Iot Smart Clock

Assembly and Programming Guide

Preface

Thank you for joining my IoT Smart Clock workshop. By the end of the workshop, you will have a working RGB pixel display that can display any sort of information as programmed, including time, weather, news, stock price and more.

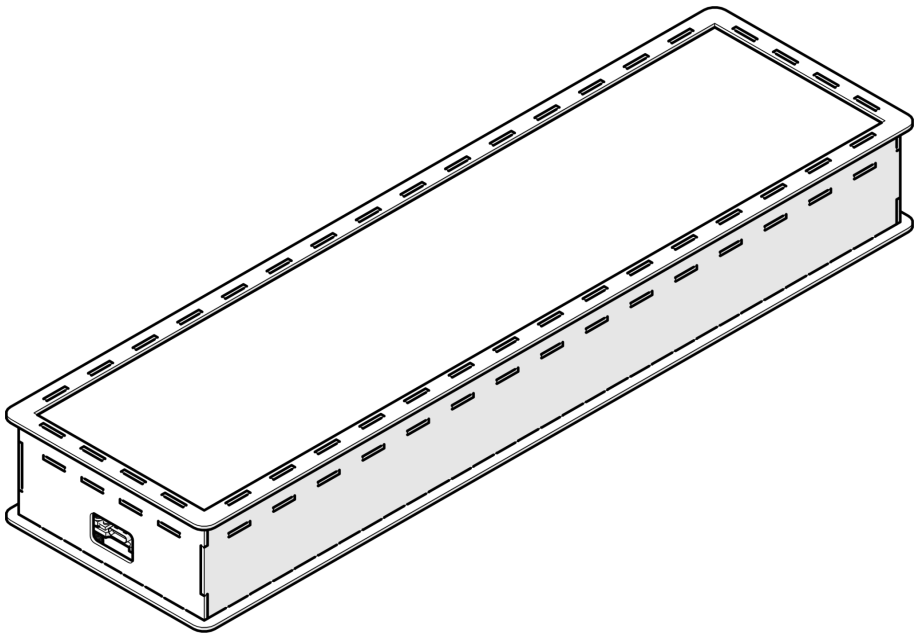
I am Justus Ip, the designer of the IoT Smart Clock. I am a year 4 Computer Science student, but also like to design IoT and hardware devices. For any inquiries, please don't hesitate to contact me at me@justusip.com. Feel free to also follow my Linkedin (https://www.linkedin.com/in/justusip/).

Happy building!

Section 1

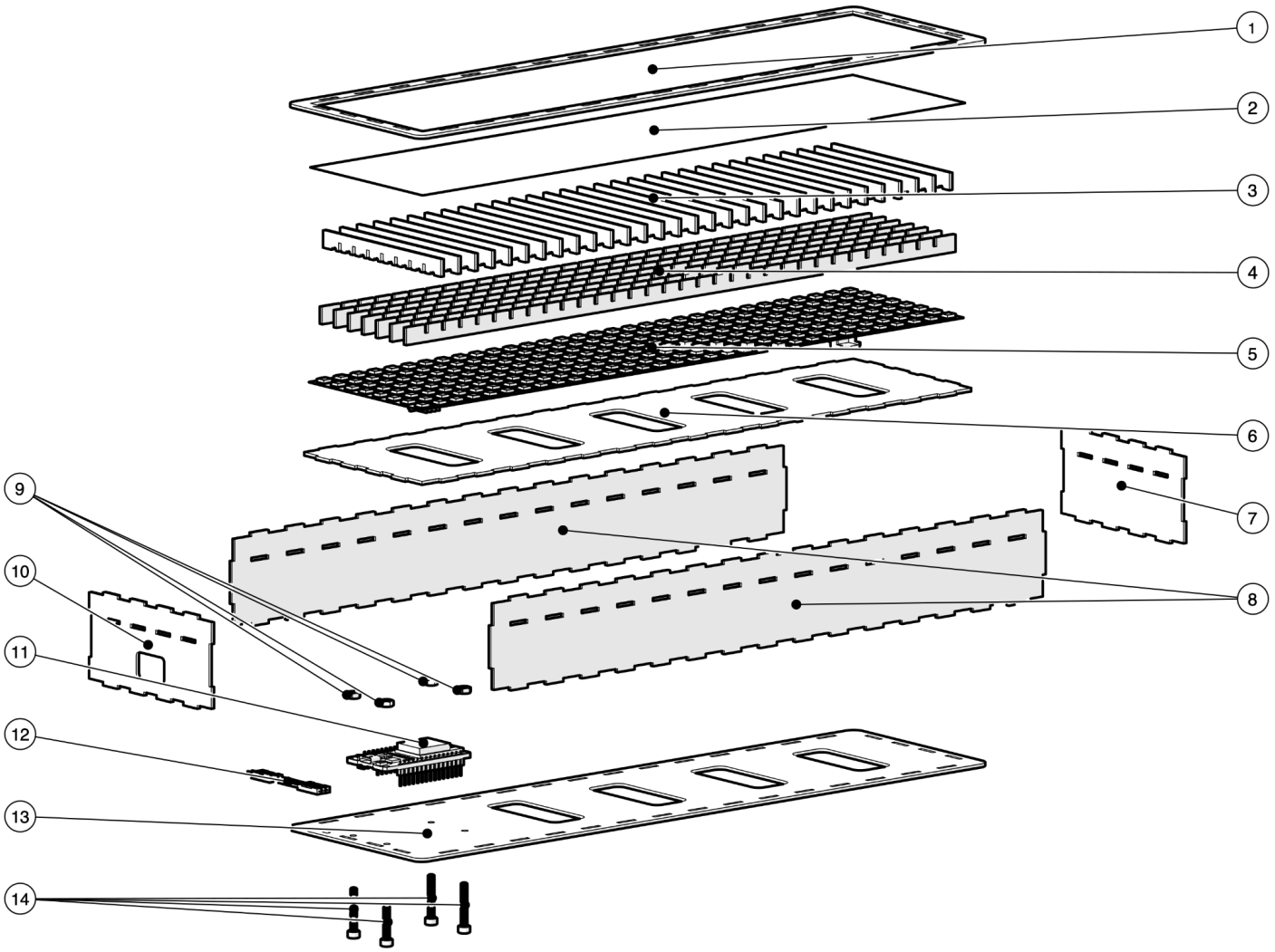
Enclosure Assembly

In this section, we will assemble the enclosure first before proceeding to the programming part. The finished product looks like the following.



1. Parts Checking

Please check if you have all the parts as listed below.



#	Part Name	Quantity
1	Front Frame	1
2	Semi-Transparent Film	1
3	Vertical Divider	31
4	Horizontal Divider	7
5	WS2812 LED Matrix Panel	1
6	Middle Pane	1
7	Right Pane	1

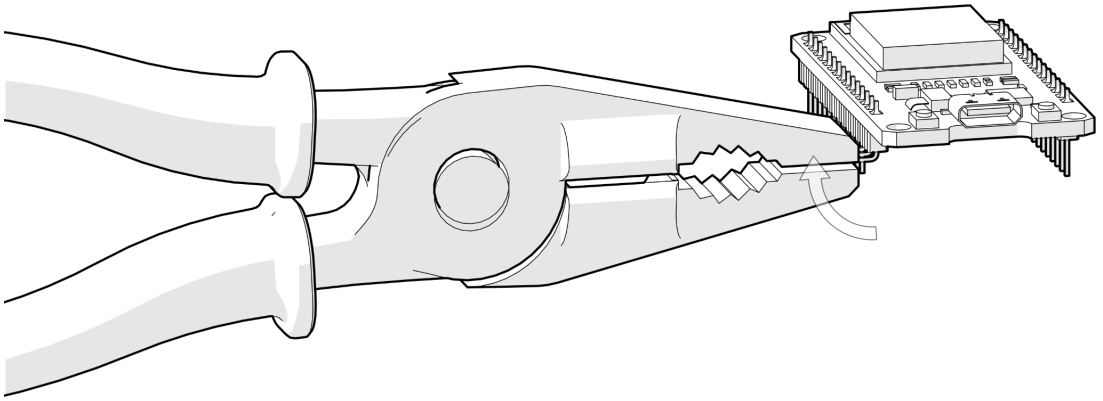
#	Part Name	Quantity
8	Top & Bottom Pane	1
9	M3 Nut	4
10	Left Pane	1
11	ESP32 Development Board	1
12	Female to Male Jumper Wire	3
13	Bottom Pane	1
14	M3 20mm Screw	4

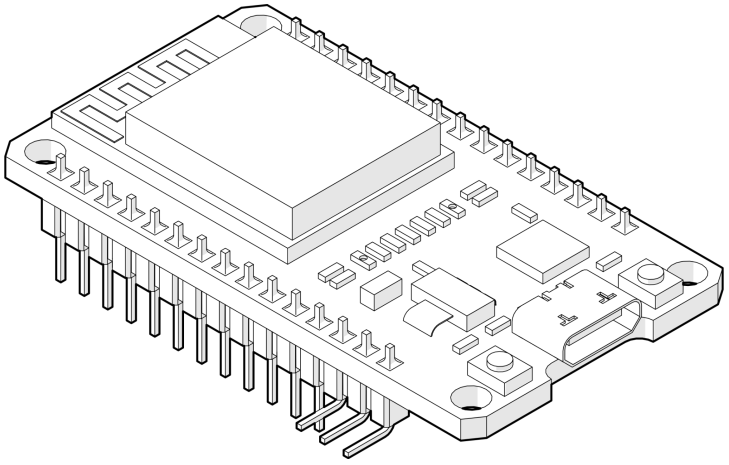
You also need these tools: pliers, screwdriver (⊕M3), hammer, hot-glue gun

2. Bend ESP32 Pins

Use a plier to bend the “VIN”, “GND” and “13” PIN of the ESP32 Development Board to a 90 degrees angle (the 3 pins most near to the EN pin).

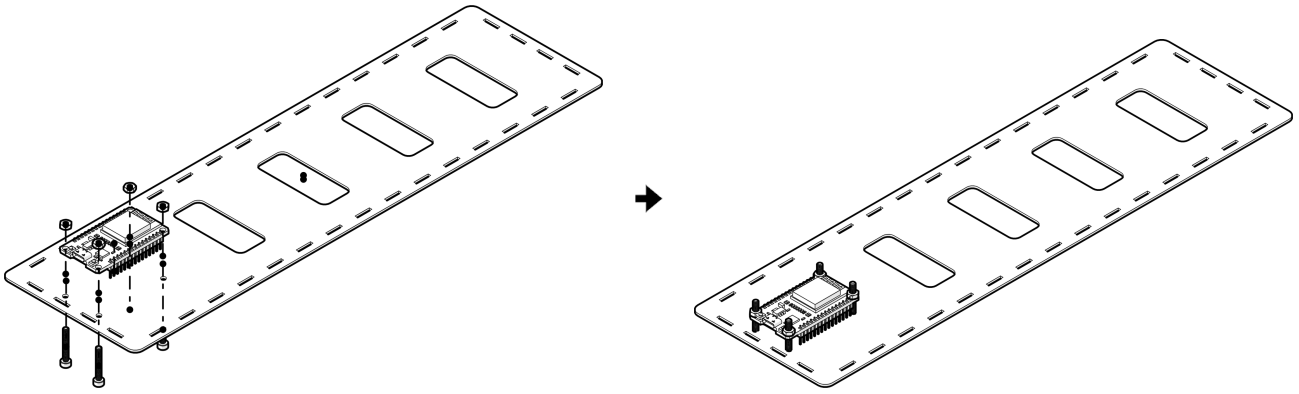
We will use only these 3 pins of ESP32 Development Board in this project, and bending these 3 pins to a 90 degrees angle will be useful later when we insert the jumper wires after the ESP32 Development Board is mounted onto the case.

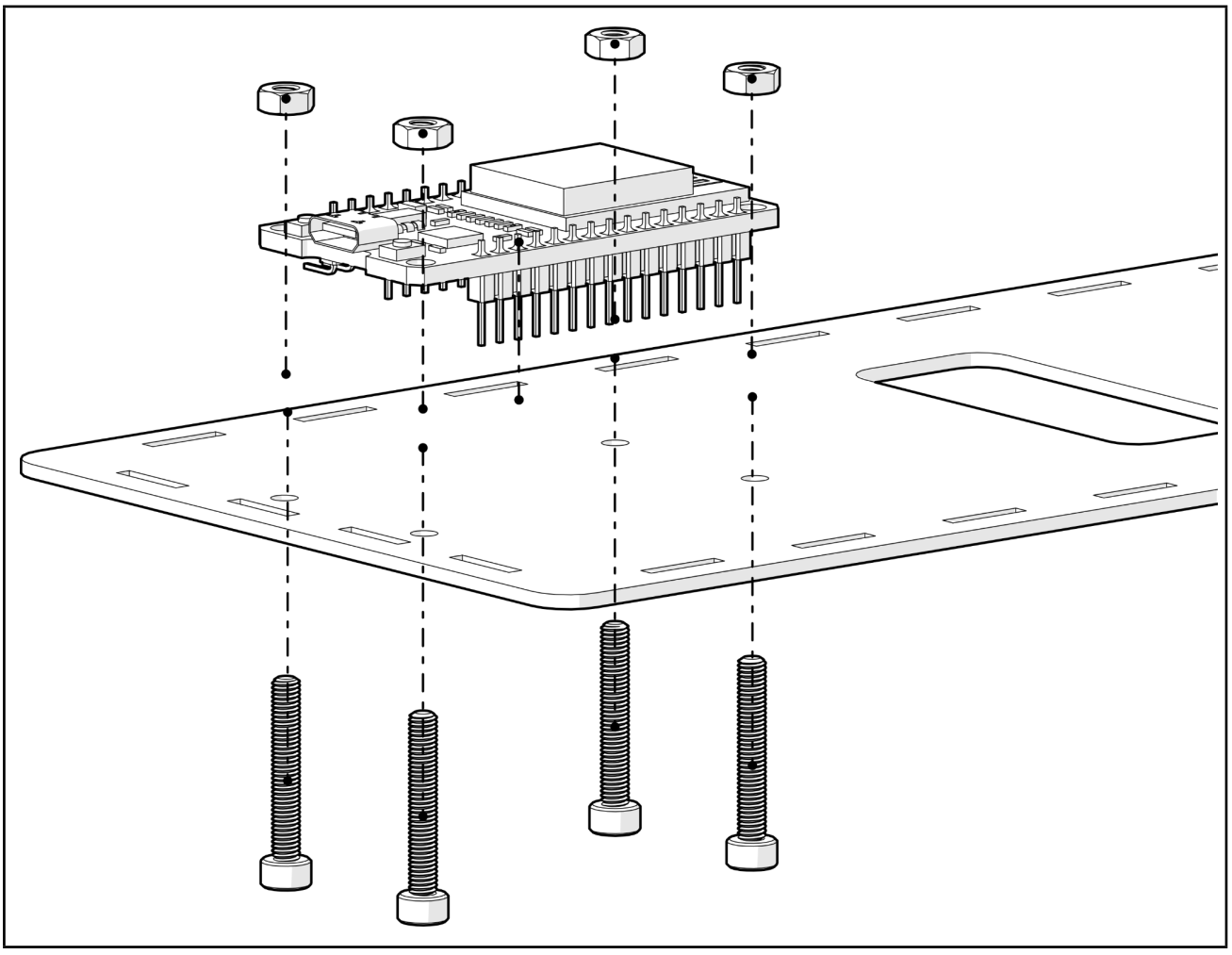




3. Bottom Part Assembly

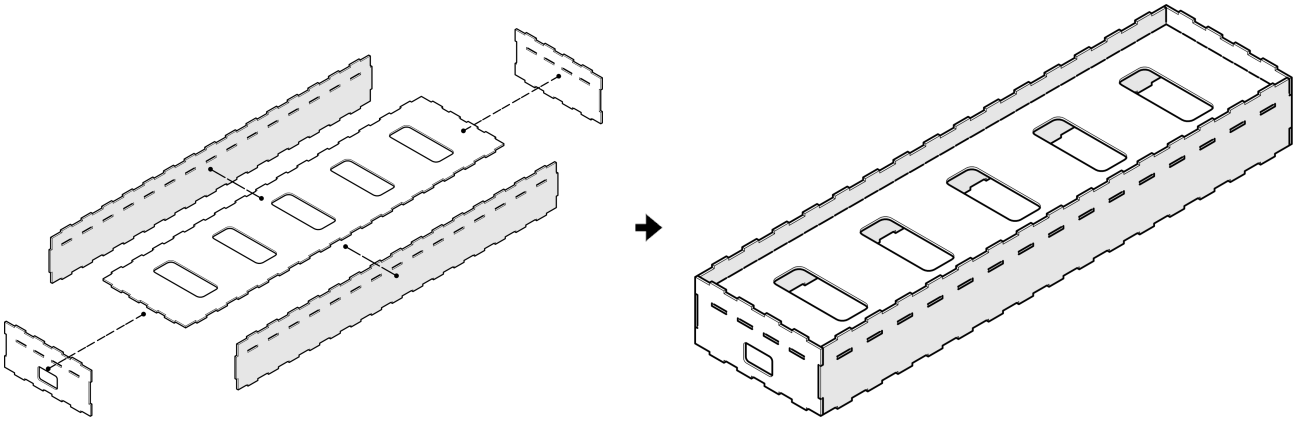
Mount the ESP32 Development Board by inserting 4 M3 20mm Screws from the bottom. Insert through the 4 holes of the bottom casing and the mounting holes of the ESP32 Development Board. Secure the screws with 4 M3 nuts from the top.





4. Middle Body Assembly

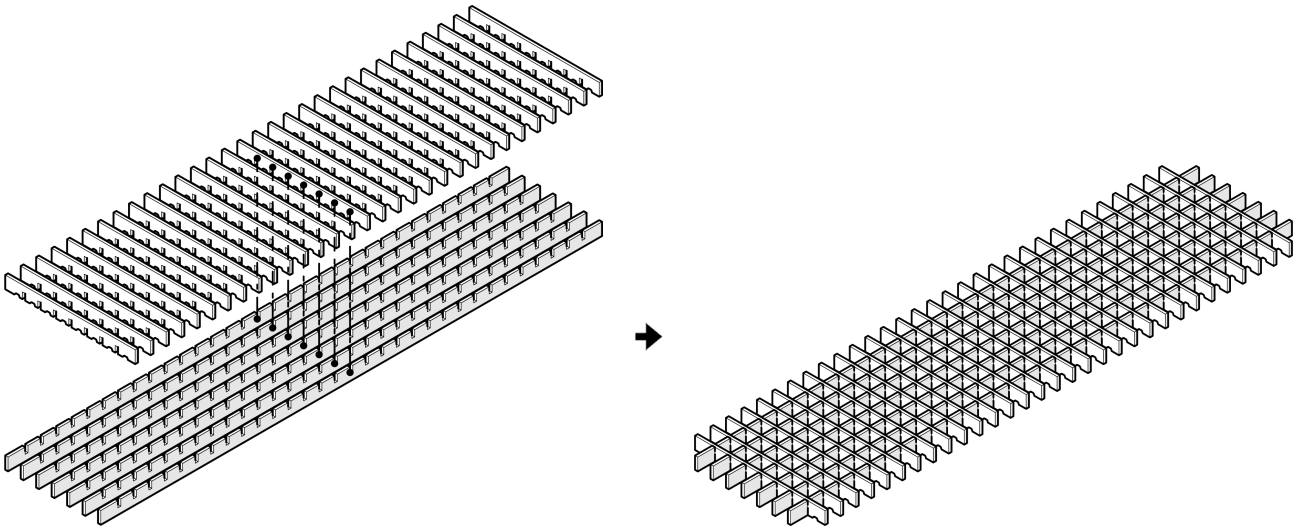
Combine the middle pane with the 4 side panes. Make sure the left pane is the one with the USB-C Port opening. Apply force on tab-and-slots of the boards to make sure they are interlocked securely.



5. Divider Assembly

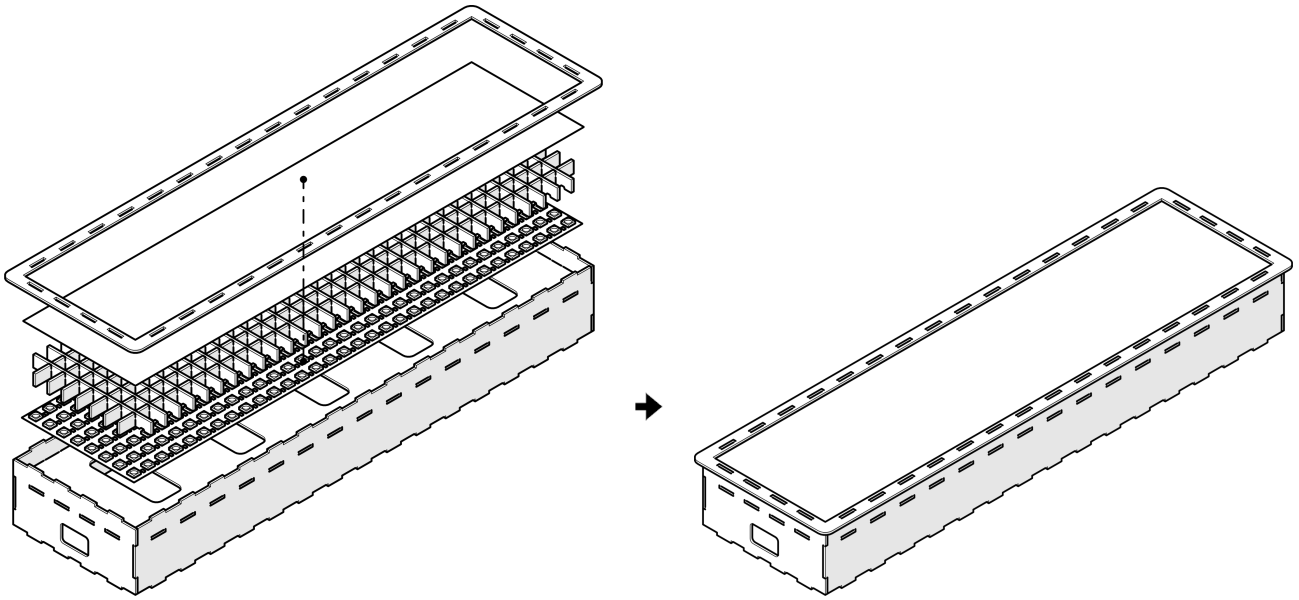
Divider is a grid-like structure to be placed on top of the LED Matrix Panel and the semi-transparent film. It is used to “divide” the lights emitted by the LED Matrix Panel so they look like they are pixelated with clear distinction from each other after passing through the semi-transparent light diffusion film.

It is composed of 31 vertical and 7 horizontal “slides”. Combine them together using the slots within them. Make sure all the dividers are strictly aligned with each other, and all vertical dividers are pushed all the way down before proceeding to the next step!

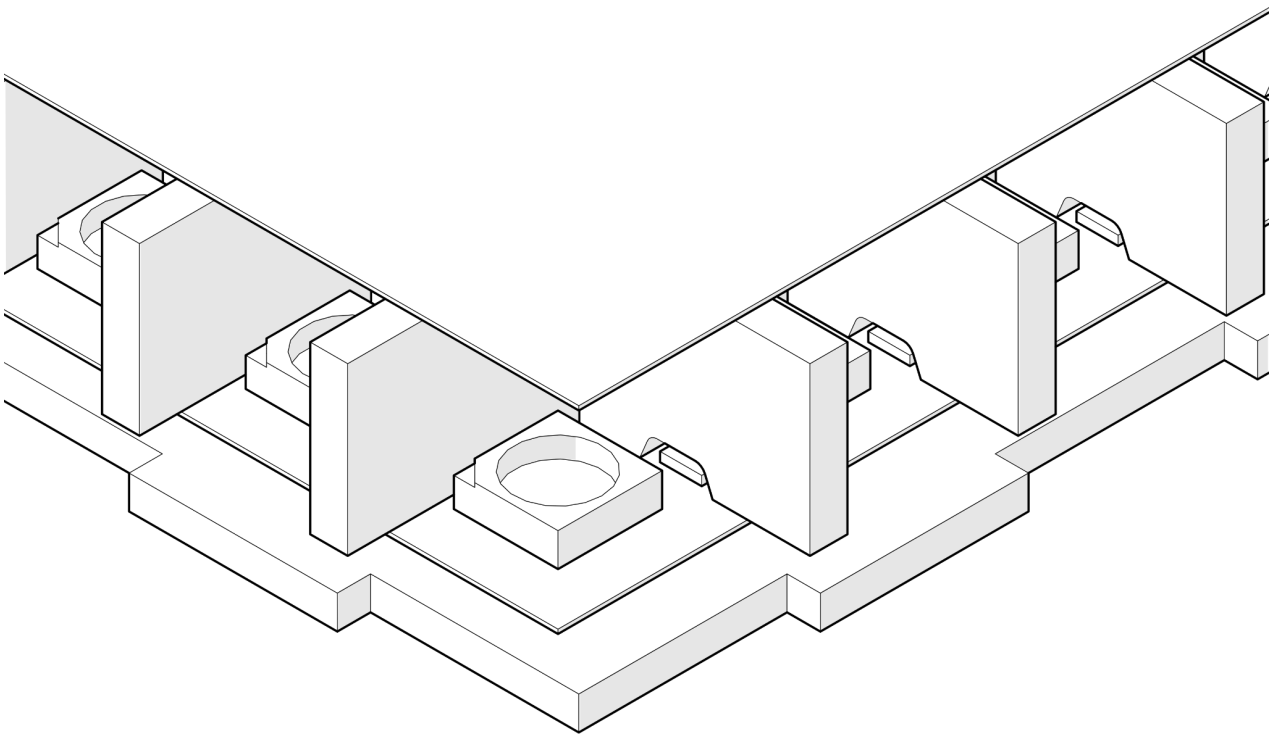


6. Top Part Assembly

The LED Matrix Panel, dividers, semi-transparent film and the front frame can now be pushed sequentially, one by one. Use a hammer to lightly push down tab-and-slots of the front frame to make sure they are interlocked securely.

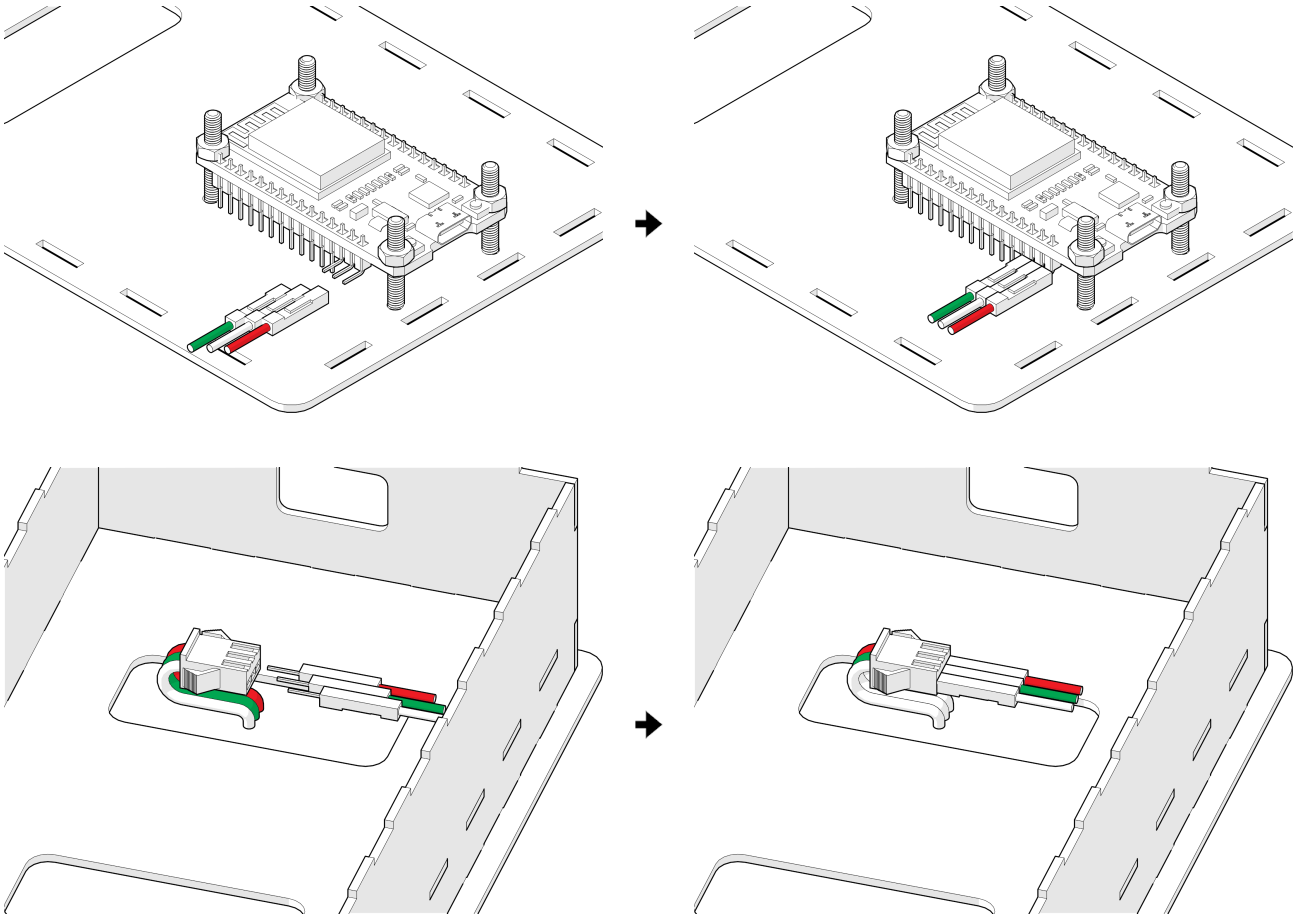


When pushing the dividers down the LED Matrix Panel, align the small openings of the vertical dividers with the resistors (the small component next to every LED light). Be careful not to break the LED Matrix Panel due to misalignment between the LED Matrix Panel and the divider when pushing other parts down from above.



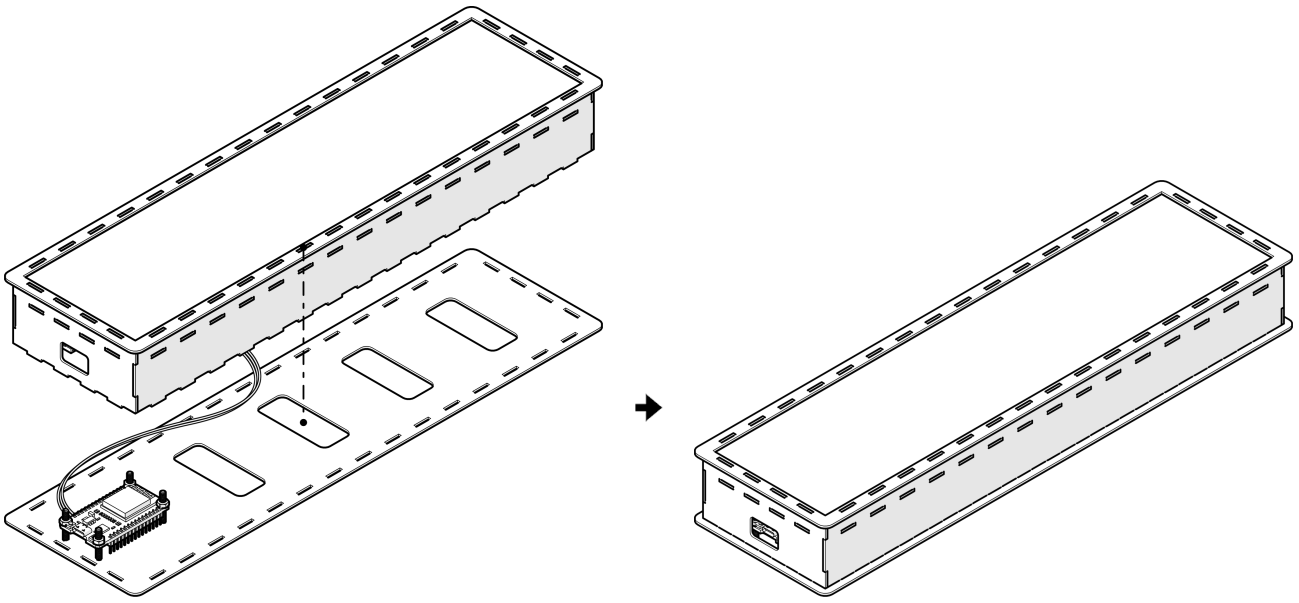
7. Connecting Wires

Using 3 female-to-male wires, connect the “VIN”pin of the ESP32 Development Board to the Red pin of the JST-SM male connector of the LED Matrix Panel, “GND” to Black, and “13” to Green. Feel free to use a hot-glue gun to fix the connections although usually it is strong enough to just leave it.



8. Combine Top & Bottom Part

After final checking all the connections, you can combine the top and the bottom parts to get the combined product.



Congrats! You may now move on to the programming part.

Programming

In this section, you can try to program the ESP32 to display some short messages, using the Arduino IDE (a code editor) and the C++ programming language. Later, in section C, you can upload the “standard firmware” that I have made which contains rich features, but for now let’s experience programming the ESP32 from scratch.

1. **Installation of the Arduino IDE**
    - Go to the Arduino website at https://www.arduino.cc/en/software
    - Download the Arduino IDE for your operating system (Windows, Mac, or Linux).
    - Install the Arduino IDE by following the on-screen instructions.
2. **Enabling ESP32 Support in Arduino IDE**
    - Open the Arduino IDE.
    - Go to File > Preferences.
    - In the "Additional Boards Manager URLs" field, enter the following URL: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
    - Click the "OK" button.
    - Go to Tools > Board > Boards Manager.
    - Search for "ESP32" and click the "Install" button for the "ESP32 by Espressif Systems" board
    - Wait for the installation to complete.
3. **Adding the Adafruit_NeoMatrix Library to Arduino IDE**
    - Go to Sketch > Include Library > Manage Libraries.
    - In the Library Manager, search for "Adafruit_NeoMatrix".
    - Click on the "Adafruit_NeoMatrix" library.
    - Select the latest version of the library (if multiple versions are available).
    - Click the "Install" button.
    - Wait for the installation to complete.
4. **Uploading your first sketch to ESP32**
    - Start a new sketch.
    - Enter the following code:

```
#include <Adafruit_GFX.h>
#include <Adafruit_NeoMatrix.h>
#include <Adafruit_NeoPixel.h>
#include <Fonts/TomThumb.h>

// Initialize an Adafruit NeoMatrix instance
Adafruit_NeoMatrix matrix = Adafruit_NeoMatrix(
   32, // Matrix width in pixels.
   8,  // Matrix height in pixels.
   13, // ESP32 pin number for NeoPixel data out.
   // Matrix layout settings - add together NEO_MATRIX_* values to declare
orientation, rotation, etc.
   NEO_MATRIX_TOP + NEO_MATRIX_LEFT + NEO_MATRIX_COLUMNS +
NEO_MATRIX_PROGRESSIVE + NEO_MATRIX_ZIGZAG,
   // NeoPixel LED type settings
   NEO_GRB + NEO_KHZ800
);

// The setup function is called once at startup
void setup() {
   // Setting preferences for the matrix instance
```

```
    matrix.begin();
    matrix.setTextWrap(false);
    matrix.setBrightness(30);
    matrix.setFont(&TomThumb);

    // Print out a simple "Hello!" message
    matrix.print("Hello!");
    matrix.show();
}

// The loop function is called repeatedly after the setup function is
finished.
void loop() {

}
```

- Connect the ESP32 to your computer via a USB-C cable.
- Click the Upload button on the top left of the Arduino IDE (A circle button with an arrow pointing to the right).
- Wait for the Arduino IDE to flash the program into ESP32.
- After the program is flashed, the ESP32 board will restart automatically and execute the program. You should see the text “Hello!” showing on the screen.

5. Mini-Challenges

Congratulations! You have finished displaying your first message in the LED Matrix. Try to understand the above code. Try to modify the above code to perform the following. Search Google for which Adafruit NeoMatrix library function to be called to perform the following.

- Change content of the message
- Change colour of the message
- Change colour of the background

6. Connecting to WiFi and Fetch Information online

We can connect ESP32 to the WiFi, fetch real-time information from the web and show them. Copy the following code and change “WiFi.begin("Your WiFi Name", "Your WiFi Password");” to your WiFi Name and WiFi Password of your mobile hotspot (or our provided WiFi hotspot). Upload the below code:

```
#include <Adafruit_GFX.h>
#include <Adafruit_NeoMatrix.h>
#include <Adafruit_NeoPixel.h>
#include <Fonts/TomThumb.h>

#include <WiFi.h>
#include <HTTPClient.h>

// Initialize an Adafruit NeoMatrix instance
Adafruit_NeoMatrix matrix = Adafruit_NeoMatrix(
   32, // Matrix width in pixels.
   8,  // Matrix height in pixels.
   13, // ESP32 pin number for NeoPixel data out.
   // Matrix layout settings - add together NEO_MATRIX_* values to declare
orientation, rotation, etc.
   NEO_MATRIX_TOP + NEO_MATRIX_LEFT + NEO_MATRIX_COLUMNS + NEO_MATRIX_PROGRESSIVE
+ NEO_MATRIX_ZIGZAG,
   // NeoPixel LED type settings
   NEO_GRB + NEO_KHZ800
);

// The setup function is called once at startup
void setup() {
   // Setting preferences for the matrix instance
   matrix.begin();
   matrix.setTextWrap(false);
   matrix.setBrightness(30);
   matrix.setFont(&TomThumb);

   // Set the username and password of the WiFi that the ESP32 is supposed to
connect to.
   // After this is set, ESP32 will connect to this WiFi and reconnect to it
automatically when disconnected later.
   WiFi.begin("Your WiFi Name", "Your WiFi Password");

   // Print out a simple "Hello!" message
   // matrix.print("Hello!");
   // matrix.show();
}

// The loop function is called repeatedly after the setup function is finished.
void loop() {
   if (WiFi.status() != WL_CONNECTED) {
       // While ESP32 is automatically trying to connect to the WiFi in the
background,
       // Show a "No WiFi!" message on the screen. Wait for 500ms before
returning to the beginning
       // of the function and trying again.
       matrix.fillScreen(matrix.Color(0, 0, 0)); // Clear the screen
       matrix.setCursor(1, 7); // The cursor has to be reset everytime before
printing something on the screen
       matrix.print("No WiFi!"); // Writes the text to be shown to the internal
buffer
       matrix.show(); // Draw text in the internal buffer on the screen
       delay(500);
       return;
   }
```

```
    // Create an instance of the HTTP client, and perform a HTTP GET request to
the below URL.
    HTTPClient http;
    http.begin("http://iot-smart-clock.justusip.com/raw/time");

    // Check the HTTP response code. If successful, the response code should be
200.
    // If it is successful, show an error message on the screen and wait for 1
second before
    // returning to the beginning of the function and trying again.
    int httpCode = http.GET();
    if (httpCode != HTTP_CODE_OK) {
        matrix.fillScreen(matrix.Color(0, 0, 0));
        matrix.setCursor(1, 7);
        matrix.print("Error!");
        matrix.show();
        delay(1000);
        return;
    }

    // If the URL content is correctly accessed, show the content on the screen.
    // For the above URL, it will return the current time in the format of "HH:MM
AM/PM" in raw text (not HTML!)
    // We can show it directly on the screen.
    String payload = http.getString();
    matrix.fillScreen(matrix.Color(0, 0, 0));
    matrix.setCursor(1, 7);
    matrix.print(payload);
    matrix.show();

    delay(5000);
}
```

Turn on mobile hotspot on your phone (or use our provided WiFi hotspot). Then, you should be able to see the current time on the screen.

7. Mini Challenges

- Change the URL to the following to display the time

http://iot-smart-clock.justusip.com/raw/time

- Change the URL to the following to display the temperature

http://iot-smart-clock.justusip.com/raw/temperature

8. Uploading the “NeoClock” Firmware

Due to time restrictions, I have completed a much more enhanced program for the IoT Smart Clock which supports displaying different types of pages including pages that display the current time, date, weather, stock prices and static message. This firmware connects to WiFi and fetches the latest time & date via NTP protocol, and weather & stock prices via my self-hosted API. You may also add your own page displaying your favourite kinds of content dynamically.

Follow the instructor’s instruction for uploading the firmware.