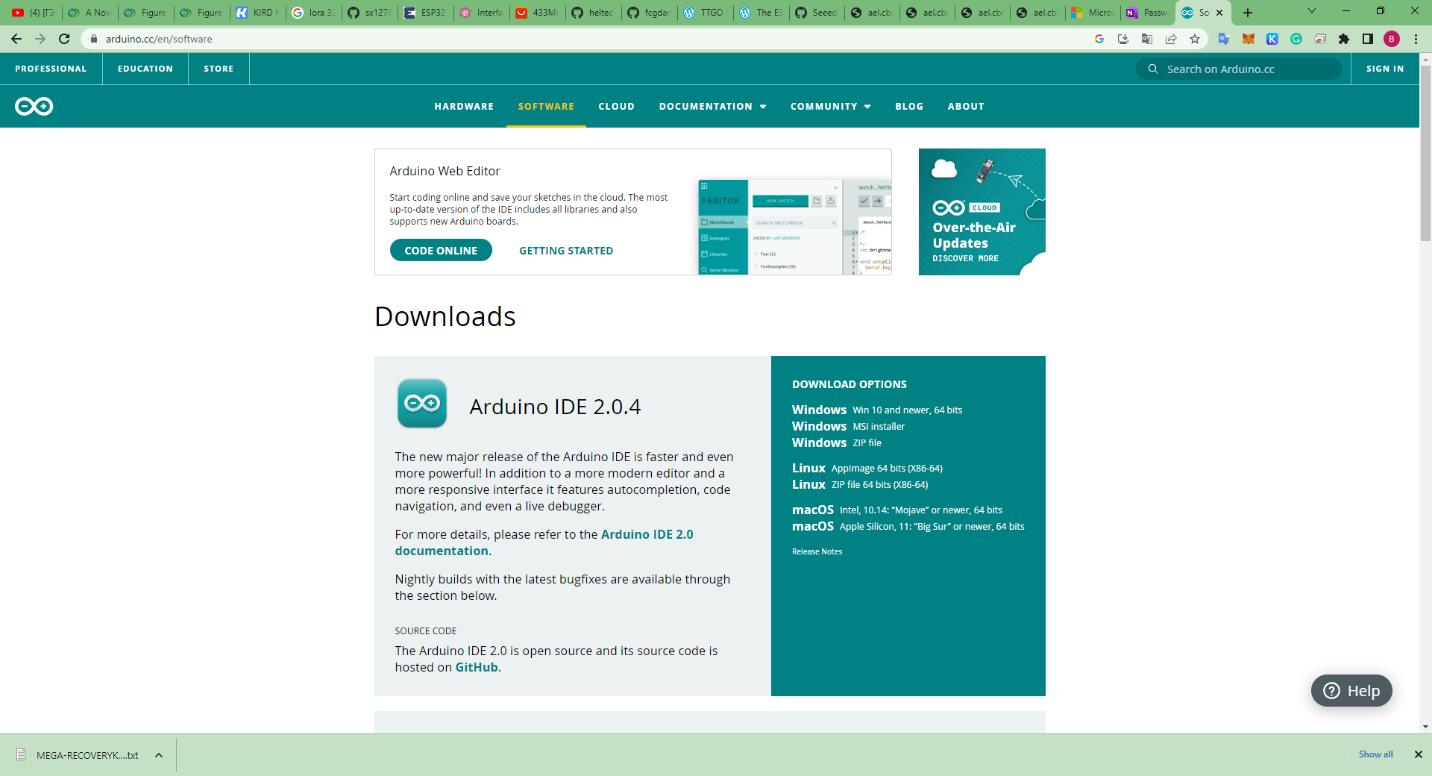
Connection esp32 to an esp32 via ESP\_NOW protocol

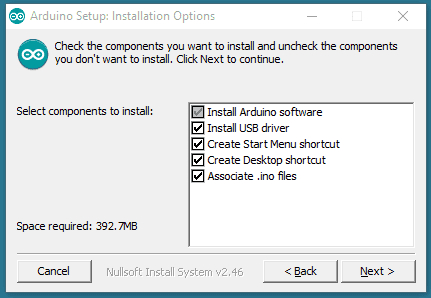
Download the Arduino Software (IDE)



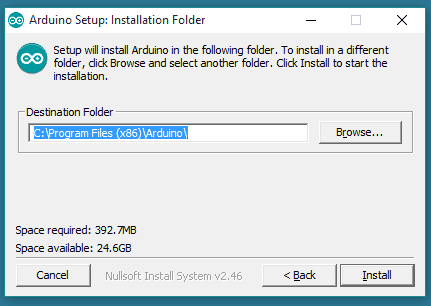
https://www.arduino.cc/en/software

Download the Arduino IDE that is compatible with your computer system.

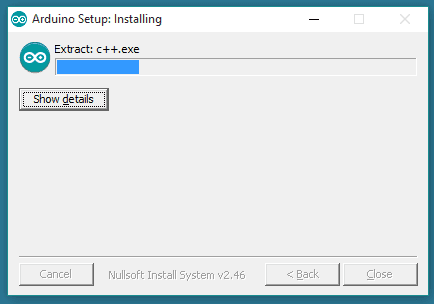
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



Choose the components to install.



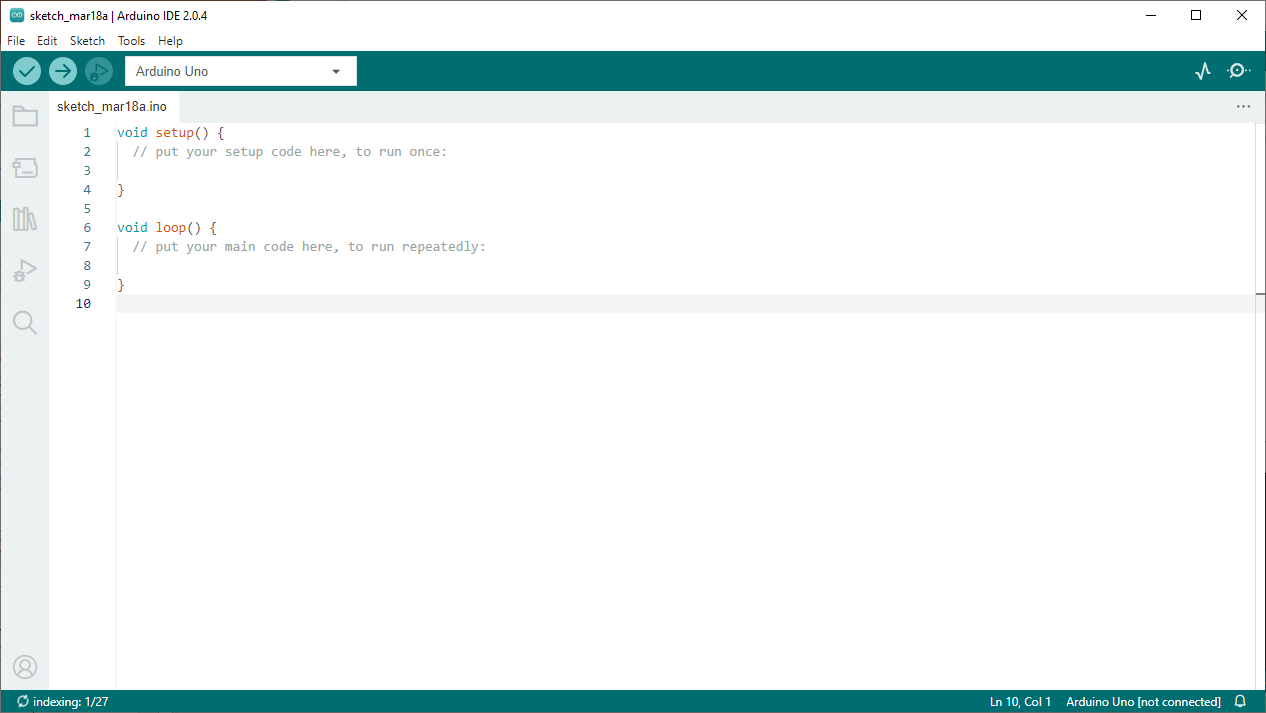
Choose the installation directory.



Installation in progress.

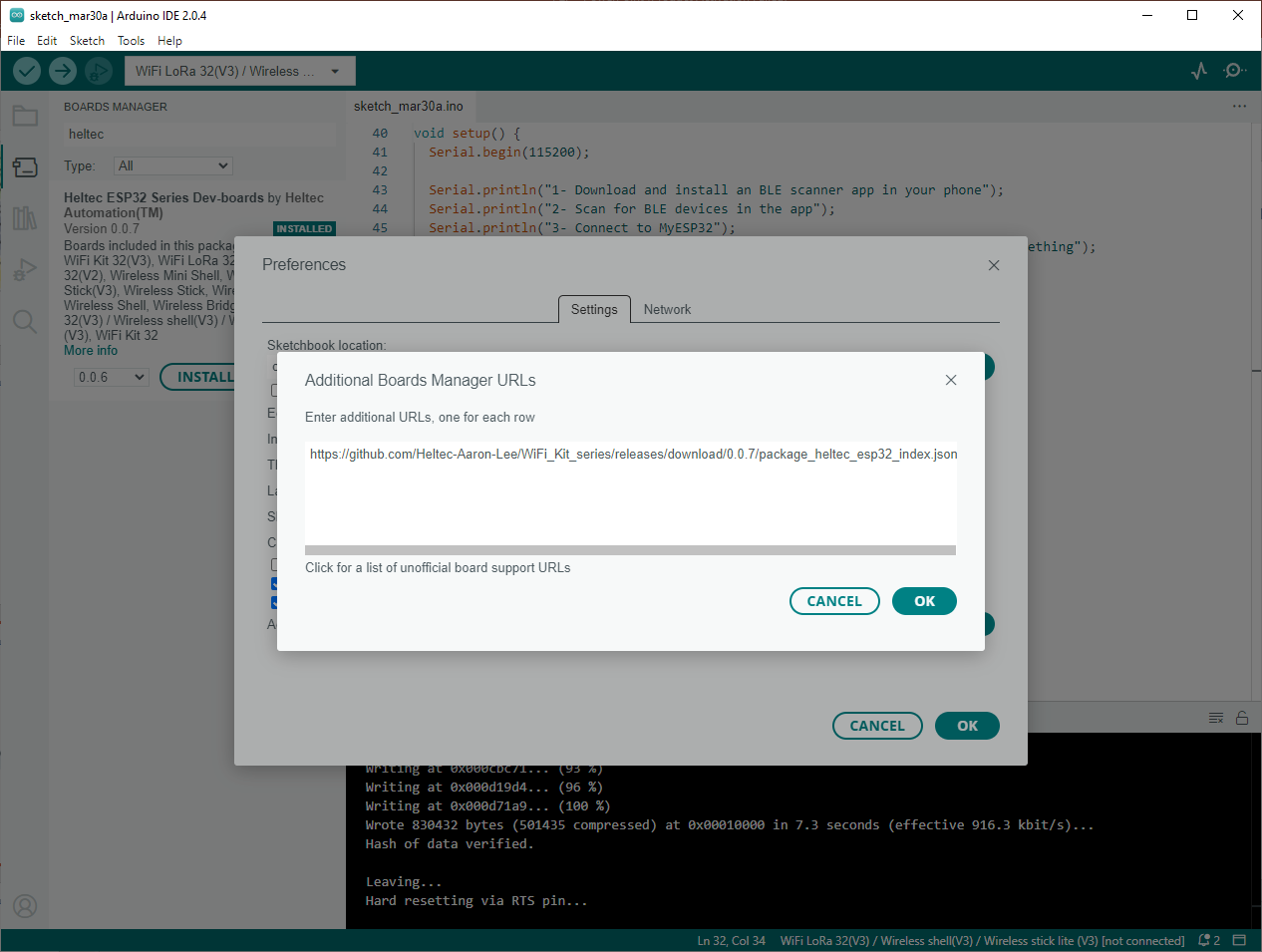
The process will extract and install all the required files to execute properly the Arduino Software (IDE)

Open the Arduino IDE.

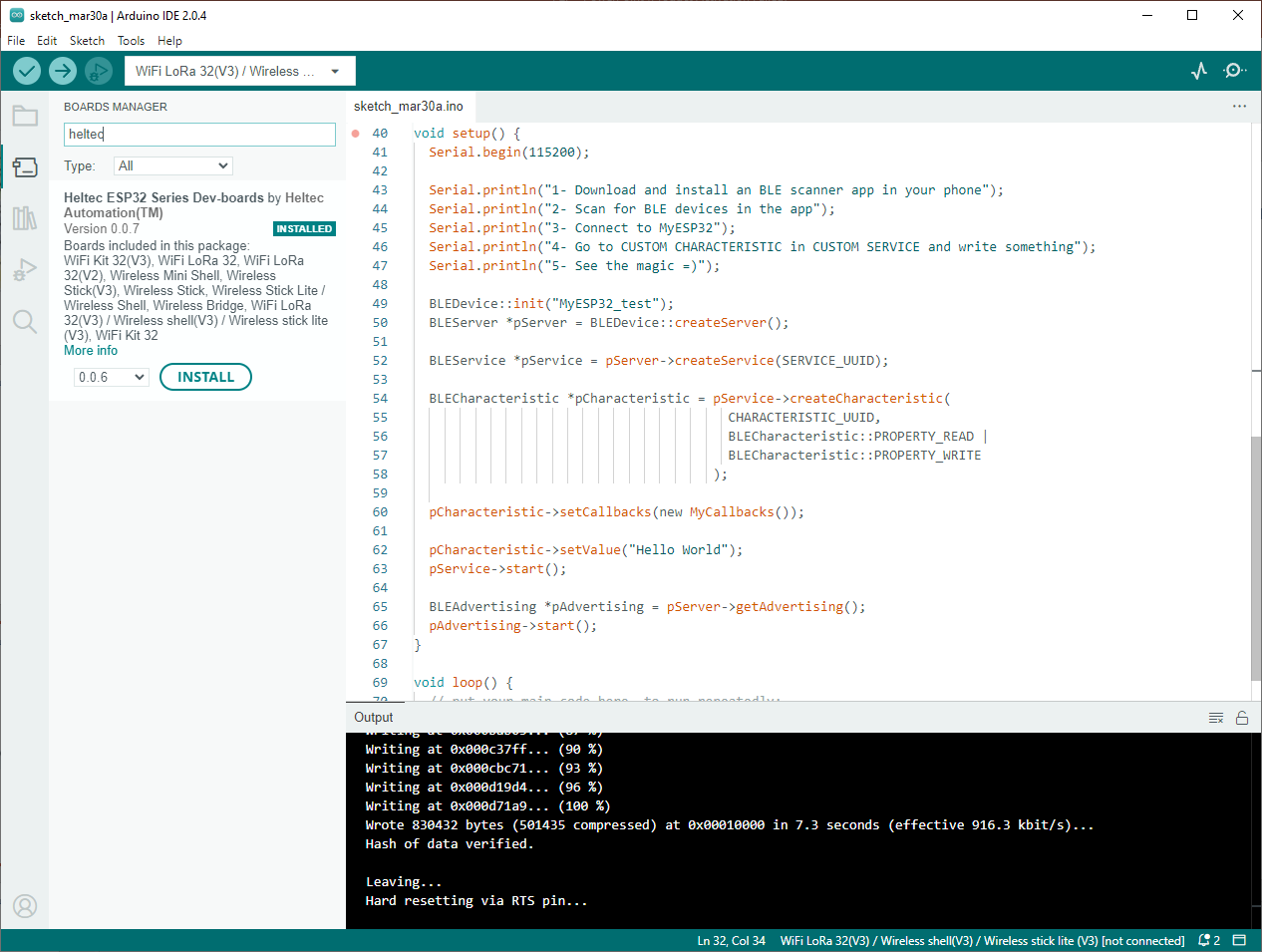


File->Preferences->Additional boards manager URLs

https://github.com/Heltec-Aaron-Lee/WiFi\_Kit\_series/releases/download/0.0.7/package\_heltec\_esp32\_index.json

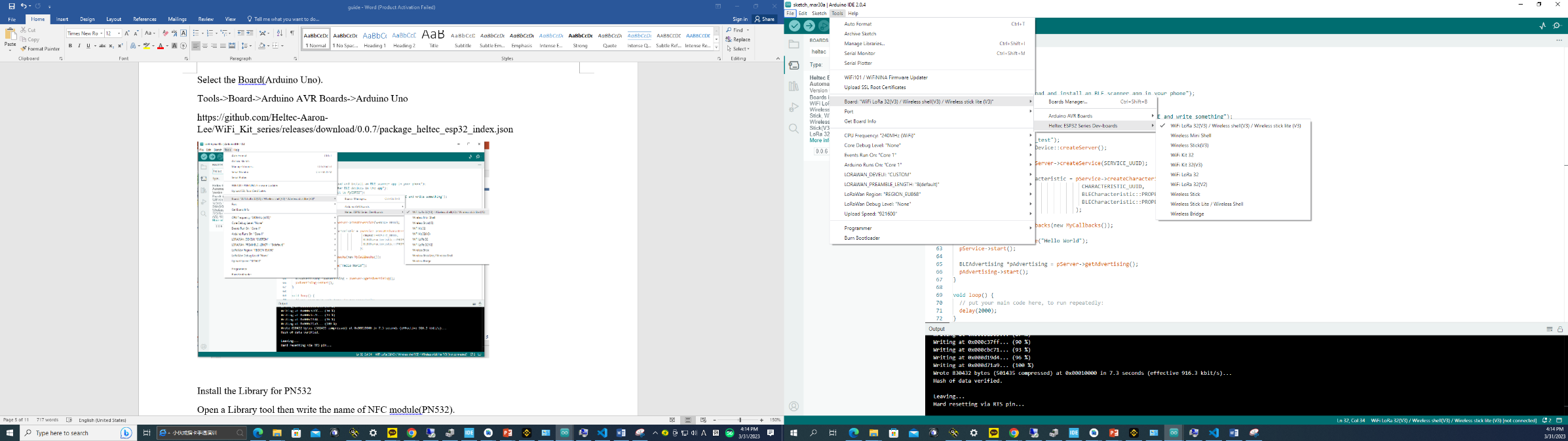


Install the Heltec ESP32 series Dev-boards

Tools->Board->board manager

Select the Board(Arduino Uno).

Tools->Board->Heltec ESP32->Wifi Lora 32(V3)

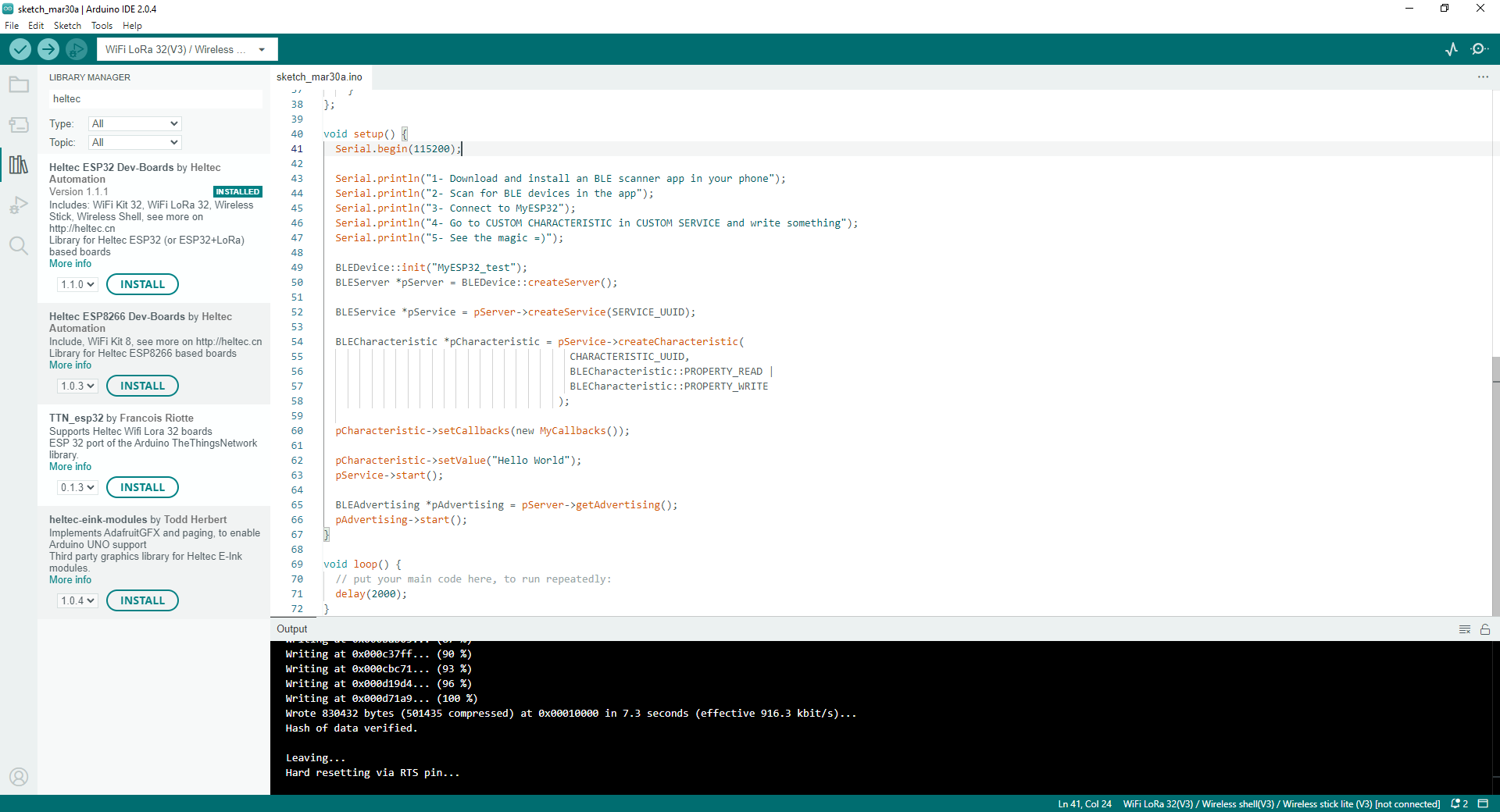


Install the Library for Heltec ESP32

Open a Library tool then write the name of ESP32 module(Heltec).

Then Install the Heltec ESP32 Dev-Boards library.

Copy and paste the below code here.

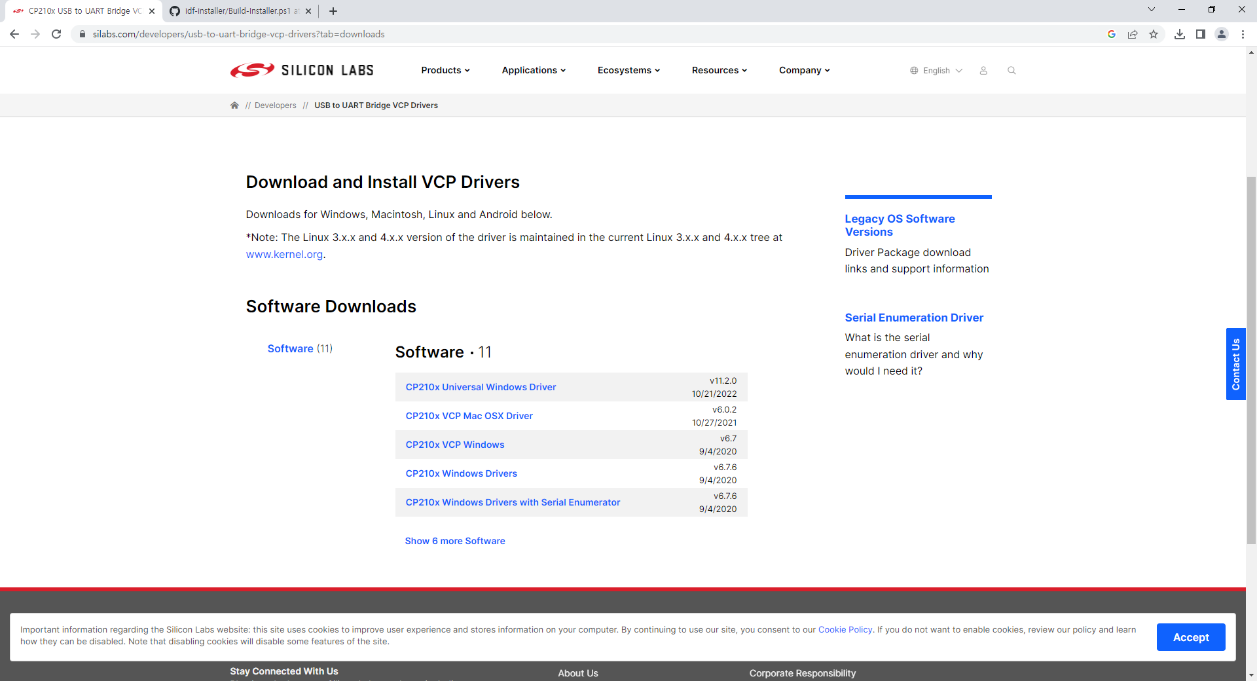


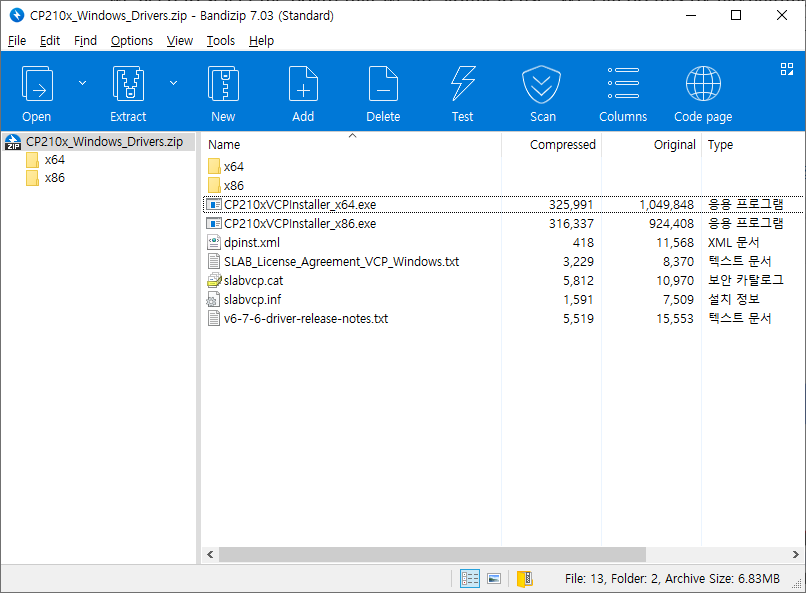
we need to select the board that we are going to use. We can do this by navigating to **Tools > Port > {Board}**. The board(s) that are connected to your computer should appear here, and we need to select it by clicking it. In this case, our board is displayed as **COM9**.

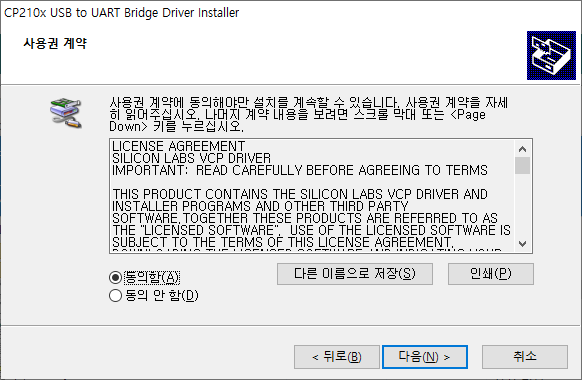
If you have connected your board to your computer but there is no com port

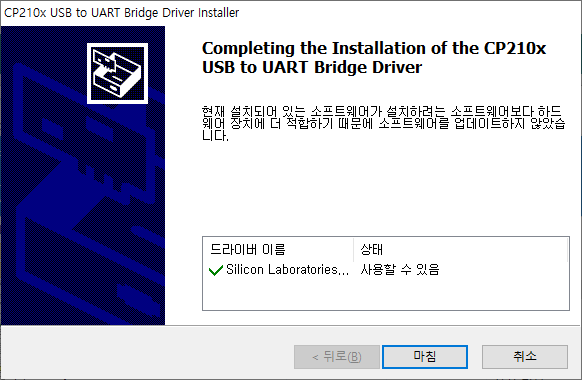
Download esp32 board usb driver and install your usb driver

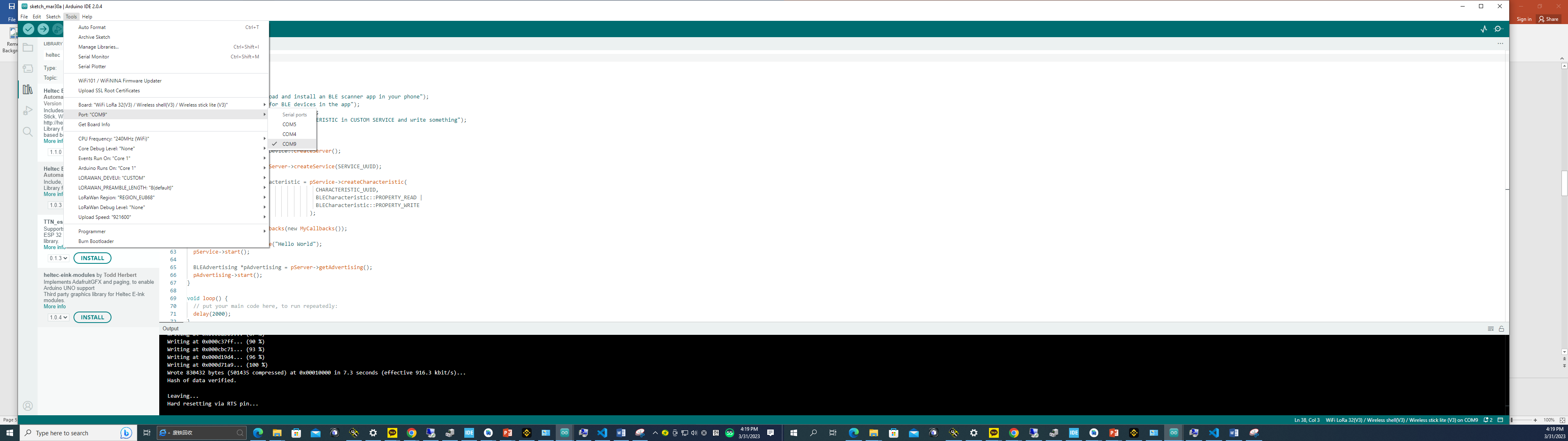
<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads>



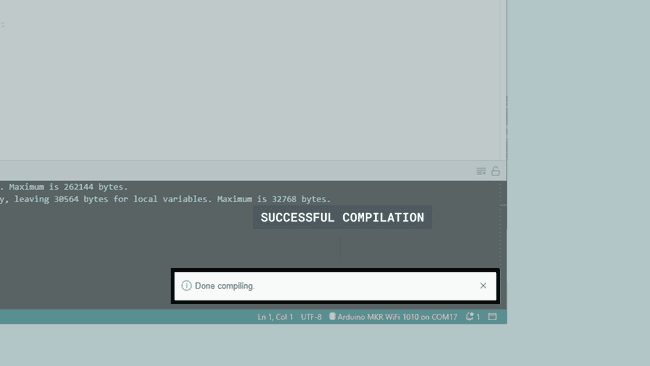






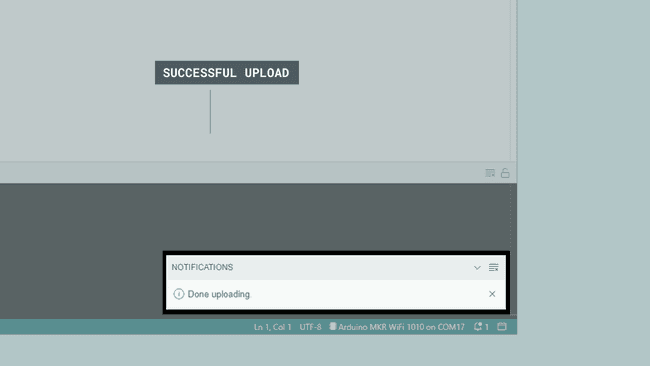


Click on the verify tool (checkmark). Since we are verifying an empty sketch, we can be sure it is going to compile. After a few seconds, we can see the result of the action in the console (black box in the bottom).



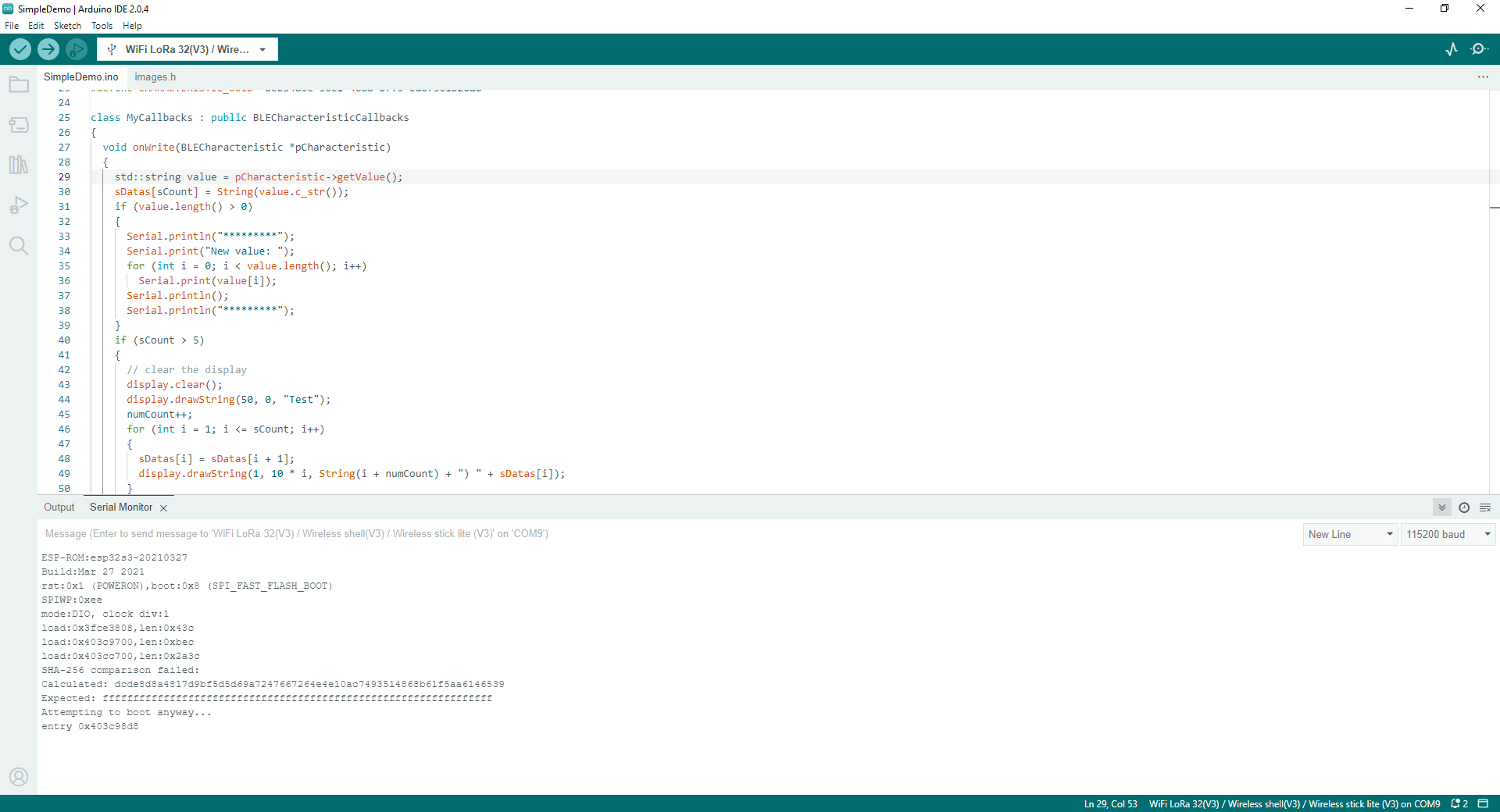
 With the board selected, we are good to go! Click on the **upload** button, and it will start uploading the sketch to the board.

When it is finished, a notification pops up in the bottom right of your IDE window. Of course, sometimes there are some complications when uploading, and these errors will be listed here as well.



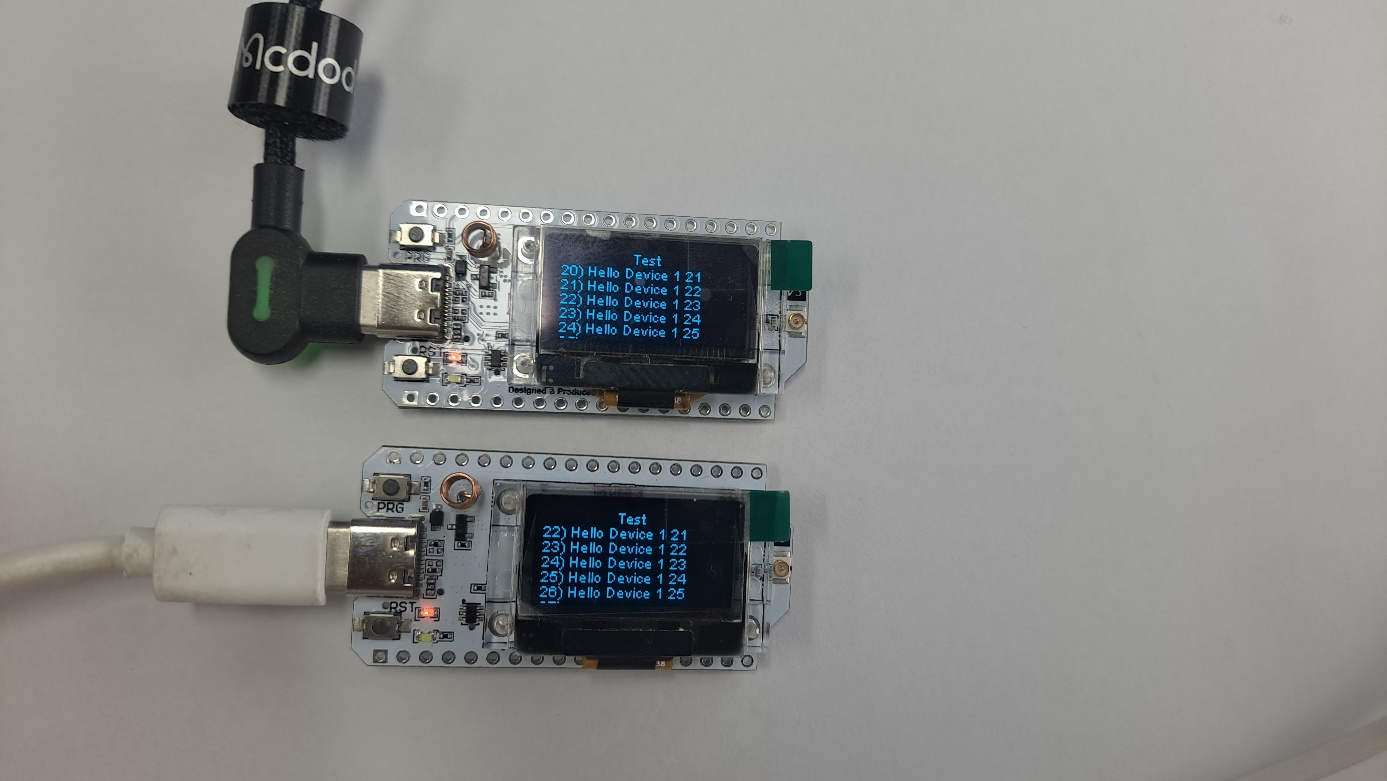
Congratulations, you have now uploaded a sketch to your Arduino board!

When it has finished uploading, click on the Serial Monitor button, located at the top right corner of the IDE. This will launch the Serial Monitor in the bottom of the IDE, replacing the console section.



**Run the following codes on each of the 2 boards according to the above instructions**

oled display



Source code.

#include <Arduino.h>

#include <WiFi.h>

#include <esp\_wifi.h>

#include <QuickEspNow.h>    // library for easy setup of ESP-NOW protocol

#include "HT\_SSD1306Wire.h" // library for the OLED display

#include <stdlib.h>

#include <string>

// Creating an object of the OLED display library

SSD1306Wire display(0x3c, 500000, SDA\_OLED, SCL\_OLED, GEOMETRY\_128\_64, RST\_OLED);

// Parameters: OLED I2C address, I2C frequency, I2C pin group, screen resolution, and reset pin

// Defining two integers and a string array

int sCount = 1, numCount = 0; // counters for messages and for keeping track of the number of messages displayed on the OLED screen

String sDatas[8];             // string array to store messages received via ESP-NOW

// Creating a string message to be sent via ESP-NOW and defining the receiver's MAC address

static const String msg = "Hello Device 1";

static uint8\_t receiver[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

#define DEST\_ADDR receiver // defining the receiver's MAC address

// #define DEST\_ADDR ESPNOW\_BROADCAST\_ADDRESS // use this instead of the above line to send messages to all devices on the network

const unsigned int SEND\_MSG\_MSEC = 2000; // delay between sending messages via ESP-NOW

// Function to handle received data via ESP-NOW

void dataReceived(uint8\_t \*address, uint8\_t \*data, uint8\_t len, signed int rssi, bool broadcast)

{

  // printing received message and details to the serial monitor

  Serial.print("Received: ");

  Serial.printf("%.\*s\n", len, data);

  Serial.printf("RSSI: %d dBm\n", rssi);

  Serial.printf("From: " MACSTR "\n", MAC2STR(address));

  Serial.printf("%s\n", broadcast ? "Broadcast" : "Unicast");

  // Converting the received message to a string and storing it in an array

  String dataString;

  for (int i = 0; i < len; i++)

  {

    dataString += (char)data[i];

  }

  Serial.println(dataString);

  sDatas[sCount] = dataString;

  // Checking if the maximum number of messages to be displayed on the OLED screen has been reached

  if (sCount > 5)

  {

    // Clearing the OLED display and displaying a message

    display.clear();

    display.drawString(50, 0, "Test");

    numCount++;

    // Shifting the messages in the array and displaying them on the OLED screen

    for (int i = 1; i <= sCount; i++)

    {

      sDatas[i] = sDatas[i + 1];

      display.drawString(1, 10 \* i, String(i + numCount) + ") " + sDatas[i]);

    }

    sCount = 5;

  }

  else

  {

    // Displaying the received message on the OLED screen

    display.drawString(1, 10 \* sCount, String(sCount) + ") " + sDatas[sCount]);

  }

  sCount++;

  // Updating the OLED screen

  display.display();

}

void setup()

{

  // Initializing the OLED display

  display.init();

  display.setFont(ArialMT\_Plain\_10);

  display.setTextAlignment(TEXT\_ALIGN\_LEFT);

  // Initialize display and display "Test"

  display.drawString(50, 0, "Test");

  display.display();

  // Initialize Serial communication

  Serial.begin(115200);

  // Initialize Wi-Fi in station mode and set MAC address

  WiFi.mode(WIFI\_MODE\_STA);

  uint8\_t mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

  // Replace the MAC address above with the MAC address you want to set.

  esp\_wifi\_set\_mac(WIFI\_IF\_STA, mac);

// Disconnect from Wi-Fi and print connection details

#if defined ESP32

  WiFi.disconnect(false, true);

#elif defined ESP8266

  WiFi.disconnect(false);

#endif // ESP32

  quickEspNow.onDataRcvd(dataReceived);

// Set Wi-Fi bandwidth if using ESP32

#ifdef ESP32

  quickEspNow.setWiFiBandwidth(WIFI\_IF\_STA, WIFI\_BW\_HT20); // Only needed for ESP32 in case you need coexistence with ESP8266 in the same network

#endif                                                     // ESP32

  // Start QuickESPNow instance

  quickEspNow.begin(1); // If you use no connected WiFi channel should be specified

}

void loop()

{

  static unsigned int counter = 0;

  // Create message to send

  String message = String(msg) + " " + String(counter++);

  // Send message using ESP-Now protocol and print result

  if (!quickEspNow.send(DEST\_ADDR, (uint8\_t \*)message.c\_str(), message.length()))

  {

    Serial.println("Message sent");

  }

  else

  {

    Serial.println("Message not sent");

  }

  // Delay for SEND\_MSG\_MSEC before sending next message

  delay(SEND\_MSG\_MSEC);

}