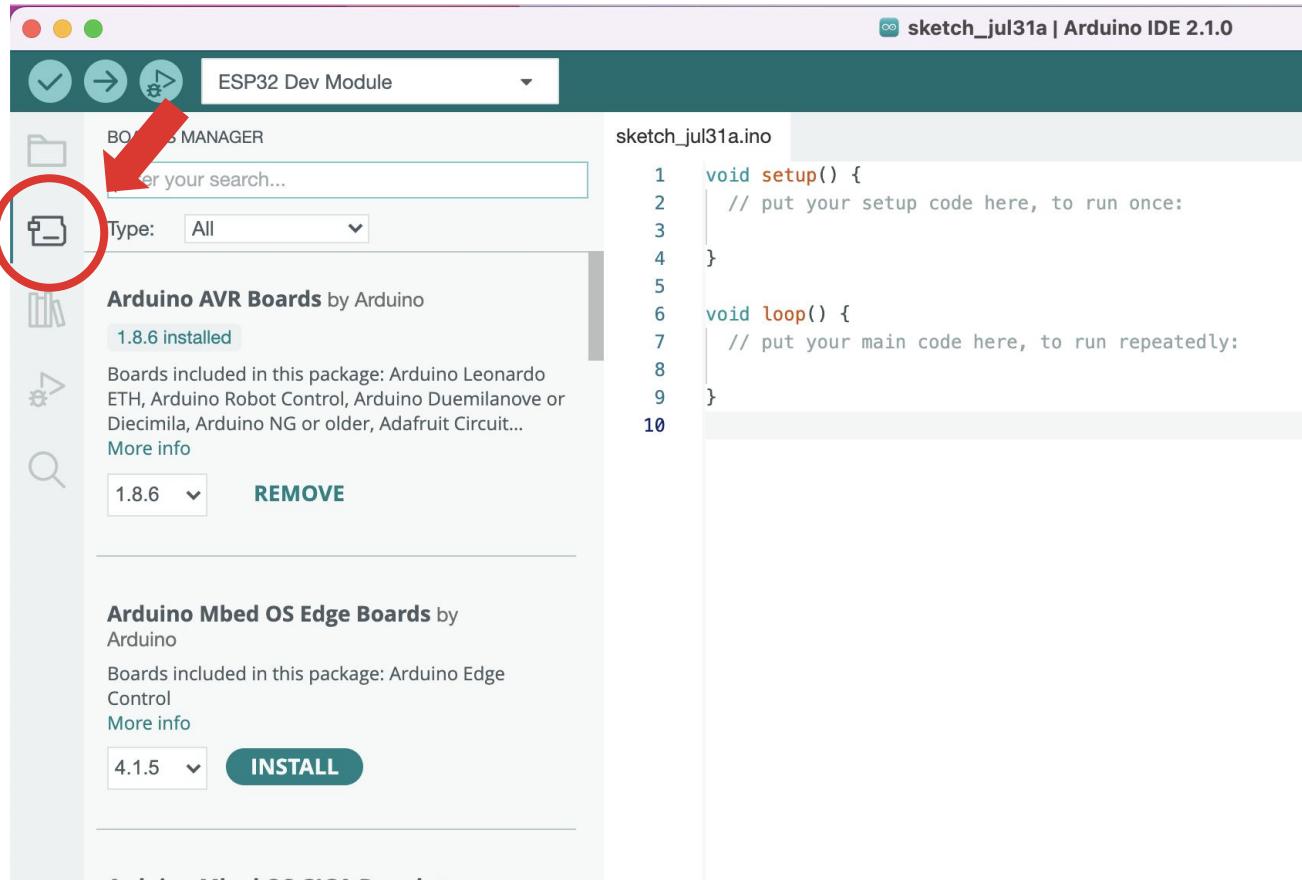


Step 3: In the search bar, type in: “**Firebase Arduino Client Library**”



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

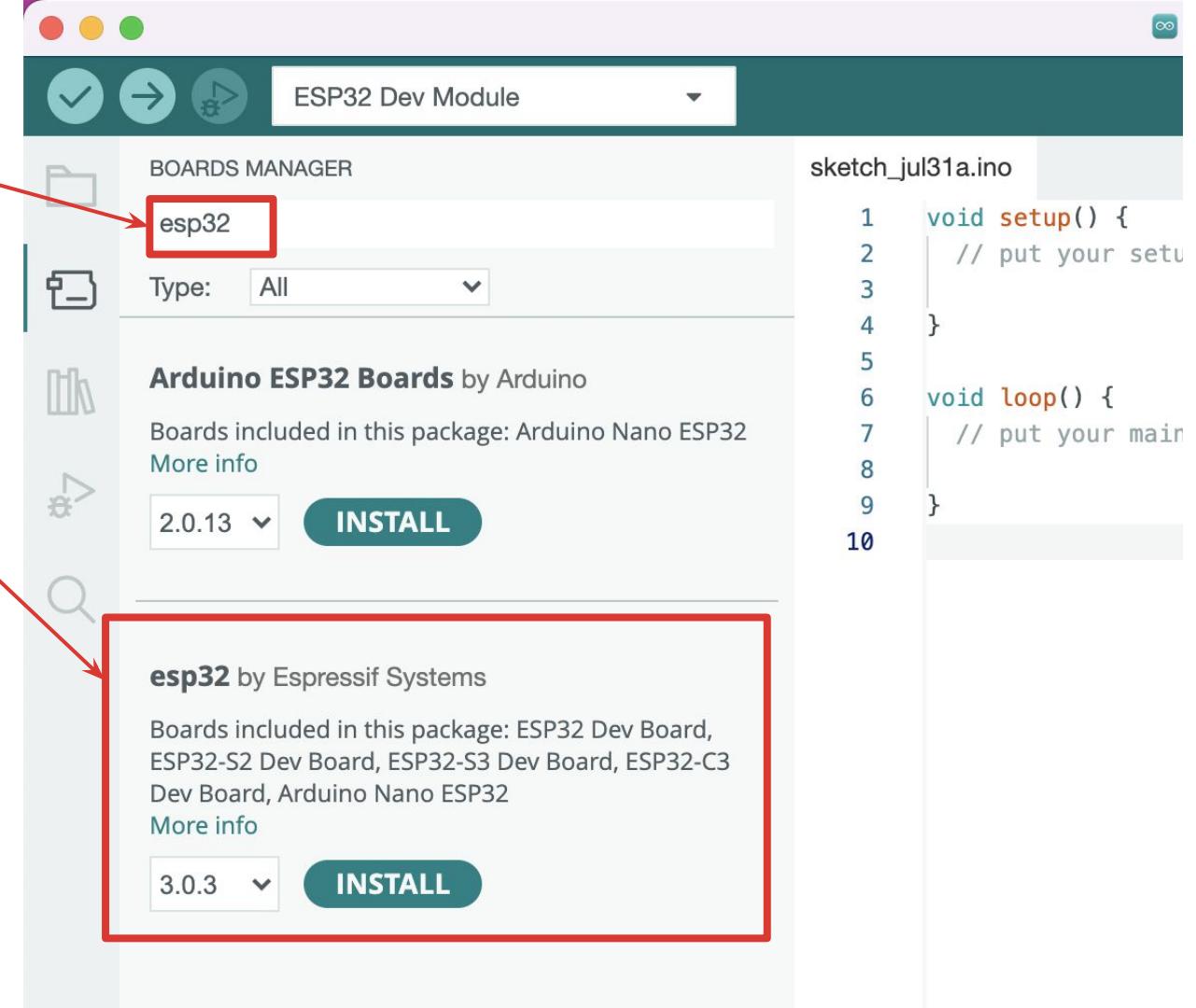
Step 5: Next, click on the Boards Manager icon.



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

Step 7: **Install** “**esp32**” library by Espressif Systems. 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**” folder.

The screenshot shows a GitHub repository named "Siggraph-Workshop". The repository is public and has 1 branch and 0 tags. The main branch is selected. There is a search bar for "Go to file" and buttons for "Add file" and "Code". A red arrow points to the "Code" folder in the file tree on the left, which is highlighted with a red box. The commit history shows the following changes:

File / Action	Description	Time
juliadaser Update README.md	43dd2f8 · 19 hours ago	55 Commits
3DModels Add files via upload	29 days ago	
Code Update Code.ino	2 days ago	
Media Add files via upload	3 days ago	
Slides Add files via upload	20 hours ago	
README.md Update README.md	19 hours ago	

At the bottom, there is a "README" file with edit and more options buttons.

Step 9: Once in the “**Code**” folder, click into “**Code.ino**”.

The screenshot shows a GitHub repository interface for the "Siggraph-Workshop" repository. The "Code" tab is selected, indicated by an orange underline. The navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. Below the navigation bar, there is a breadcrumb navigation showing "main" and "Siggraph-Workshop / Code". A commit message from "juliadaser" is visible, stating "Update Code.ino". The main content area displays a list of files. A red box highlights the "Code.ino" file, and a red arrow points to it from the bottom left. The "Name" column lists "Code.ino" and the "Last commit message" column shows "Update Code.ino".

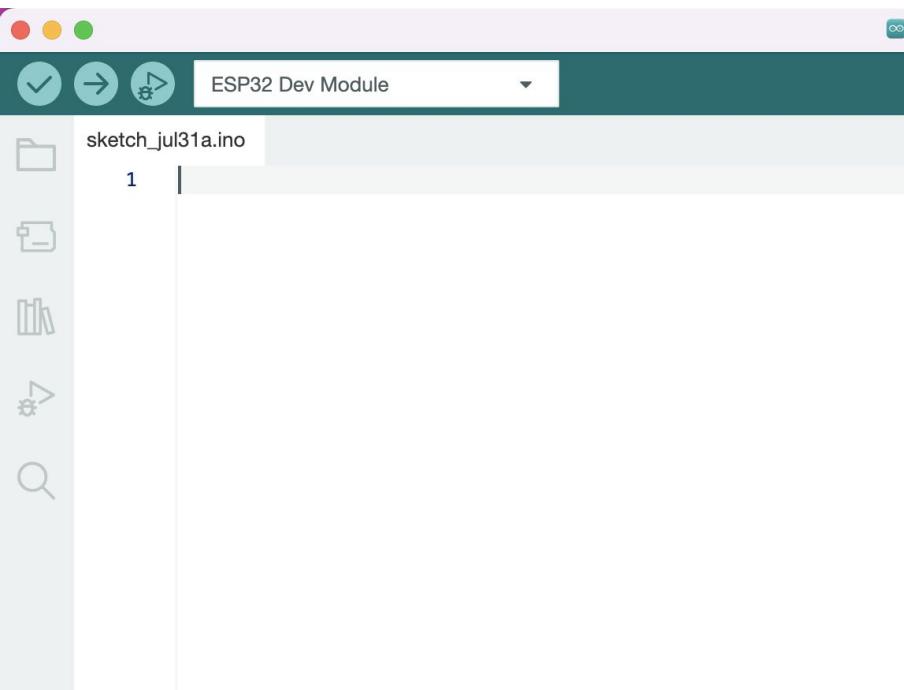
Name	Last commit message
...	Update Code.ino
Code.ino	Update Code.ino

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

The screenshot shows a GitHub code editor interface. At the top, there's a navigation bar with a sidebar icon, a dropdown for 'main', the repository name 'Siggraph-Workshop / Code / Code.ino', a search bar labeled 'Go to file', and a 'History' button. Below the header, a commit message from 'juliadaser' is shown: 'Update Code.ino' made at '41fb077 · 2 days'. The main area contains the 'Code' tab selected, showing 107 lines of C++ code for an ESP32. The code includes imports for WiFi and Firebase libraries, defines for Wi-Fi credentials, and constants for Firebase API key and database URL. A GitHub Copilot note indicates 'Code 55% faster with GitHub Copilot'. On the right side of the code editor, there's a toolbar with several icons, one of which is circled in red and has a red arrow pointing to it from above. This circled icon is the standard copy-to-clipboard icon.

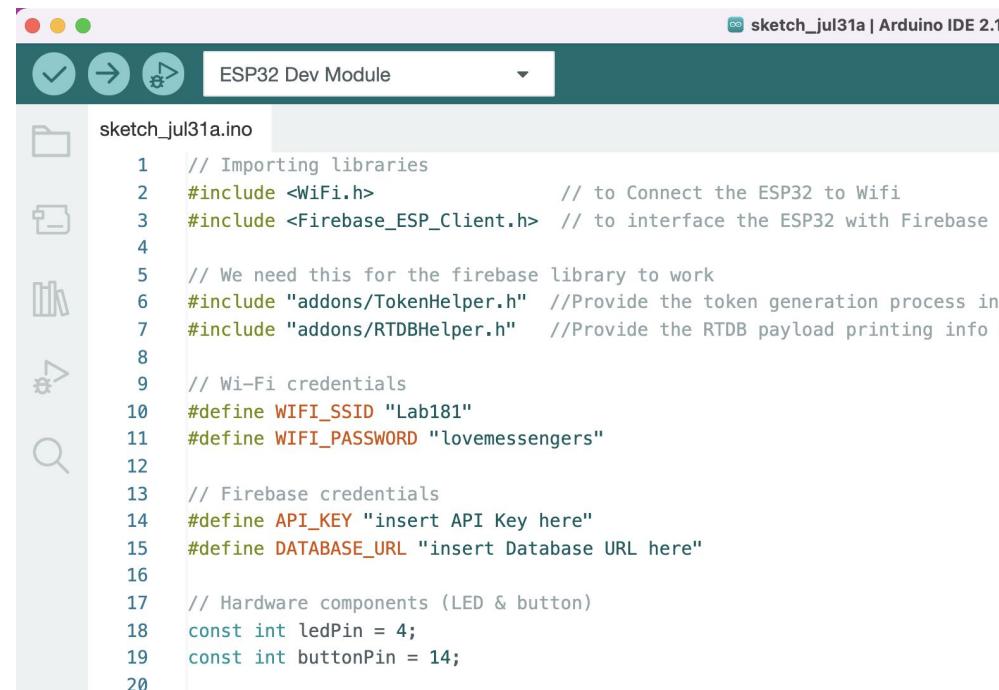
```
1 // Importing libraries
2 #include <WiFi.h> // to Connect the ESP32 to Wifi
3 #include <Firebase_ESP_Client.h> // to interface the ESP32 with Firebase
4
5 // We need this for the firebase library to work
6 #include "addons/TokenHelper.h" //Provide the token generation process info.
7 #include "addons/RTDBHelper.h" //Provide the RTDB payload printing info and other helper functions.
8
9 // Wi-Fi credentials
10 #define WIFI_SSID "Lab181"
11 #define WIFI_PASSWORD "lovemessengers"
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;
```

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



The screenshot shows the Arduino IDE interface. The title bar says "ESP32 Dev Module". The left sidebar has icons for file operations like new, open, save, and search. The main area shows a single tab named "sketch_jul31a.ino". The code editor is empty, with only the number "1" at the top left corner.

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



The screenshot shows the Arduino IDE interface with the same setup as the previous image, but now containing code. The title bar says "sketch_jul31a | Arduino IDE 2.0". The code editor displays the following C++ code:

```
// Importing libraries
#include <WiFi.h> // to Connect the ESP32 to Wifi
#include <Firebase_ESP_Client.h> // to interface the ESP32 with Firebase
// We need this for the firebase library to work
#include "addons/TokenHelper.h" //Provide the token generation process in
#include "addons/RTDBHelper.h" //Provide the RTDB payload printing info
// Wi-Fi credentials
#define WIFI_SSID "Lab181"
#define WIFI_PASSWORD "lovemessengers"
// Firebase credentials
#define API_KEY "insert API Key here"
#define DATABASE_URL "insert Database URL here"
// Hardware components (LED & button)
const int ledPin = 4;
const int buttonPin = 14;
```

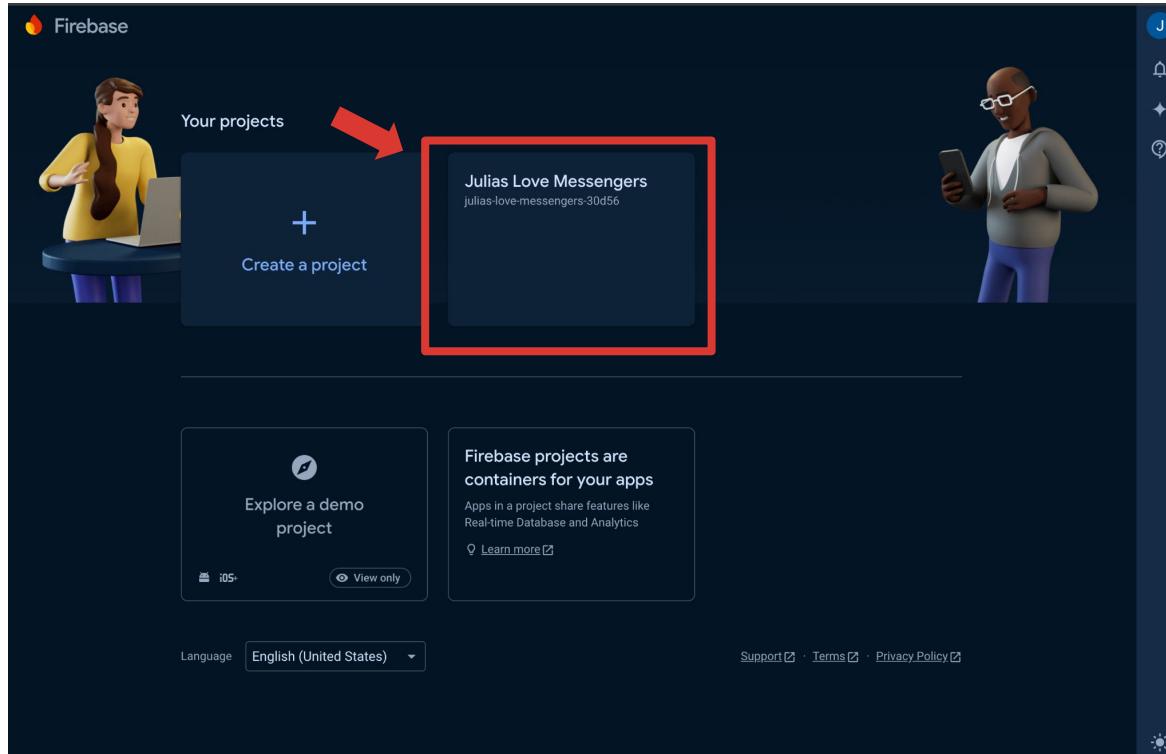
Step 13: Insert your home WIFI Name and Password.

```
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/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 "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 }
```

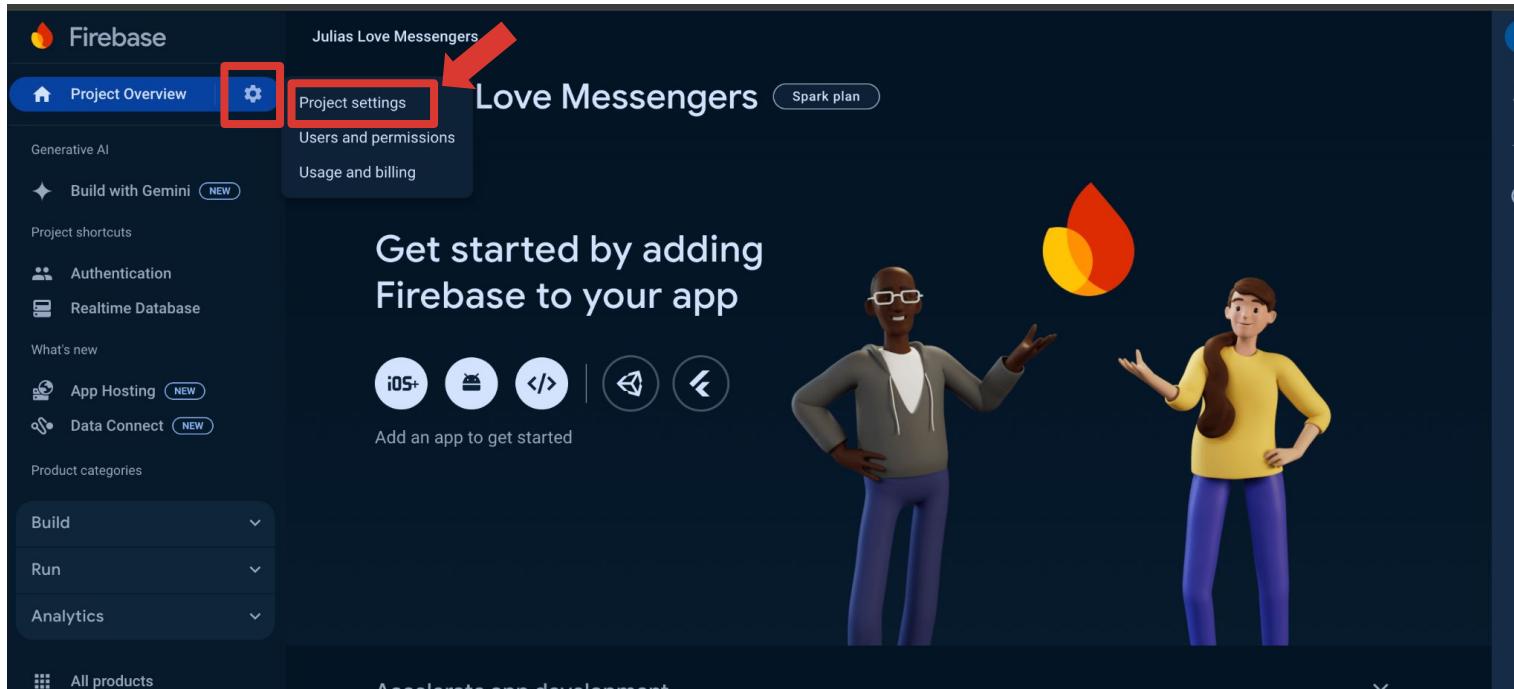
For example:

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

Step 14: Next, you insert the unique API Key of your Database in the code. To find it, head to 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 15: Inside your Firebase Console, select the little **gear icon**, and go to **“Project Settings”**



Step 16: 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, the "Web API Key" field is highlighted with a red box, containing the value "AlzaSyAYMhb4PoR5BuE9RndOaYAZ409srs4gNow".

Project settings

General Cloud Messaging Integrations Service accounts Data privacy Users and permissions

Your project

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

Environment

This setting customizes your project for different stages of the app lifecycle

Environment type Unspecified

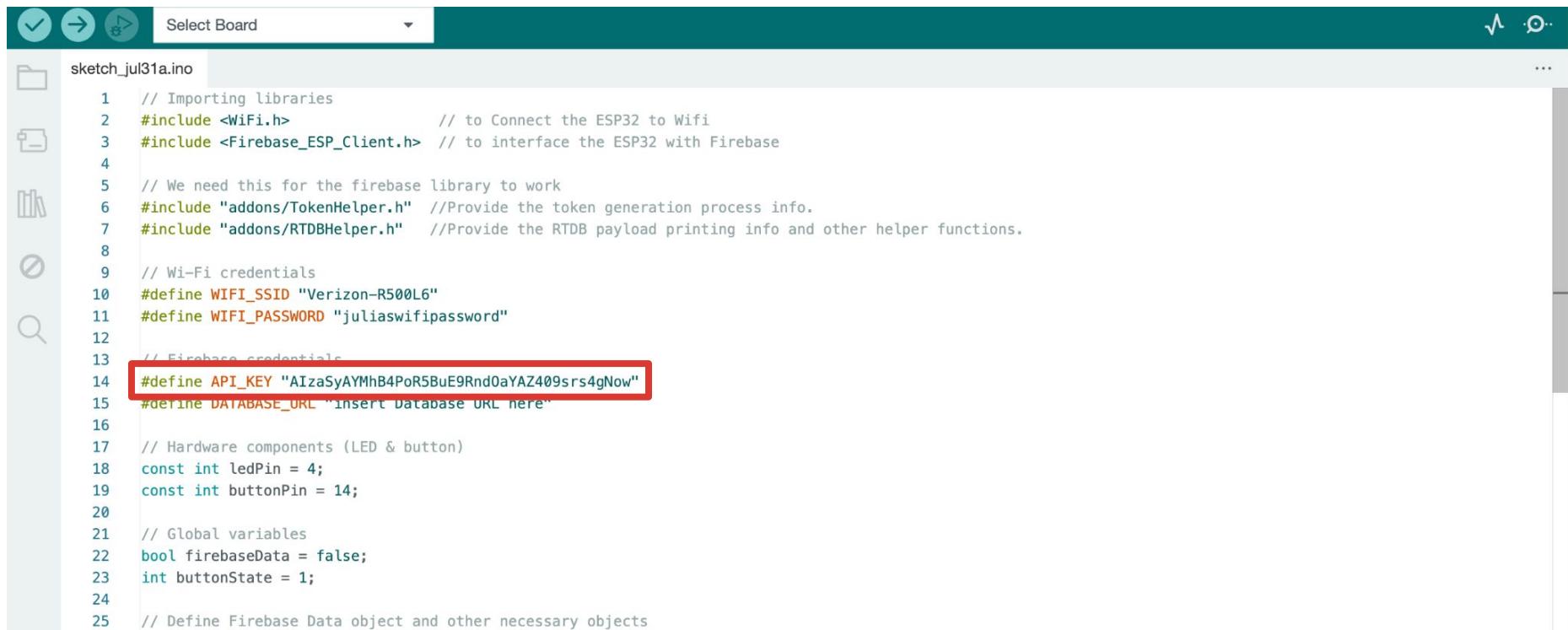
Your apps

There are no apps in your project

Select a platform to get started

iOS+ | Android | Web | React Native | Flutter

Step 17: 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
```

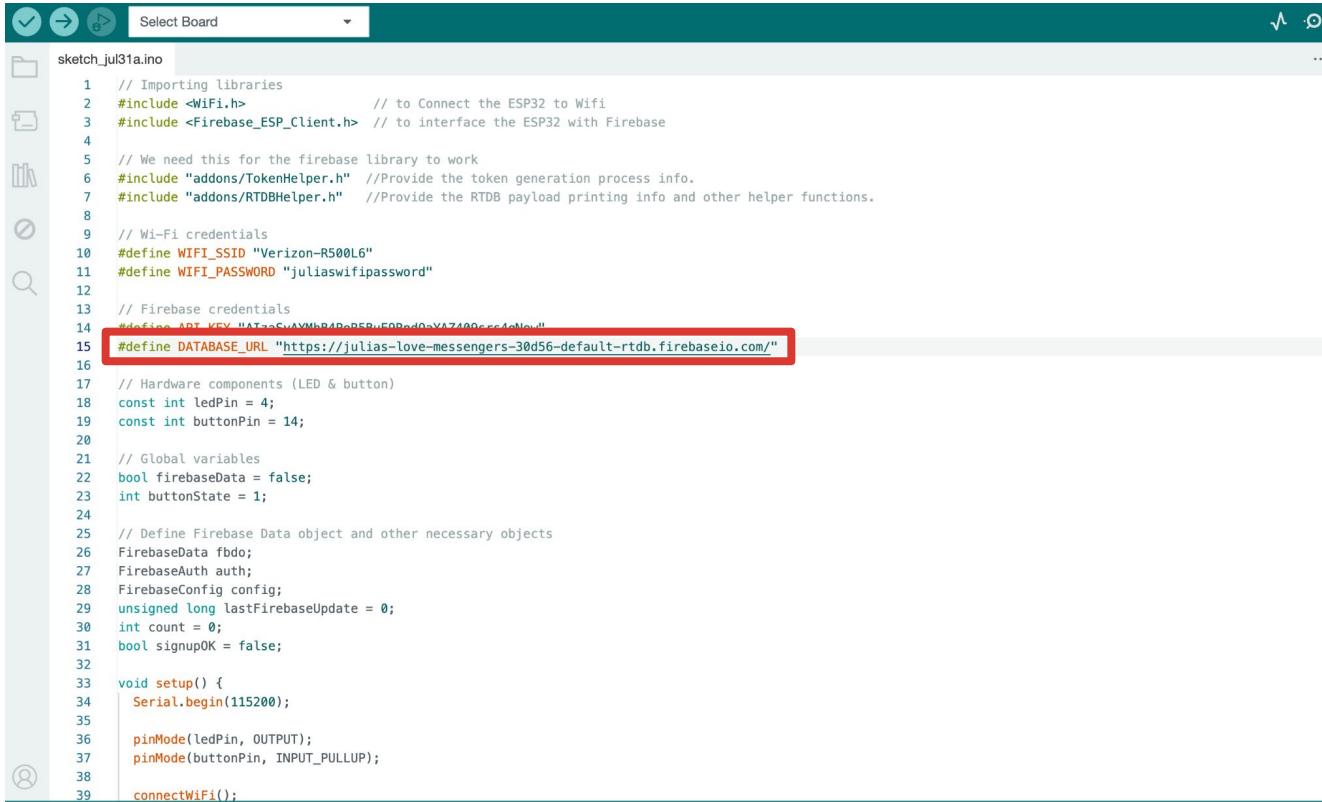
Step 18: Next, we will find your Database- URL. On Firebase, head to “**Build**” and select “**Realtime Database**”

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 bottom of the sidebar to the "Realtime Database" link. The main content area has tabs for General, Cloud Messaging, Integrations, Service accounts, Data privacy, and Users and permissions. Under "General", there is a "Your project" section with fields for Project name (Julias Love Messengers), Project ID (julias-love-messengers-30d56), Project number (759000092849), and Web API Key (AlzaSyAYMhB4PoR5BuE9RnD0aYAZ409srs4gNow). Below that is an "Environment" section with an "Unspecified" environment type. At the bottom, there is a "Your apps" section stating "There are no apps in your project" and a "Select a platform to get started" button, followed by icons for iOS+, Android, Web, and other platforms.

Step 19: 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 background), App Hosting, Data Connect, Product categories, Build, Run, Analytics, and All products. Below these are Related development tools: IDB (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 contains 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 X. Below the banner, the database URL is displayed as a redacted link: <https://julia's-love-messengers-30d56-default-rtbd.firebaseio.com>. Underneath the URL, the database structure is shown with a single node: user_1.0 and user_2.0. At the bottom of the page, it says "Database location: United States (us-central1)".

Step 20: Paste the Database URL into the Arduino Code into the double quotes after "#define DATABASE_URL"



The screenshot shows the Arduino IDE interface with the file "sketch_jul31a.ino" open. The code is written in C++ and defines various constants and functions for an ESP32 connected to WiFi and interfacing with Firebase. A specific line of code, "#define DATABASE_URL", is highlighted with a red rectangular box. This line contains the database URL for a Firebase Realtime Database instance.

```
// Importing libraries
#include <WiFi.h> // to Connect the ESP32 to Wifi
#include <firebase_ESP_Client.h> // to interface the ESP32 with Firebase

// We need this for the firebase library to work
#include "addons/TokenHelper.h" //Provide the token generation process info.
#include "addons/RTDBHelper.h" //Provide the RTDB payload printing info and other helper functions.

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

// Firebase credentials
#define APP_KEY "AIzaSyAVMbR4DnDFBvF0Dad0aVAZ400cc4aMowd"
#define DATABASE_URL "https://julias-love-messengers-30d56-default.firebaseio.com/"

// Hardware components (LED & button)
const int ledPin = 4;
const int buttonPin = 14;

// Global variables
bool firebaseData = false;
int buttonState = 1;

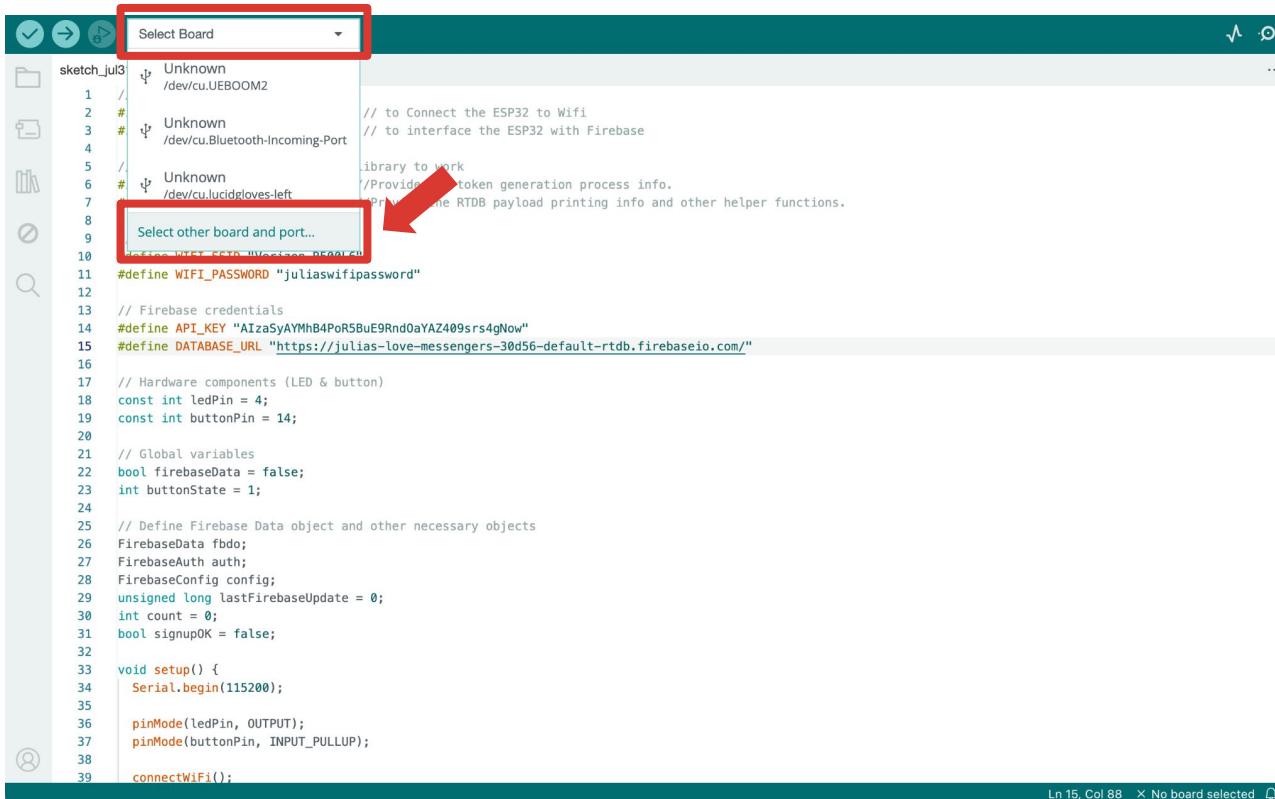
// Define Firebase Data object and other necessary objects
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
unsigned long lastFirebaseUpdate = 0;
int count = 0;
bool signupOK = false;

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

  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);

  connectWiFi();
}
```

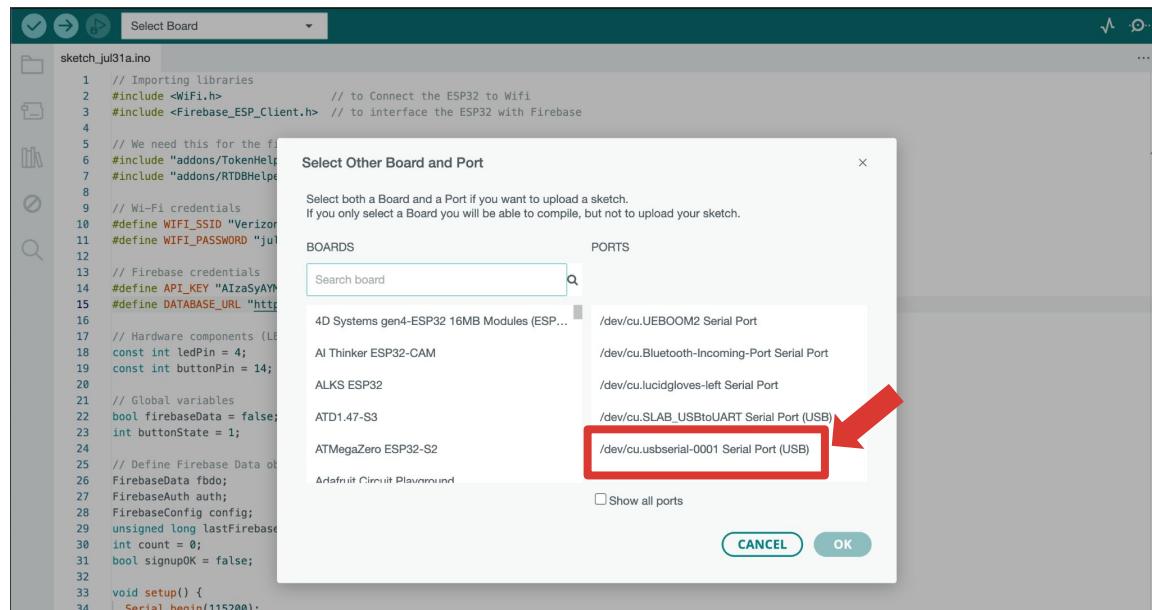
Step 21: The code is now ready to be uploaded to the first Love Messenger.
Go to “**Select Board**”, and go to “**Select other board and port...**”



Step 22: Plug in your first Love Messenger to your Computer using a data-transfer USB cable.

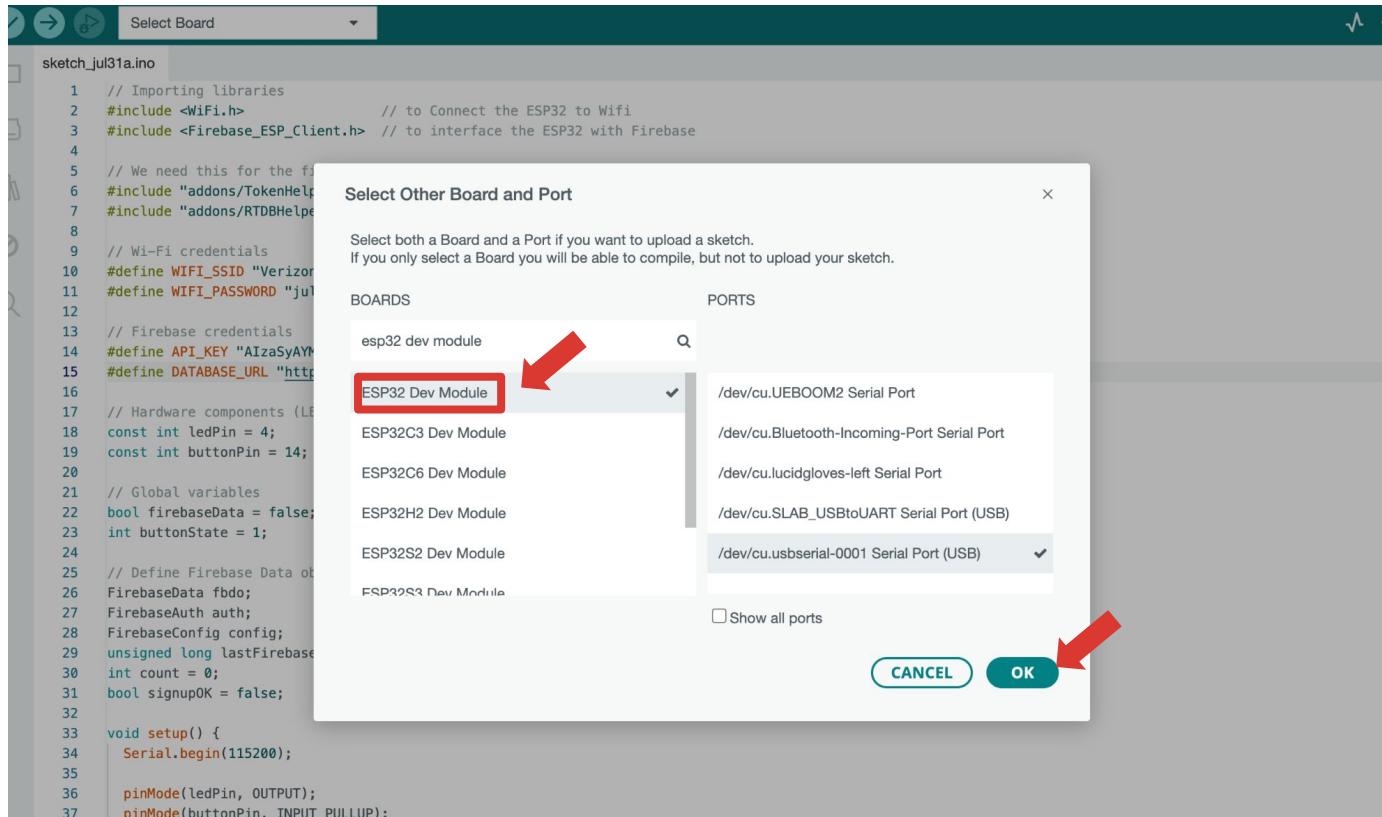
Once the Love Messenger is plugged in, a new port should appear in the “Ports” section. Depending on if you are using a Mac or PC, the port names can vary (“COM” for PC, and “usbserial” for Mac).

Select the new port.

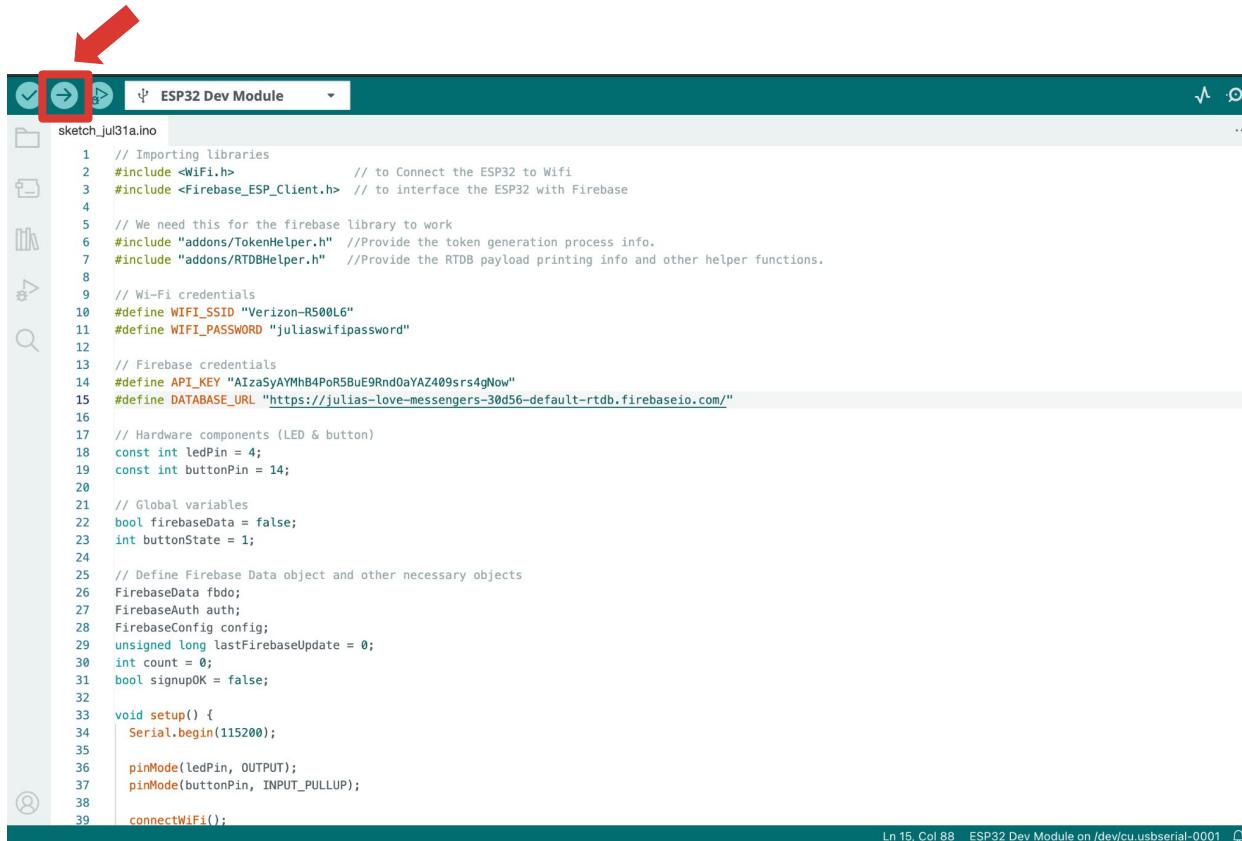


**If your port does not show up, you might be using a charge-only USB wire that doesn't transmit data. You need to use a USB wire that also can do data-transfer. We have the ones we used in the workshop linked on Github.*

Step 23: In the “**Boards**” section, search for “**ESP32 Dev Module**”, and select it. Then Press OK



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.

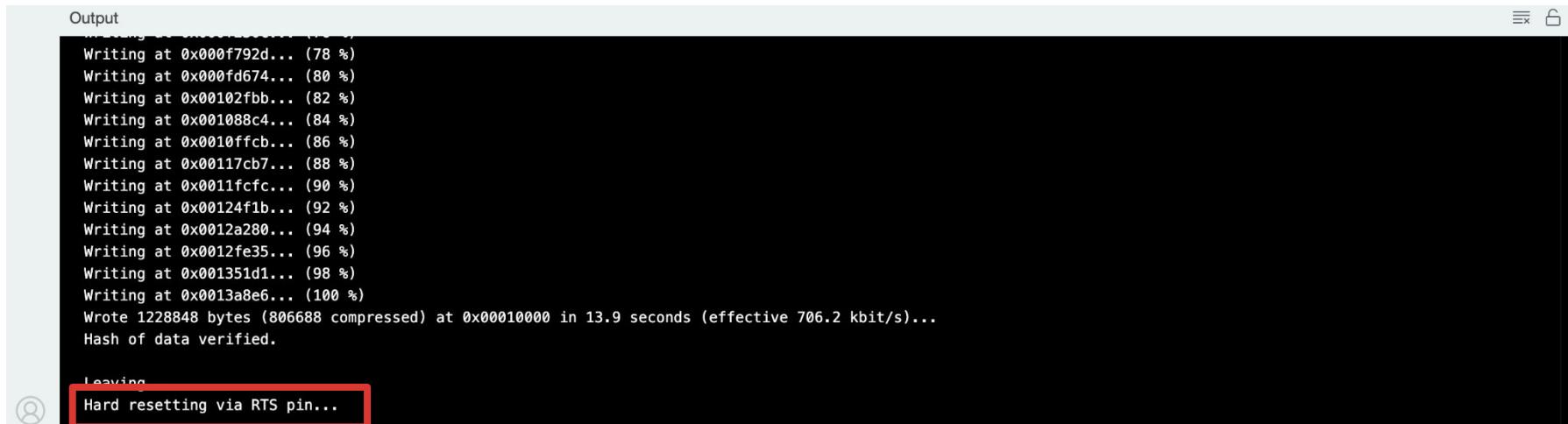


```
sketch_jul31a.ino
1 // Importing libraries
2 #include <WiFi.h>           // to Connect the ESP32 to Wifi
3 #include <Firebase_ESP_Client.h> // to interface the ESP32 with Firebase
4
5 // We need this for the firebase library to work
6 #include "addons/TokenHelper.h" //Provide the token generation 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...**”

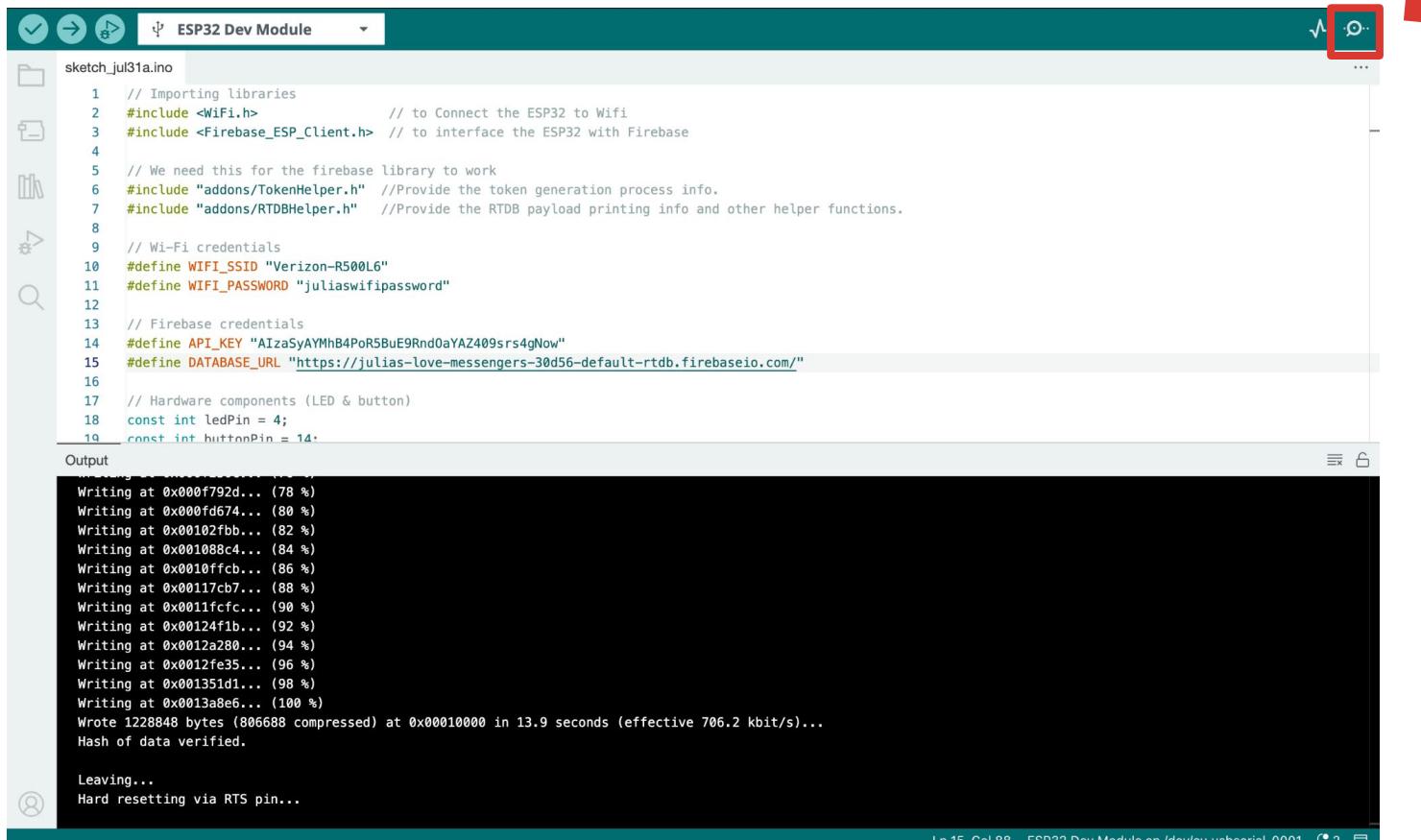


Output

```
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:** The main window displays 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:** A red arrow points to the "Serial Monitor" icon (a small monitor with a circular refresh button) in the top right corner of the main window.

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”

```
82 void uploadData(int buttonstate) {  
83     if (Firebase.RTDB.setInt(&fbdo, "/user_1" buttonstate)) {  
84         Serial.printf("Data UPLOAD successful, ");  
85     } else {  
86         Serial.println("Data UPLOAD failed, ");  
87     }  
88 }
```

Change to

```
82 void uploadData(int buttonstate) {  
83     if (Firebase.RTDB.setInt(&fbdo, "/user_2" buttonstate)) {  
84         Serial.printf("Data UPLOAD successful, ");  
85     } else {  
86         Serial.println("Data UPLOAD failed, ");  
87     }  
88 }
```

Step 30: Similarly, in your code, in the **download data** function, change “/user_2” to “/user_1”

```
90 void downloadData() {  
91     if (Firebase.RTDB.getInt(&fbdo, "/user_2")) {  
92         firebaseData = fbdo.intData();  
93         Serial.println("Data DOWNLOAD successful");  
94     } else {  
95         Serial.println("Data DOWNLOAD failed");  
96     }  
97 }
```

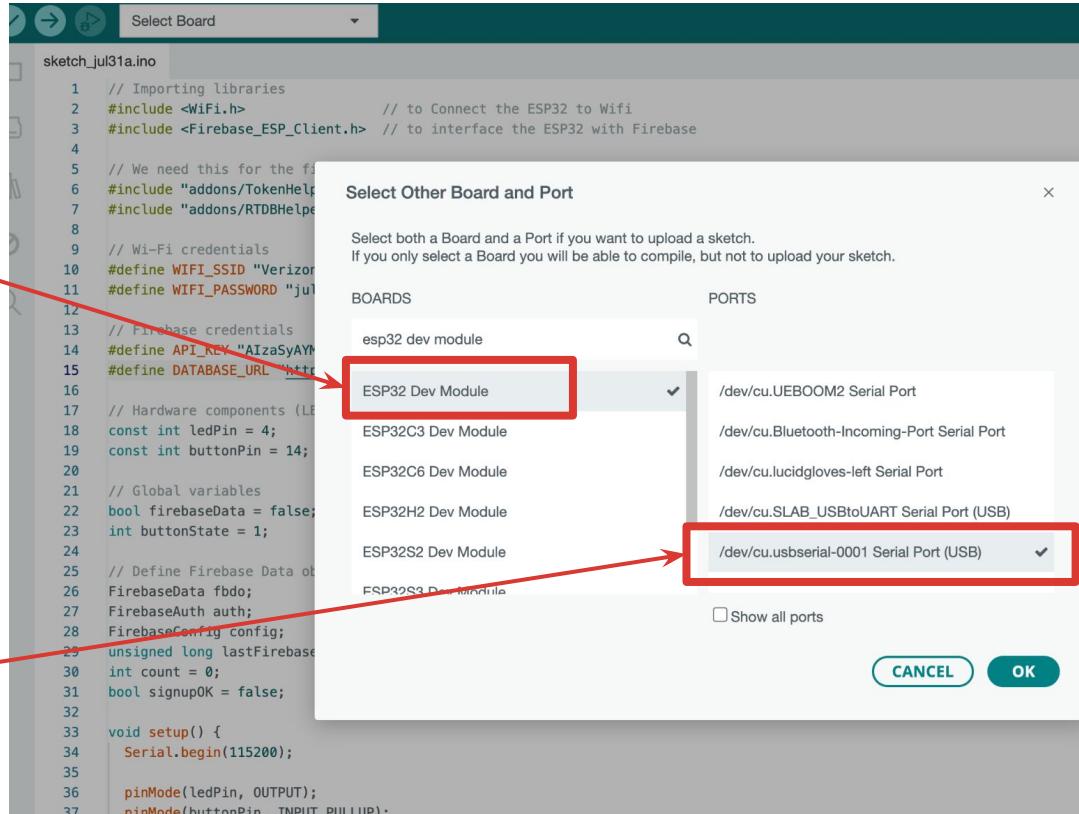
Change to

```
90 void downloadData() {  
91     if (Firebase.RTDB.getInt(&fbdo, "/user_1")) {  
92         firebaseData = fbdo.intData();  
93         Serial.println("Data DOWNLOAD successful");  
94     } else {  
95         Serial.println("Data DOWNLOAD failed");  
96     }  
97 }
```

Step 31:

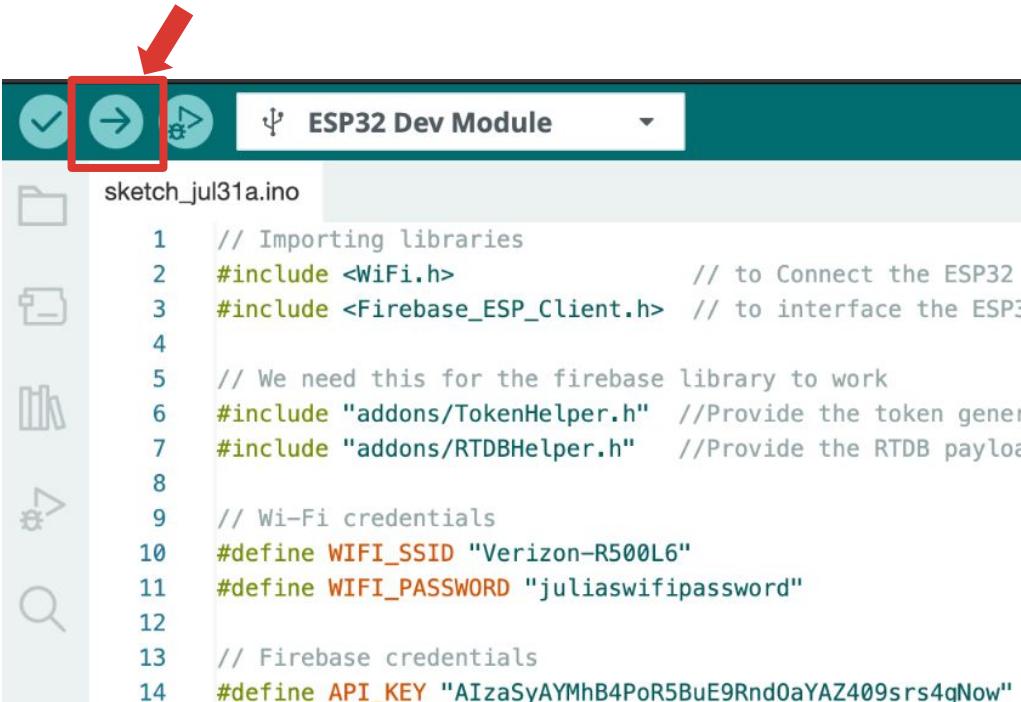
Once again, under “Select Board”, you should have **“ESP32 Dev Module”** selected under BOARDS.

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 X: 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 34: 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 34: While the Love Messengers are working, you should be able to see Firebase updating in real time.

Default Firebase states

🔗 <https://julias-love-messengers-30d56-default-rtdb.firebaseio.com>

```
https://julias-love-messengers-30d56-default-rtdb.firebaseio.com/  
  └── user_1: 1  
  └── user_2: 1
```

When user_1's button is successfully pressed- both Love Messengers should light up!

🔗 <https://julias-love-messengers-30d56-default-rtdb.firebaseio.com>

```
https://julias-love-messengers-30d56-default-rtdb.firebaseio.com/  
  └── user_1: 0  
  └── user_2: 1
```

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

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

We are more than happy to help! ❤️