

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 also has a glowing center. The hands are positioned to showcase the glowing objects.

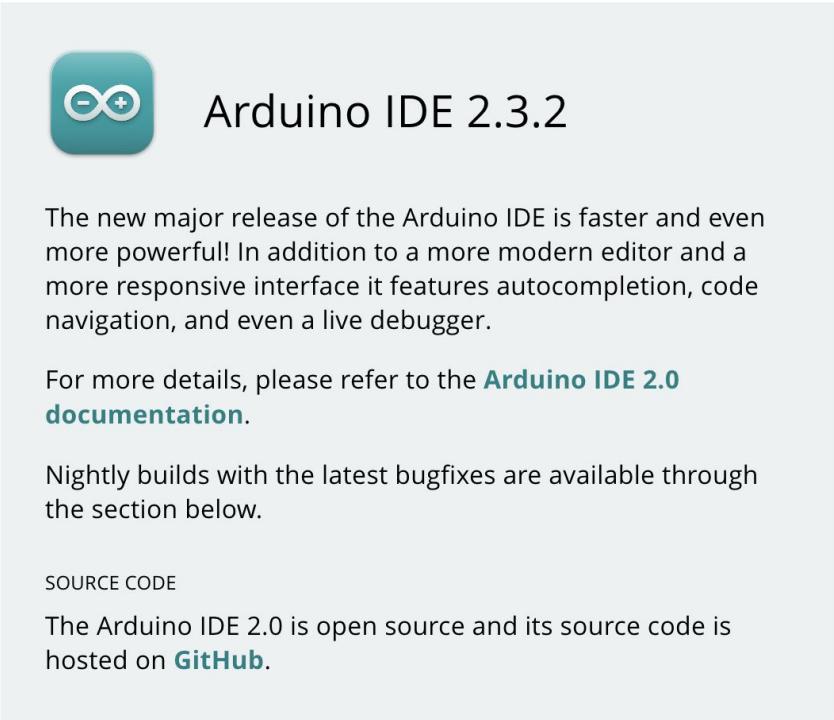
Instructions

Uploading the Arduino Code

@ NYC Resistor

Step 1: Download Arduino IDE Version 2.3.2 here:

<https://www.arduino.cc/en/software>



The image shows the Arduino IDE 2.3.2 landing page. It features a teal header with the Arduino logo and the text "Arduino IDE 2.3.2". Below the header, there is a large text block describing the new features of the release. A red arrow points from the "DOWNLOAD OPTIONS" section on the right towards the "Windows" download links.

DOWNLOAD OPTIONS

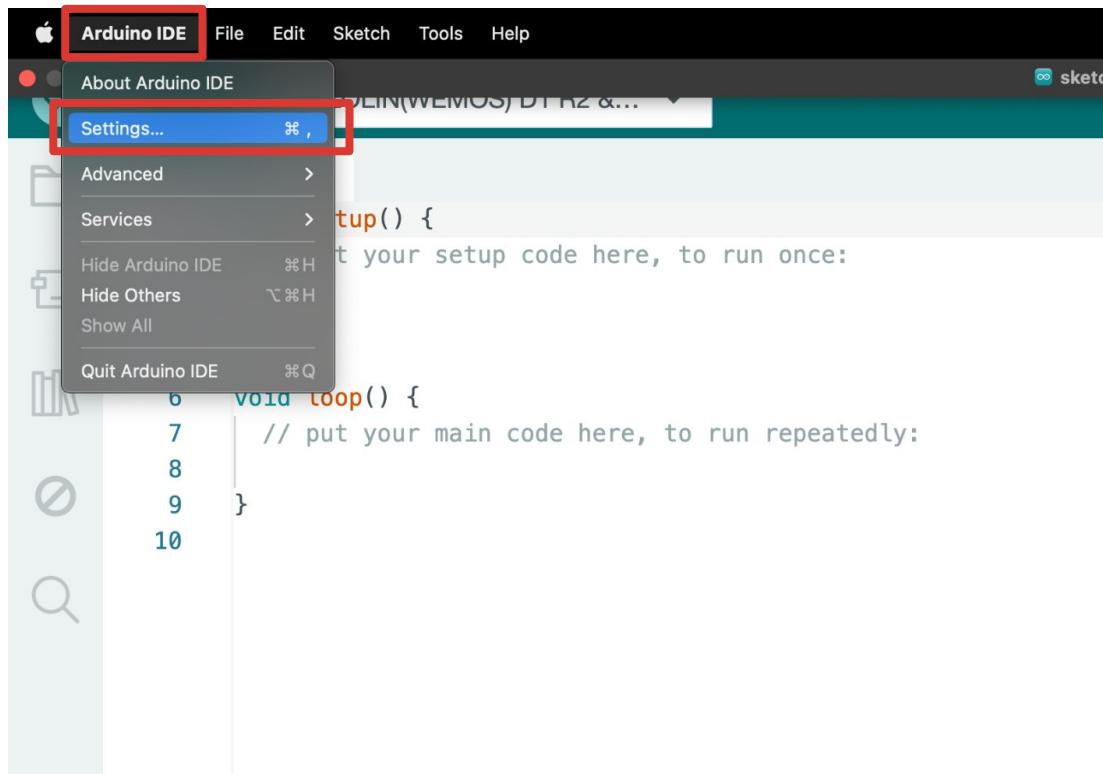
Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

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

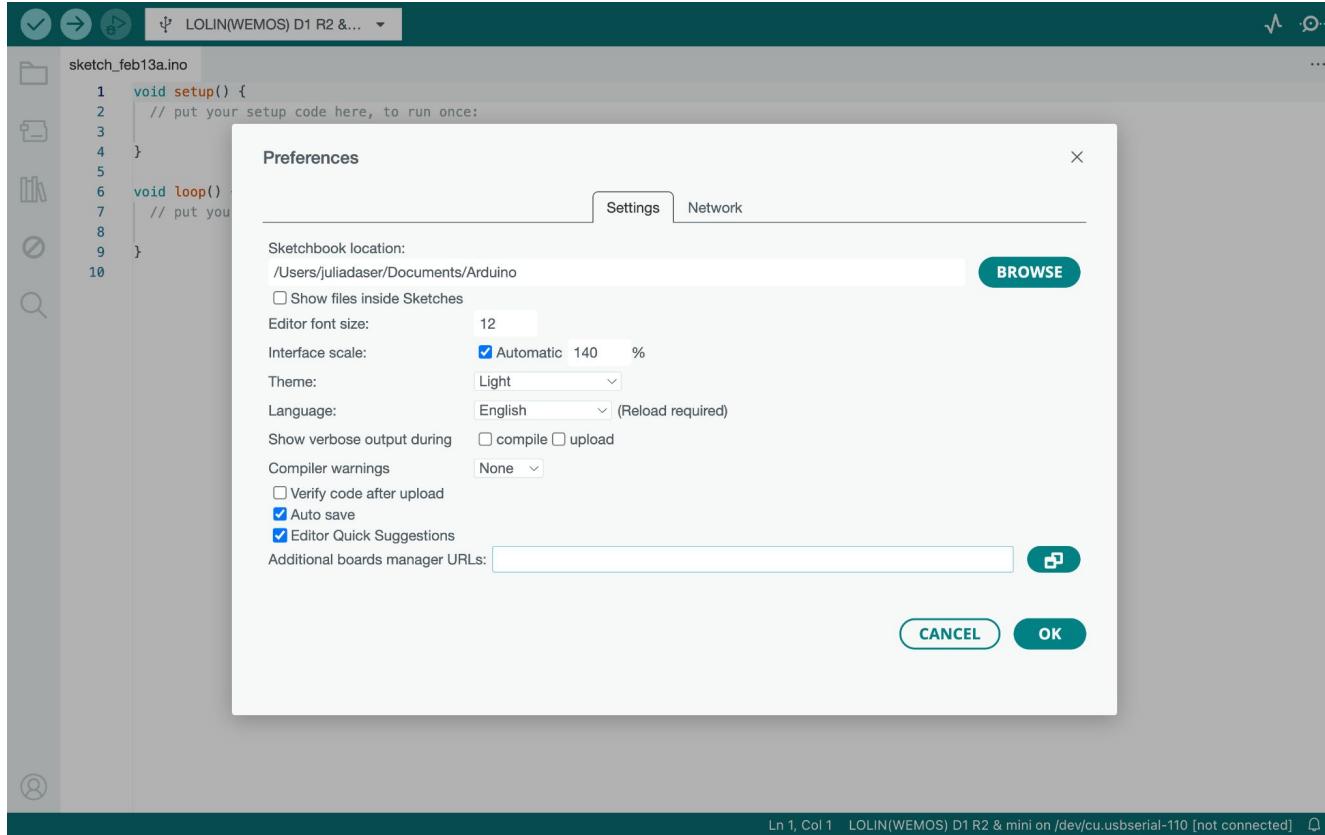
macOS Intel, 10.15: "Catalina" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

Step 2: Open the Software. If you are using a Mac click on “**Arduino IDE**” - “**Settings...**”. If you are using a windows PC, click on “**File**” - “**Preferences**”



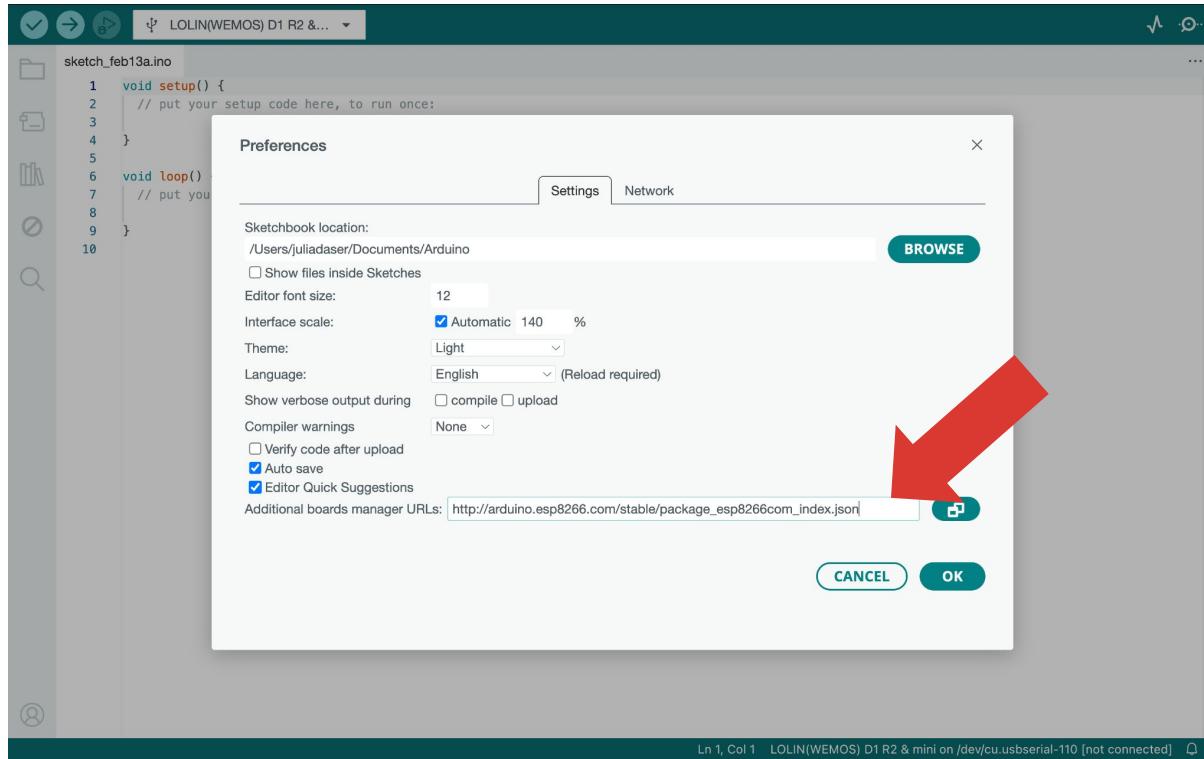
This pop-up window will open.



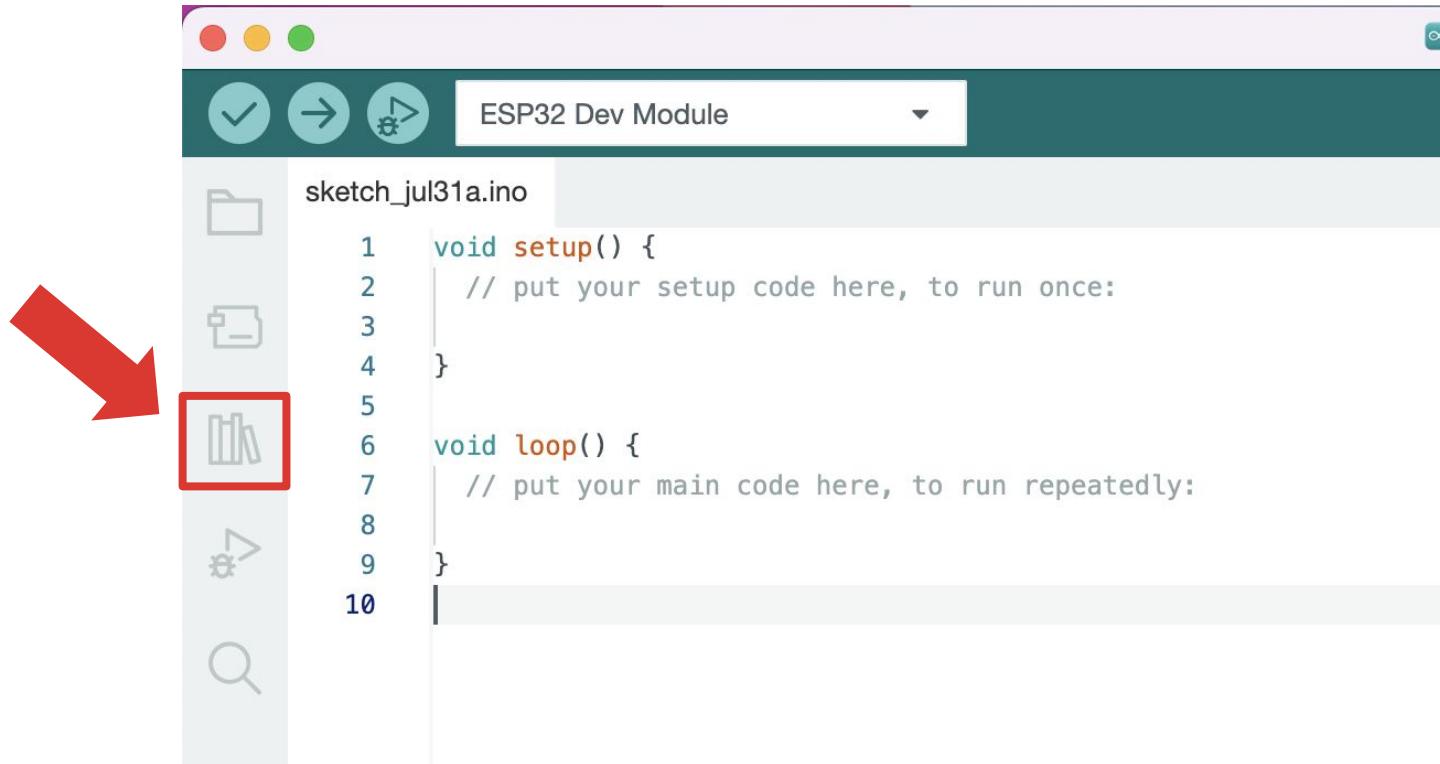
Step 3: Paste the link below into the ‘Additional boards manager URL’s” field.

http://arduino.esp8266.com/stable/package_esp8266com_index.json

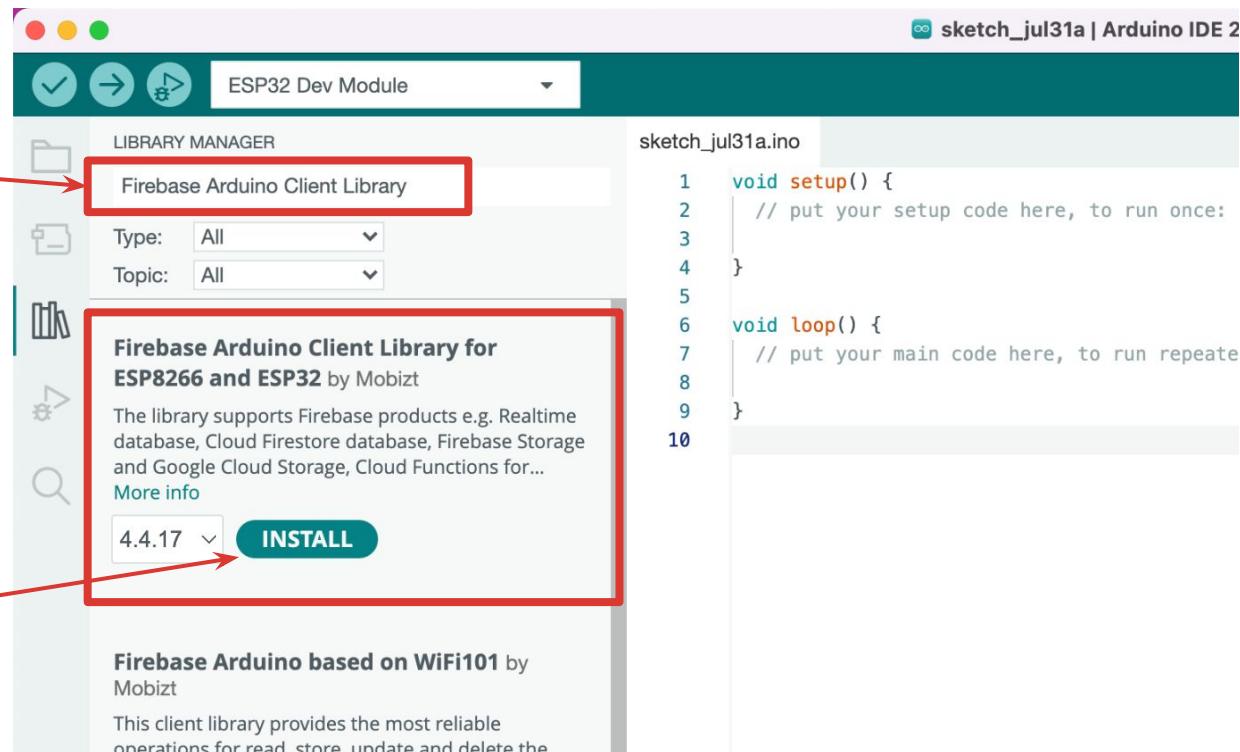
Then click ‘OK’



Step 4: Next, click on the “**Libraries**” icon.

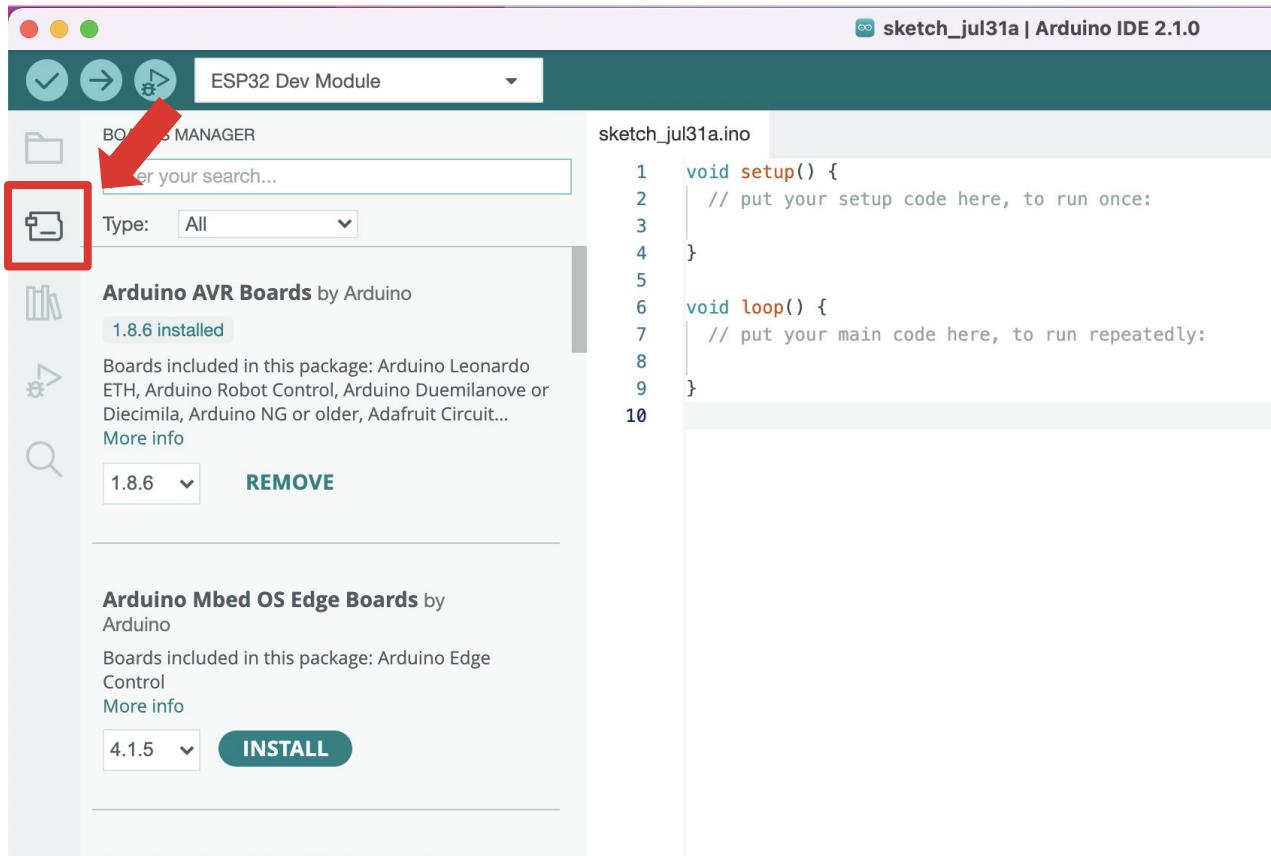


Step 5: In the search bar, type in: “**Firebase Arduino Client Library**”

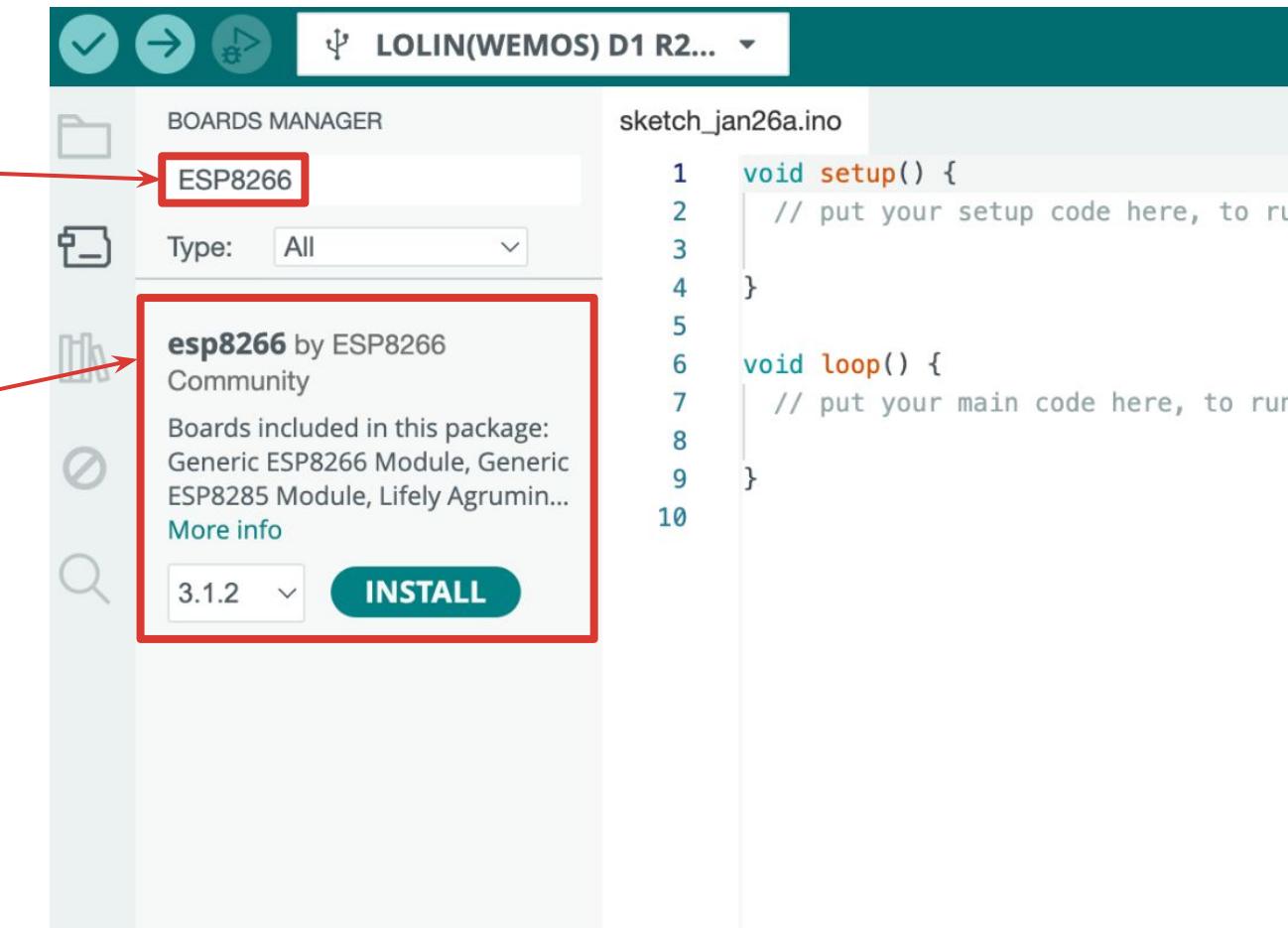


Step 6: Install “Firebase Arduino Client Library for ESP8266 and ESP32” by Mobitz

Step 7: Next, click on the Boards Manager icon.



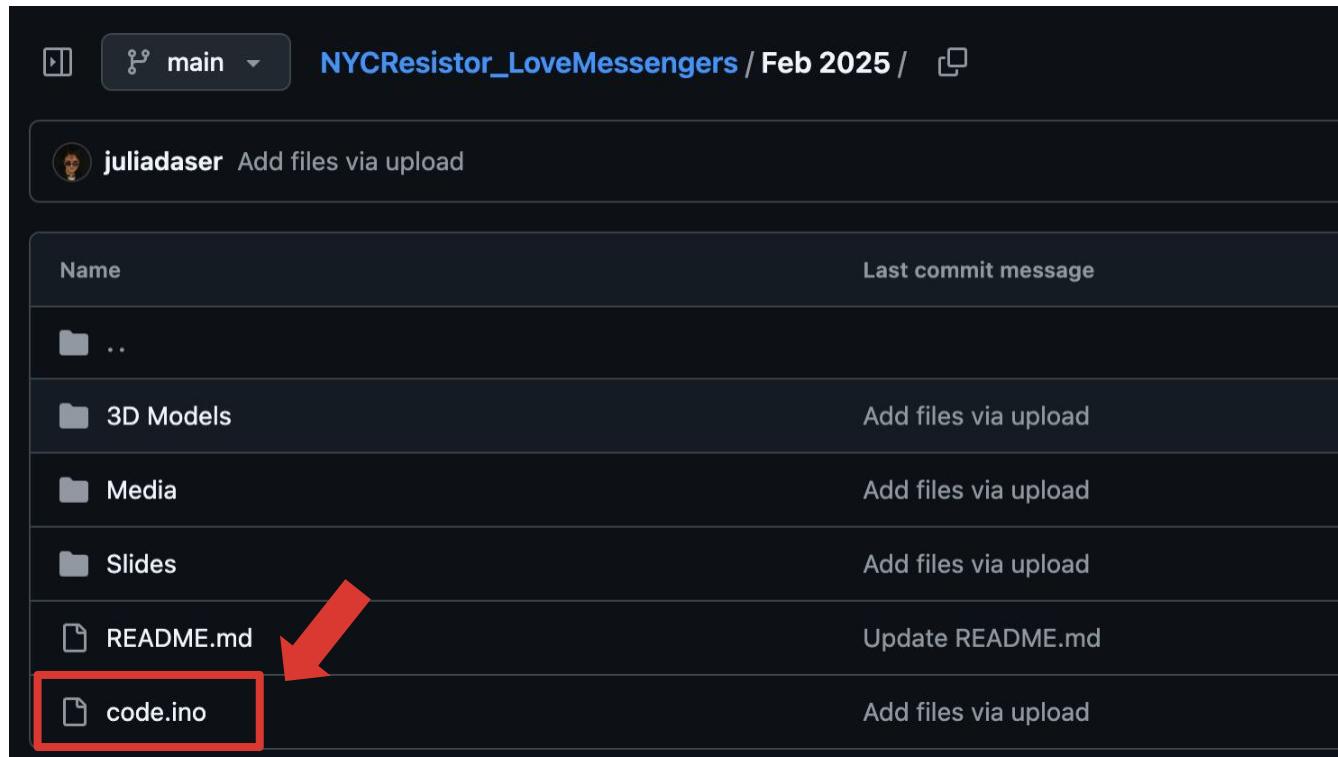
Step 8: In the search bar, search “**ESP8266**”



Step 9: Install “esp8266” library by ESP8266 Community.
This could take some time.

Step 10: Head to our GitHub repository:

https://github.com/pepzicles/NYCResistor_LoveMessengers/tree/main/Feb%202025 and go into the “**Code.ino**” file.



Step 11: Hit this button to copy all the code.

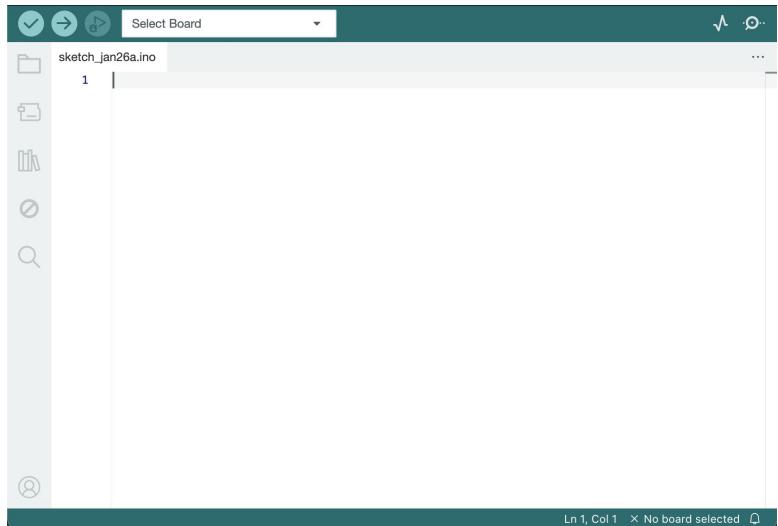


Code Blame

Raw

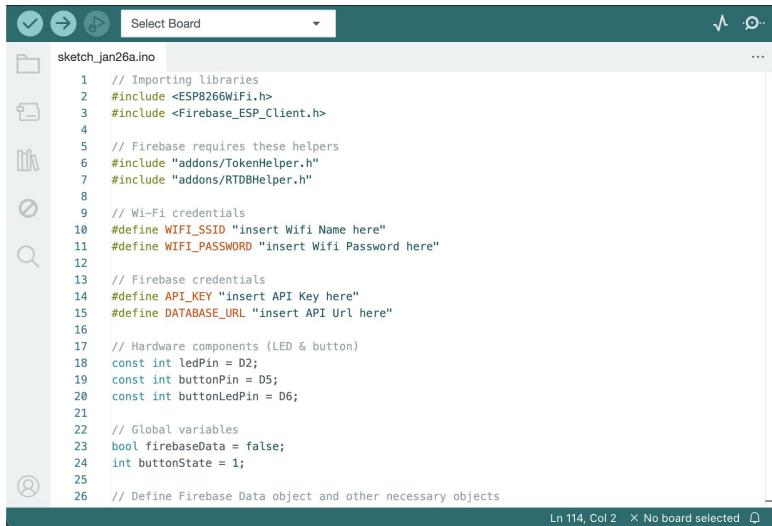
```
1 // Importing libraries
2 #include <ESP8266WiFi.h>
3 #include <Firebase_ESP_Client.h>
4
5 // Firebase requires these helpers
6 #include "addons/TokenHelper.h"
7 #include "addons/RTDBHelper.h"
8
9 // Wi-Fi credentials
10 #define WIFI_SSID "insert Wifi Name here"
11 #define WIFI_PASSWORD "insert Wifi Password here"
12
13 // Firebase credentials
14 #define API_KEY "insert API Key here"
15 #define DATABASE_URL "insert API Url here"
16
17 // Hardware components (LED & button)
18 const int ledPin = D2;
19 const int buttonPin = D5;
20 const int buttonLedPin = D6;
21
22 // Global variables
23 bool firebaseData = false;
24 int buttonState = 1;
25
26 // Define Firebase Data object and other necessary objects
27 FirebaseData fbdo;
28 FirebaseAuth auth;
```

Step 12: Delete all the code that is currently in your Arduino IDE.



The screenshot shows the Arduino IDE interface. The top bar has icons for file operations and a dropdown menu labeled "Select Board". Below the menu is a toolbar with icons for file operations like Open, Save, and Print. The main workspace is titled "sketch_jan26a.ino" and contains a single line of code: "1". On the left side, there's a vertical sidebar with icons for file, folder, and search functions. At the bottom, there's a status bar with the text "Ln 1, Col 1" and "No board selected".

Step 13: Paste all the code you have copied from the GitHub repository into here.



The screenshot shows the Arduino IDE interface with a much longer sketch. The top bar and toolbar are identical to the first screenshot. The main workspace now contains 26 lines of code, which is a copy of the provided GitHub repository content. The code includes imports for WiFi and Firebase libraries, defines WiFi SSID and password, sets up Firebase credentials, defines hardware components (LED and button pins), and initializes global variables. The status bar at the bottom indicates "Ln 114, Col 2" and "No board selected".

```
1 // Importing libraries
2 #include <ESP8266WiFi.h>
3 #include <Firebase_ESP_Client.h>
4
5 // Firebase requires these helpers
6 #include "addons/TokenHelper.h"
7 #include "addons/RTDBHelper.h"
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 API Url here"
16
17 // Hardware components (LED & button)
18 const int ledPin = D2;
19 const int buttonPin = D5;
20 const int buttonLedPin = D6;
21
22 // Global variables
23 bool firebaseData = false;
24 int buttonState = 1;
25
26 // Define Firebase Data object and other necessary objects
```

Step 14: Insert your home WIFI Name and Password.

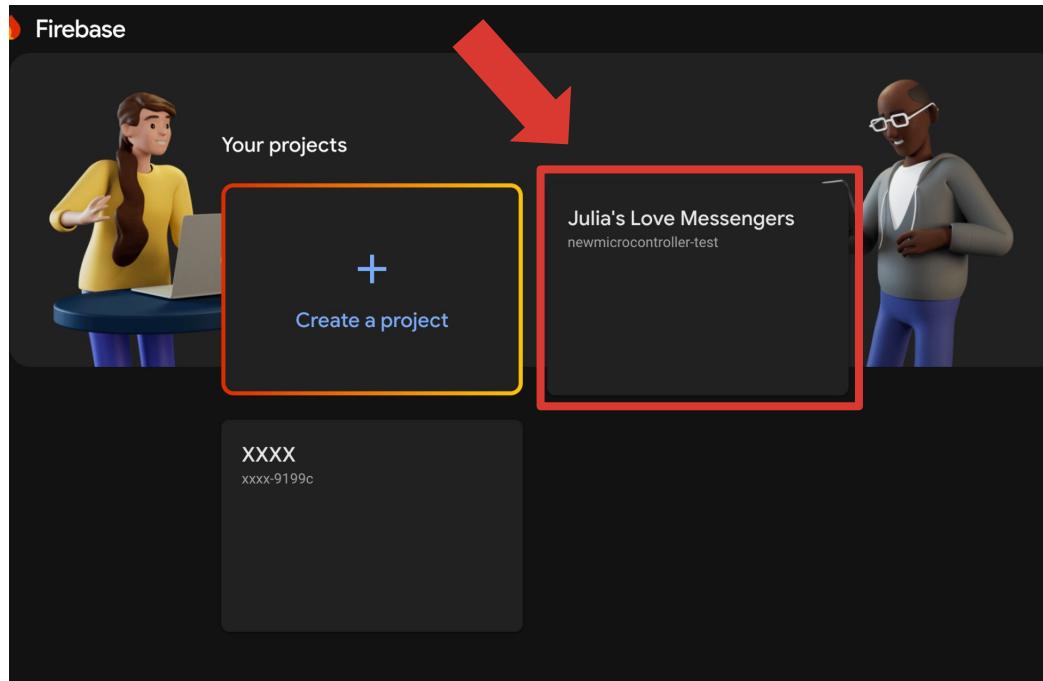
```
sketch_jan26a.ino
1 // Importing libraries
2 #include <ESP8266WiFi.h>
3 #include <Firebase_ESP_Client.h>
4
5 // Firebase requires these helpers
6 #include "addons	TokenNameHelper.h"
7 #include "addons/RTDBHelper.h"
8
9
10 // Wi-Fi credentials
11 #define WIFI_SSID "insert Wifi Name here"
12 #define WIFI_PASSWORD "insert Wifi Password here"
13
14 // Firebase credentials
15 #define API_KEY "insert API Key here"
16 #define DATABASE_URL "insert API Url here"
17
18 // Hardware components (LED & button)
19 const int ledPin = D2;
20 const int buttonPin = D5;
21 const int buttonLedPin = D6;
22
23 // Global variables
24 bool firebaseData = false;
25 int buttonState = 1;
26
27 // Define Firebase Data object and other necessary objects
```

Ln 114, Col 2 X No board selected

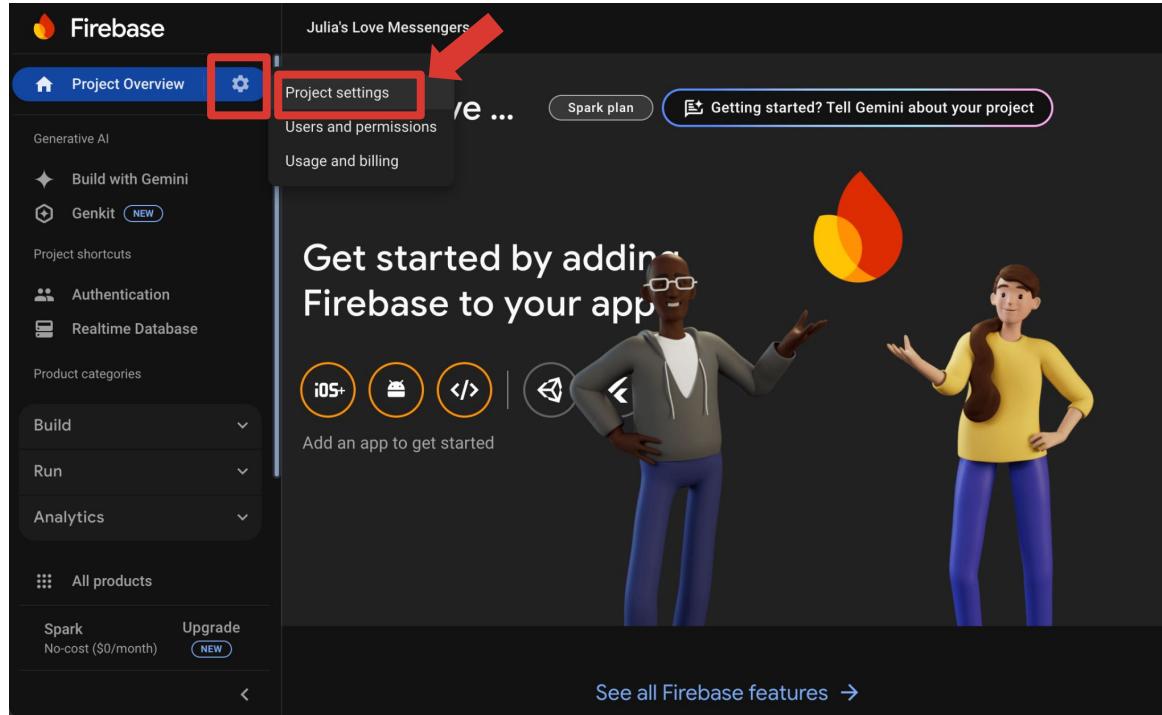
For example:

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

Step 15: 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 16: Inside your Firebase Console, select the little **gear icon**, and go to “**Project Settings**”



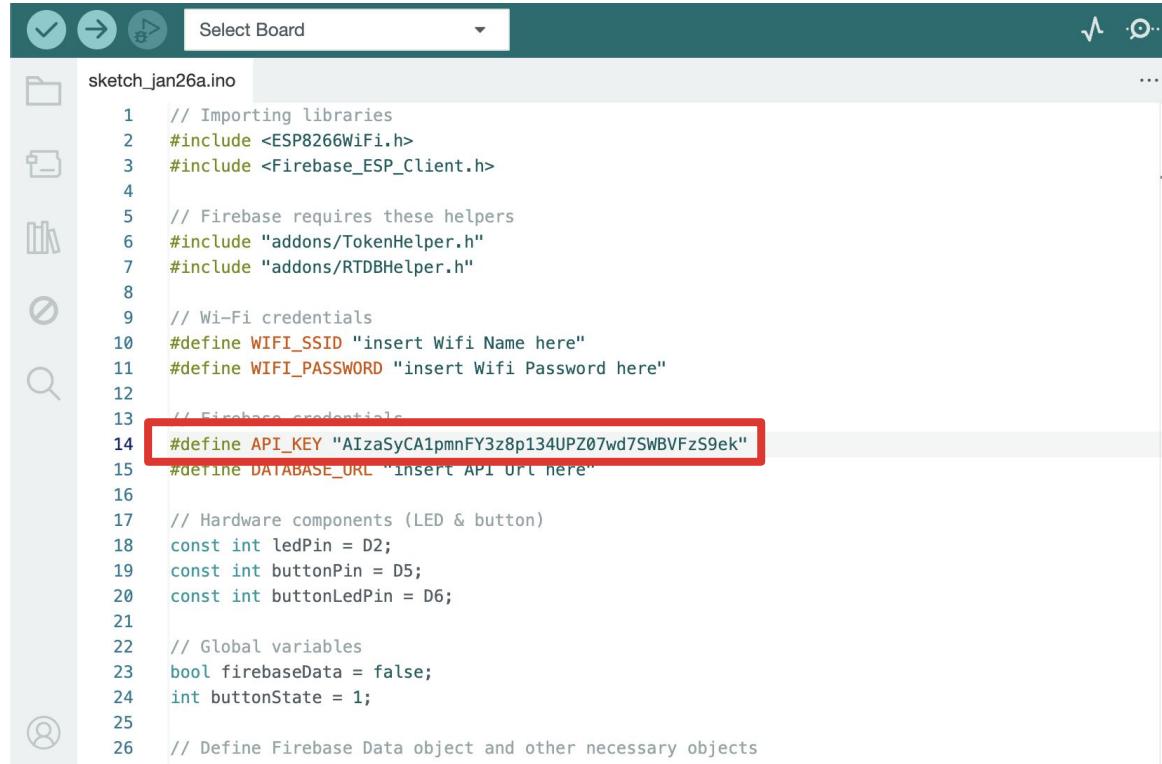
Step 17: Inside your Project Settings, you will find your API Key. Copy the API Key.

The screenshot shows the 'Project settings' page for a project named 'Julia's Love Messengers'. The 'General' tab is selected. In the 'Your project' section, there is a table with the following data:

Project name	Julia's Love Messengers
Project ID	newmicrocontroller-test
Project number	130675919122
Web API Key	AlzaSyCA1pmnFY3z8p134UPZ07wd7SWBFzS9ek

The 'Web API Key' row is highlighted with a red box. Below the table, there is a section titled 'Environment' with the subtext: 'This setting customizes your project for different stages of the app lifecycle'. The 'Environment type' is listed as 'Unspecified'.

Step 18: Head back to the Arduino code, and insert the API Key inside the double quotes after “#define API_KEY”



```
sketch_jan26a.ino
1 // Importing libraries
2 #include <ESP8266WiFi.h>
3 #include <Firebase_ESP_Client.h>
4
5 // Firebase requires these helpers
6 #include "addons	TokenName.h"
7 #include "addons/RTDBHelper.h"
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 "AIzaSyCA1pmnFY3z8p134UPZ07wd7SWBFzS9ek"
15 #define DATABASE_URL "insert API Url here"
16
17 // Hardware components (LED & button)
18 const int ledPin = D2;
19 const int buttonPin = D5;
20 const int buttonLedPin = D6;
21
22 // Global variables
23 bool firebaseData = false;
24 int buttonState = 1;
25
26 // Define Firebase Data object and other necessary objects
```

Step 19: Next, we will find your Database- URL. On Firebase, head to “**Build**” and select “**Realtime Database**”

The screenshot shows the Firebase Project settings interface. On the left, a sidebar menu is open under the “Build” tab, which is highlighted with a red box. The “Realtime Database” option is also highlighted with a red box and has a red arrow pointing towards it from the bottom right. The main content area is titled “Project settings” and shows the “General” tab selected. It displays various project details: Project name (Julia's Love Messengers), Project ID (newmicrocontroller-test), Project number (130675919122), and Web API Key (AlzaSyCA1pmnFY3z8p134UPZ07wd7SWBVFzS9ek). Below this, there is a section for “Environment” with a note: “This setting customizes your project for different stages of the app lifecycle”. The “Environment type” is listed as “Unspecified”.

Step 20: Copy your Database URL.

The screenshot shows the Firebase Realtime Database console for the project "Julia's Love Messengers". The left sidebar has a "Realtime Database" section selected. The main area displays the database structure under the URL `https://newmicrocontroller-test-default.firebaseio.com/`. The structure includes two child nodes: `user_1:1` and `user_2:1`.

Firebase

Julia's Love Messengers ▾

Project Overview | ⚙️

Generative AI

◆ Build with Gemini

⬢ Genkit NEW

Project shortcuts

👤 Authentication

💻 Realtime Database

Product categories

Build

Run

Analytics

All products

Spark No-cost (\$0/month) Upgrade NEW

Realtime Database Need help with Realtime Database? Ask Gemini

Data Rules Backups Usage Extensions

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

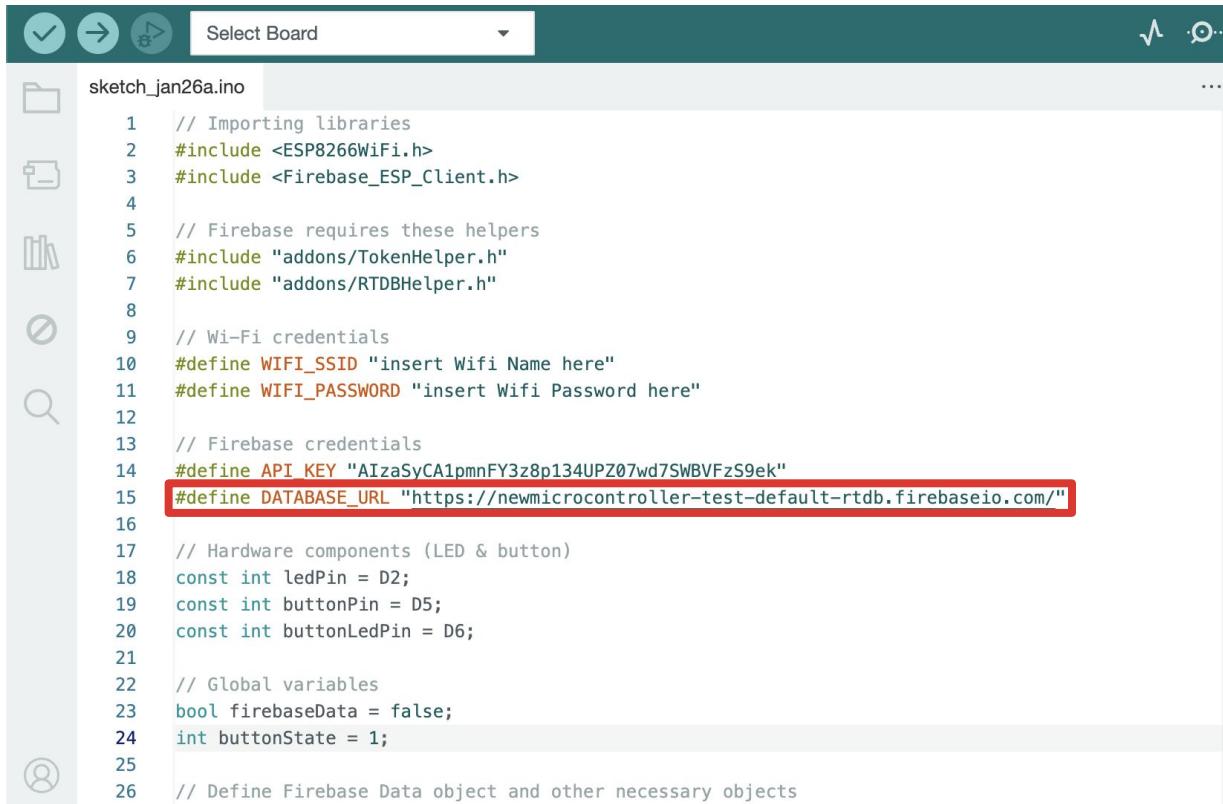
https://newmicrocontroller-test-default.firebaseio.com/

user_1:1

user_2:1

Database location: United States (us-central1)

Step 21: Paste the Database URL into the Arduino Code into the double quotes after "#define DATABASE_URL"

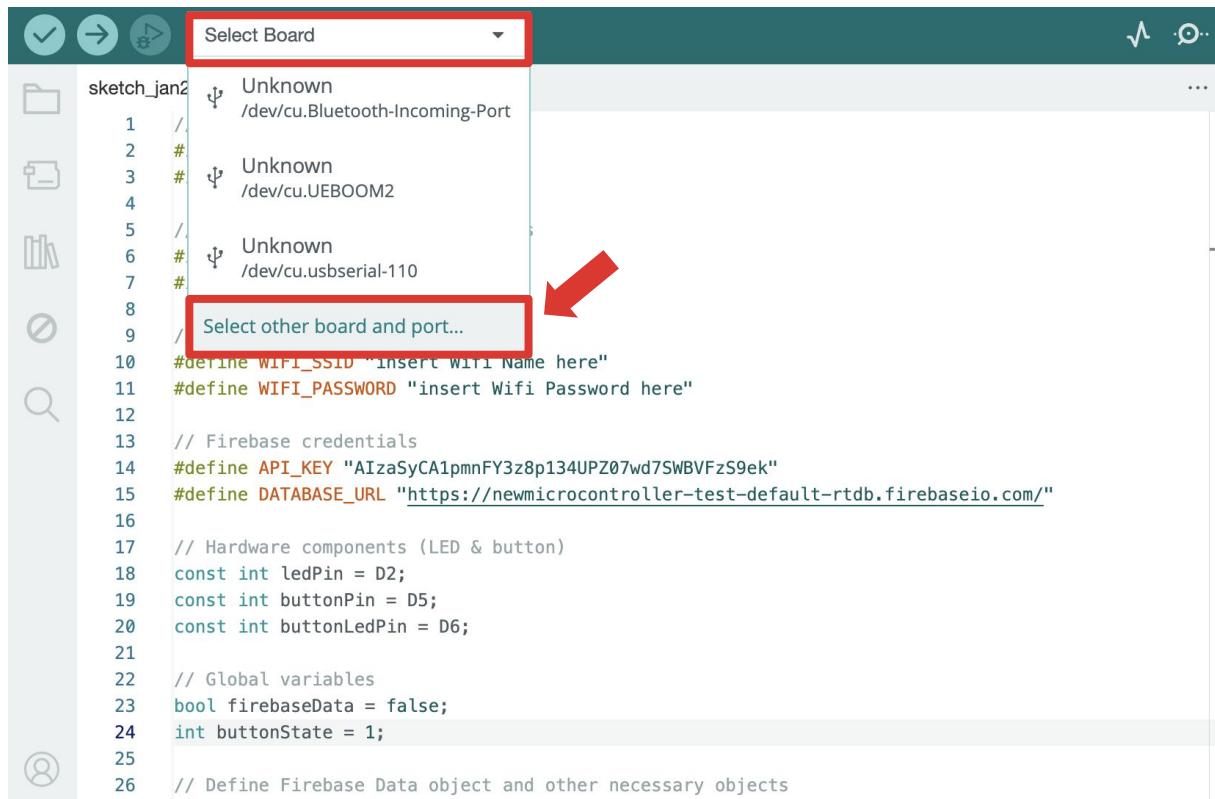


The screenshot shows the Arduino IDE interface with the following details:

- Toolbar:** Includes icons for file operations (checkmark, arrow, folder), a "Select Board" dropdown, and other standard IDE tools.
- Sketch Name:** The current sketch is named "sketch_jan26a.ino".
- Code Editor:** Displays the Arduino sketch code. The line containing the database URL is highlighted with a red rectangle.
- Code Content:**

```
1 // Importing libraries
2 #include <ESP8266WiFi.h>
3 #include <Firebase_ESP_Client.h>
4
5 // Firebase requires these helpers
6 #include "addons/TokenHelper.h"
7 #include "addons/RTDBHelper.h"
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 "AIzaSyCA1pmnFY3z8p134UPZ07wd7SWBVFzS9ek"
15 #define DATABASE_URL "https://newmicrocontroller-test-default.firebaseio.com/"
16
17 // Hardware components (LED & button)
18 const int ledPin = D2;
19 const int buttonPin = D5;
20 const int buttonLedPin = D6;
21
22 // Global variables
23 bool firebaseData = false;
24 int buttonState = 1;
25
26 // Define Firebase Data object and other necessary objects
```

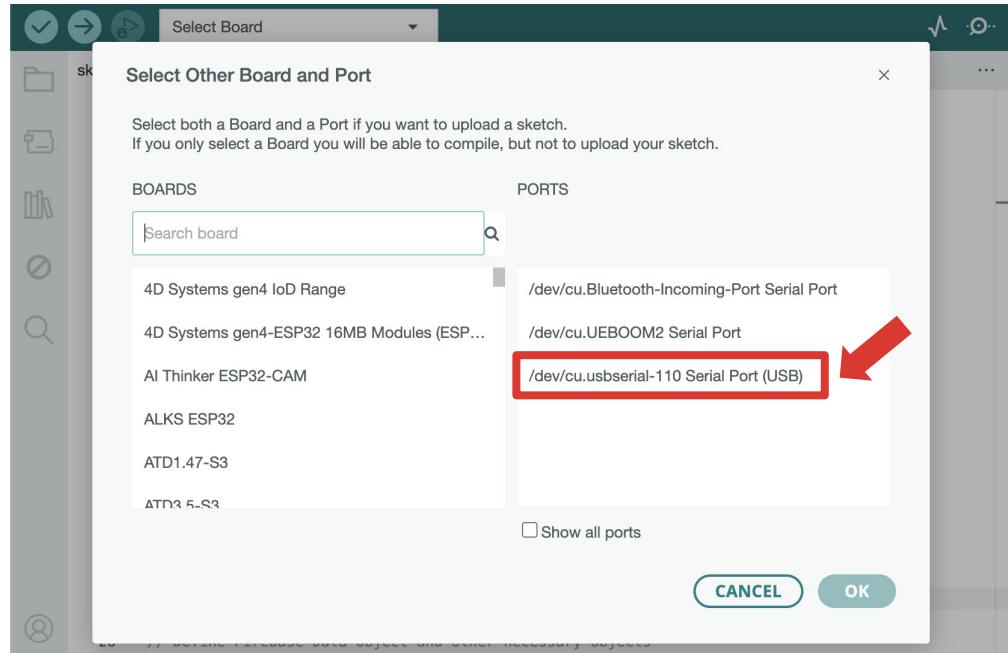
Step 22: The code is now ready to be uploaded to the first Love Messenger.
Go to “**Select Board**”, and go to “**Select other board and port...**”



Step 23: Plug in your first Love Messenger to your Computer using a data-transfer USB cable.

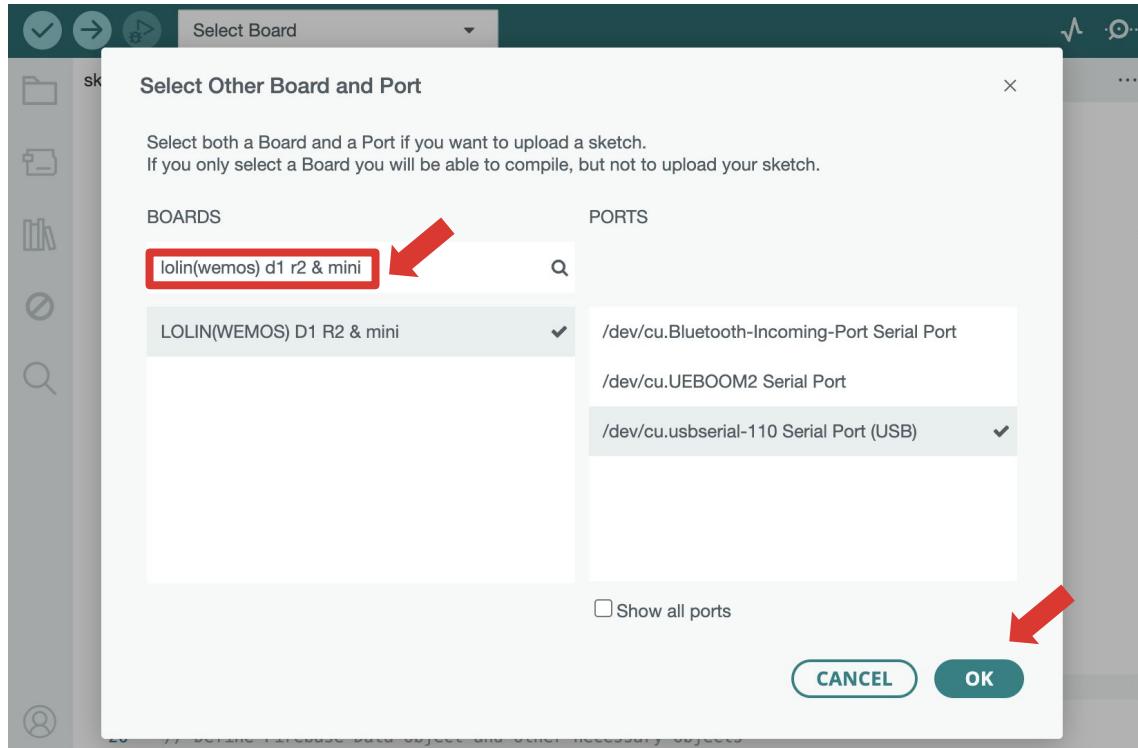
Once the Love Messenger is plugged in, a new port should appear in the “Ports” section. Depending on if you are using a Mac or PC, the port names can vary (“COM” for PC, and “usbserial” for Mac).

Select the new port.

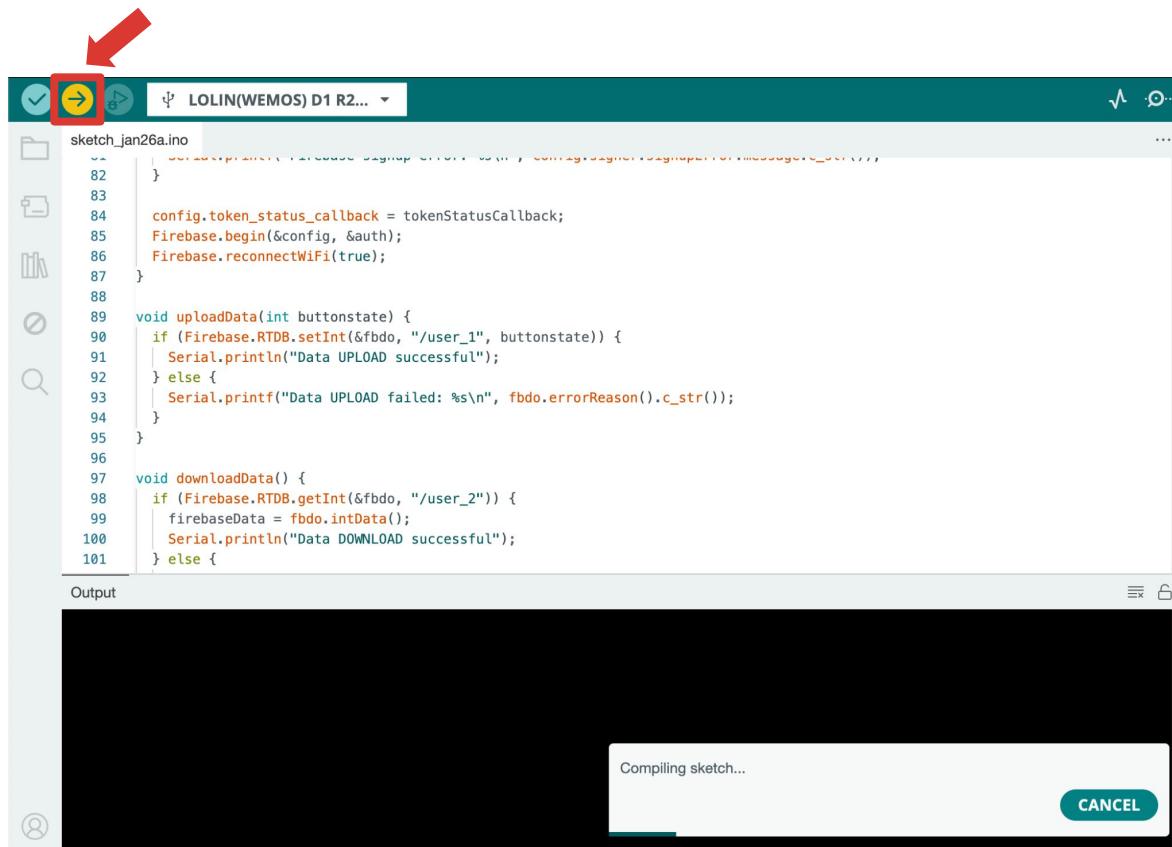


**If your port does not show up, you might be using a charge-only USB wire that doesn't transmit data. You need to use a USB wire that also can do data-transfer. We have the ones we used in the workshop linked on Github.*

Step 24: In the “**Boards**” section, search for “**lolin(wemos) d1 r2 & mini**”, and select it. Then Press OK



Step 25: 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.



Step 26: The Sketch will first compile and then upload.

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

The screenshot shows the Arduino IDE interface. The top bar displays the board as "LOLIN(WEMOS) D1 R2...". The main area shows the code for "sketch_jan26a.ino". The code includes functions for token status callback, uploading data based on button state, and downloading data from Firebase RTDB. The "Output" window at the bottom shows the progress of the upload process, including writing to memory and a final message indicating the chip is being reset via the RTS pin.

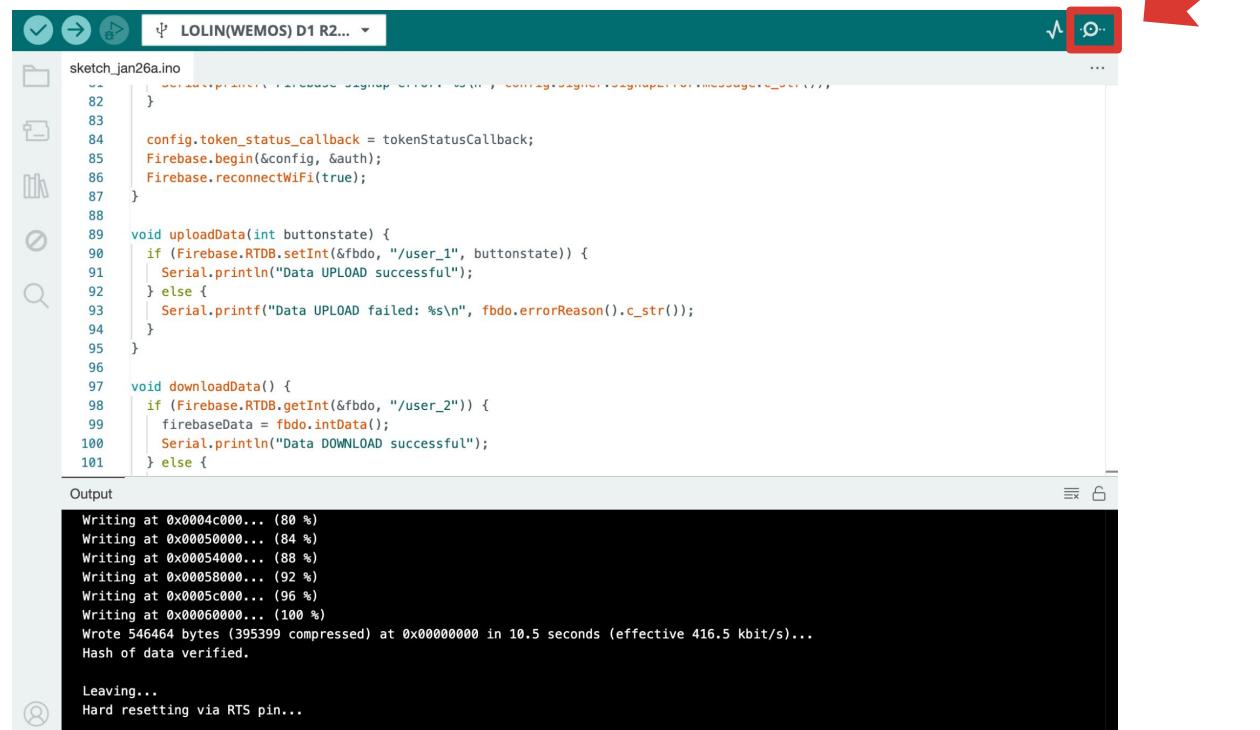
```
sketch_jan26a.ino
82 }
83 config.token_status_callback = tokenStatusCallback;
84 Firebase.begin(&config, &auth);
85 Firebase.reconnectWiFi(true);
86 }
87 }
88 void uploadData(int buttonstate) {
89   if (Firebase.RTDB.setInt(&fbdo, "/user_1", buttonstate)) {
90     Serial.println("Data UPLOAD successful");
91   } else {
92     Serial.printf("Data UPLOAD failed: %s\n", fbdo.errorReason().c_str());
93   }
94 }
95 }
96 void downloadData() {
97   if (Firebase.RTDB.getInt(&fbdo, "/user_2")) {
98     firebaseData = fbdo.intData();
99     Serial.println("Data DOWNLOAD successful");
100 } else {
101 }
```

Output

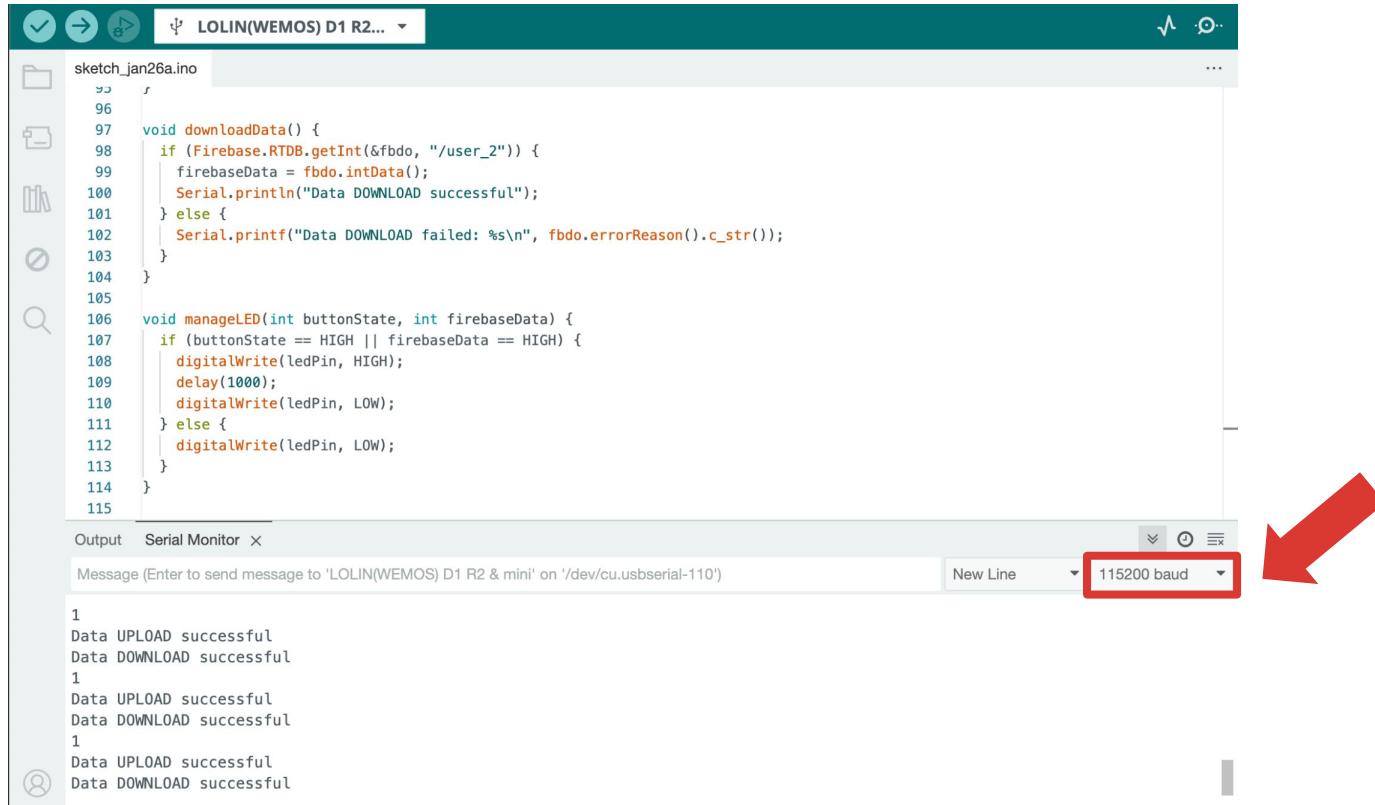
```
Writing at 0x0004c000... (80 %)
Writing at 0x00050000... (84 %)
Writing at 0x00054000... (88 %)
Writing at 0x00058000... (92 %)
Writing at 0x0005c000... (96 %)
Writing at 0x00060000... (100 %)
Wrote 546464 bytes (395399 compressed) at 0x0000 in 10.5 seconds (effective 416.5 kbit/s)...
Hash of data verified.

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

Step 27: To check if our code works, open the Serial Monitor.



Step 28: Change the baud rate to **115200 baud**.



The screenshot shows the Arduino IDE interface. The top bar displays the connection information: "LOLIN(WEMOS) D1 R2...". The left sidebar shows the file structure for "sketch_jan26a.ino". The main code area contains two functions: `downloadData()` and `manageLED()`. The `downloadData()` function checks if a value in the Firebase database at `/user_2` is 1, sets `firebaseData` to 1, prints "Data DOWNLOAD successful", and handles errors. The `manageLED()` function toggles an LED based on button state and firebase data. The bottom section shows the Serial Monitor window with the message input field containing "Message (Enter to send message to 'LOLIN(WEMOS) D1 R2 & mini' on '/dev/cu.usbserial-110')". To the right of the input field is a dropdown menu set to "115200 baud". A red arrow points to this dropdown menu, indicating it needs to be changed from its current setting of 110.

```
sketch_jan26a.ino
96
97 void downloadData() {
98     if (Firebase.RTDB.getInt(&fbdo, "/user_2")) {
99         firebaseData = fbdo.intData();
100        Serial.println("Data DOWNLOAD successful");
101    } else {
102        Serial.printf("Data DOWNLOAD failed: %s\n", fbdo.errorReason().c_str());
103    }
104 }
105
106 void manageLED(int buttonState, int firebaseData) {
107     if (buttonState == HIGH || firebaseData == HIGH) {
108         digitalWrite(ledPin, HIGH);
109         delay(1000);
110         digitalWrite(ledPin, LOW);
111     } else {
112         digitalWrite(ledPin, LOW);
113     }
114 }
```

Output Serial Monitor x

Message (Enter to send message to 'LOLIN(WEMOS) D1 R2 & mini' on '/dev/cu.usbserial-110')

New Line 115200 baud

```
1
Data UPLOAD successful
Data DOWNLOAD successful
1
Data UPLOAD successful
Data DOWNLOAD successful
1
Data UPLOAD successful
Data DOWNLOAD successful
```

Step 29: 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.

The screenshot shows the Arduino Serial Monitor window titled "Serial Monitor". It has tabs for "Output" and "Serial Monitor". Below the tabs is a message input field: "Message (Enter to send message to 'ESP32 Dev Module' on '/dev/cu.usbserial-0001')". The main area displays a log of messages from the ESP32. A red arrow points from the text "The Love Messenger will first connect to Wifi..." to the log entries for connecting to WiFi. A blue arrow points from the text "The Wifi is connected" to the message "Connected with IP: 172.22.52.167". A green arrow points from the text "Once the Serial Monitor prints this, we can move on to the second Love Messenger. Everything is set up." to the message "Firebase successful". The log entries are as follows:

```
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 30: Now, unplug your first Love Messenger, and plug in your second Love Messenger.

Step 31: In your code, in the **uploadData** function, change “/user_1” to “/user_2”

```
89 void uploadData(int buttonstate) {  
90     if (Firebase.RTDB.setInt(&fbdo, "/user_1", buttonstate)) {  
91         Serial.println("Data UPLOAD successful");  
92     } else {  
93         Serial.printf("Data UPLOAD failed: %s\n", fbdo.errorReason().c_str());  
94     }  
95 }
```

Change to

```
89 void uploadData(int buttonstate) {  
90     if (Firebase.RTDB.setInt(&fbdo, "/user_2", buttonstate)) {  
91         Serial.println("Data UPLOAD successful");  
92     } else {  
93         Serial.printf("Data UPLOAD failed: %s\n", fbdo.errorReason().c_str());  
94     }  
95 }
```

Step 32: Similarly, in your code, in the **download data** function, change “/user_2” to “/user_1”

```
97  void downloadData() {  
98    if (Firebase.RTDB.getInt(&fbdo, "/user_2")) {  
99      firebaseData = fbdo.intData();  
100     Serial.println("Data DOWNLOAD successful");  
101    } else {  
102      Serial.printf("Data DOWNLOAD failed: %s\n", fbdo.errorReason().c_str());  
103    }  
104 }
```

Change to

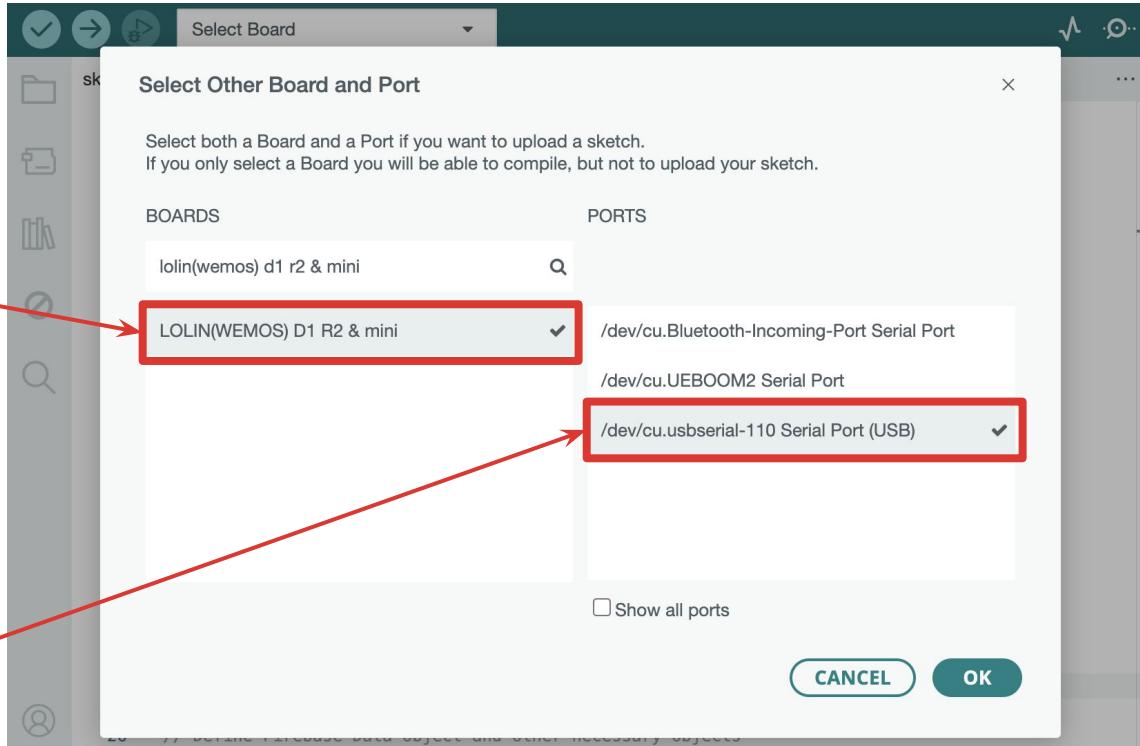
```
97  void downloadData() {  
98    if (Firebase.RTDB.getInt(&fbdo, "/user_1")) {  
99      firebaseData = fbdo.intData();  
100     Serial.println("Data DOWNLOAD successful");  
101    } else {  
102      Serial.printf("Data DOWNLOAD failed: %s\n", fbdo.errorReason().c_str());  
103    }  
104 }
```

Step 33:

Once again, under “Select Board”, you should have

“lolin(wemos) d1 r2 & mini” 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 34: Now, upload the code to the second Love Messenger!



A screenshot of the Arduino IDE interface. At the top, there's a toolbar with icons for file operations, a yellow circular button, and a dropdown menu showing 'LOLIN(WEMOS) D1 R2...'. Below the toolbar is a code editor window titled 'sketch_jan26a.ino' containing C++ code for interacting with a Firebase database. The code includes functions for token status callback, uploading data based on button state, and downloading data. In the bottom right corner of the code editor, the text 'Compiling sketch...' is visible. On the far left of the IDE, there's a vertical sidebar with icons for file, folder, book, and search.

```
sketch_jan26a.ino
81     if (Firebase.RTDB.setInt(&fbdo, "/user_1", buttonstate)) {
82     | Serial.println("Data UPLOAD successful");
83   } else {
84     | Serial.printf("Data UPLOAD failed: %s\n", fbdo.errorReason().c_str());
85   }
86 }
87
88 void uploadData(int buttonstate) {
89   if (Firebase.RTDB.setInt(&fbdo, "/user_1", buttonstate)) {
90     | Serial.println("Data UPLOAD successful");
91   } else {
92     | Serial.printf("Data UPLOAD failed: %s\n", fbdo.errorReason().c_str());
93   }
94 }
95
96 void downloadData() {
97   if (Firebase.RTDB.getInt(&fbdo, "/user_2")) {
98     firebaseData = fbdo.intData();
99     Serial.println("Data DOWNLOAD successful");
100  } else {
101    }
```

Output

Compiling sketch...

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

Step 35: 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 35: 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 36: 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!

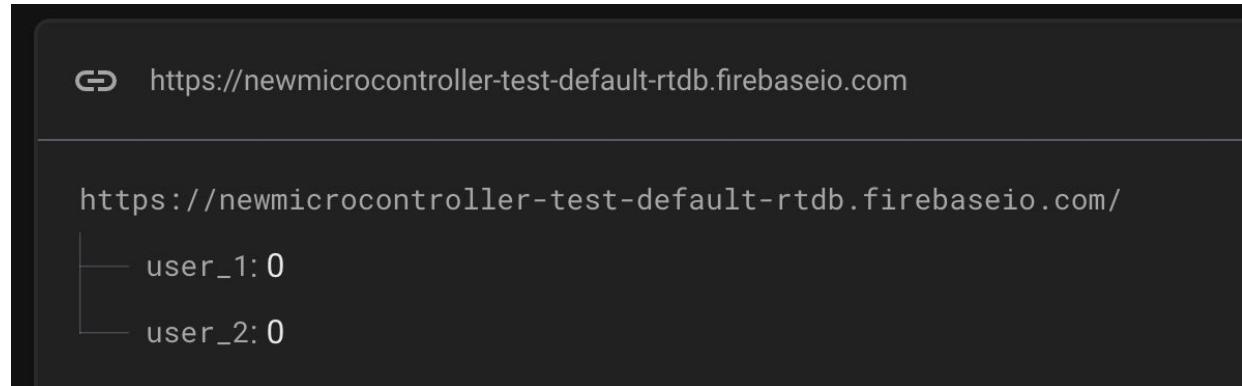


Tips:

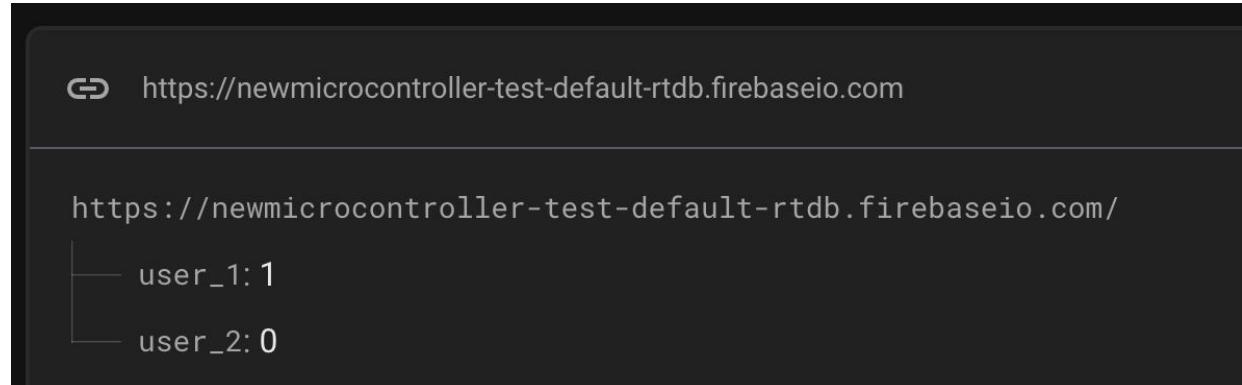
1. Be patient: press the button for a few seconds until you see your Love Messenger light up
2. Unplugging and replugging the Love Messenger can help too!

Step 37: While the Love Messengers are working, you should be able to see Firebase updating in real time.

Default Firebase states



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



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! ❤️