



Facial Detection and Recognition with MEMENTO

Created by Brent Rubell



<https://learn.adafruit.com/facial-detection-and-recognition-with-memento>

Last updated on 2024-10-16 11:29:35 AM EDT

Table of Contents

Overview	3
• Parts	
Wiring	5
Upload Code	6
• Upload Code using Web Serial ESPTool	
• Download the application for your board	
• Connect and Upload	
Usage	10
• Face Detection	
• Face Detection Explanation	
• Face Recognition	
• Going Further - "Tricking" the Facial Recognition	
Code Walkthrough	19
• Example/Demo Code History and Explanation	
• Capturing a photo	
• Face Detection	
• Face Recognition	
Restoring MEMENTO Demo and UF2 Bootloader	25
Modify Code using PlatformIO	25
• Install PlatformIO	
• Configure Your Workspace	
• Build and Upload with PlatformIO	

Overview



Want to play around with computer vision "at the edge" without the overhead and complexity of compiling a dataset and training a model?

Upload the code in this guide to your Adafruit MEMENTO Camera Board and **turn the MEMENTO into a camera that can both detect and recognize faces!**

Going further - the example code in this guide may be used and modified to build your next facial detection or recognition electronics project!

About the code in this guide

This project uses an example written by Me-No-Dev for Espressif Systems, [CameraWebServer.ino](https://adafru.it/19fj) (<https://adafru.it/19fj>). This example was modified by the author of this guide to reduce the flash size overhead (we removed the web functionality, isolated the face detection/recognition calls, brought this overhead into **main.cpp** and **ra_filter.h**), add compatibility for the Adafruit MEMENTO development board (added camera compatibility and added "blitting" the camera's raw image to the MEMENTO's TFT instead of to a webpage), set up a PlatformIO build environment to decrease compile time, and created an interactive demo around the code.

Parts



[MEMENTO - Python Programmable DIY Camera - Bare Board](https://www.adafruit.com/product/5420)

Make memories, or just a cool camera-based project, with Adafruit's MEMENTO Camera Board. It's a development board with everything you need to create...

<https://www.adafruit.com/product/5420>

1 x [Woven USB A to USB C Cable](https://www.adafruit.com/product/5153)

<https://www.adafruit.com/product/5153>

Pink and Purple Woven USB A to USB C Cable - 1 meter long

1 x [256MB Micro SD Card](https://www.adafruit.com/product/5251)

<https://www.adafruit.com/product/5251>

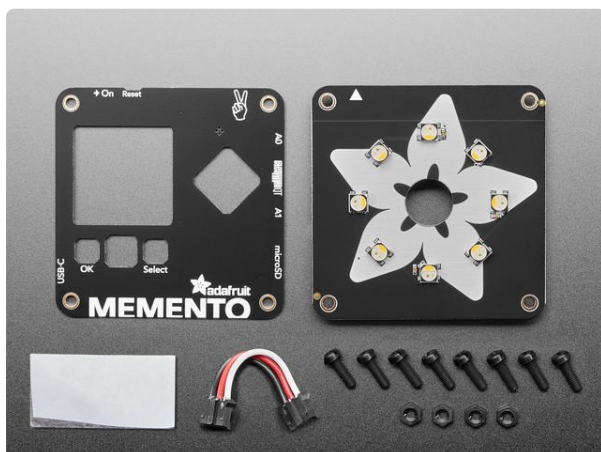
256MB Micro SD Memory Card

1 x [420mAh LIPO Battery](https://www.adafruit.com/product/4236)

<https://www.adafruit.com/product/4236>

Lithium Ion Polymer Battery with Short Cable - 3.7V
420mAh

You also may want an RGBW LED Ring with 8 bright NeoPixels. We include this with the MEMENTO Camera Enclosure Kit:

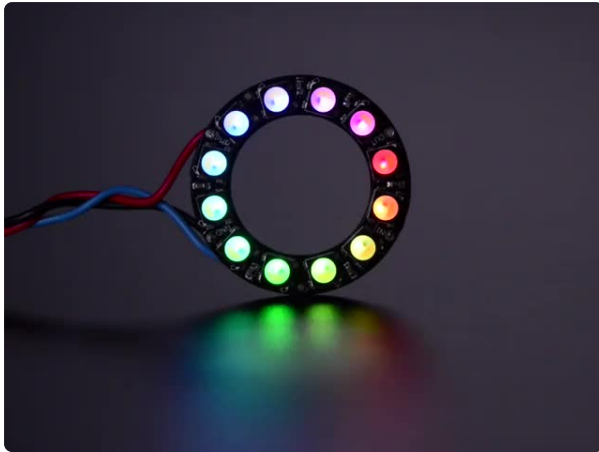


[Adafruit MEMENTO Camera Enclosure & Hardware Kit](https://www.adafruit.com/product/5843)

Once you've picked up your MEMENTO Camera and you're ready to take it out into the world, here is a chic and minimalist enclosure that will look great on the...

<https://www.adafruit.com/product/5843>

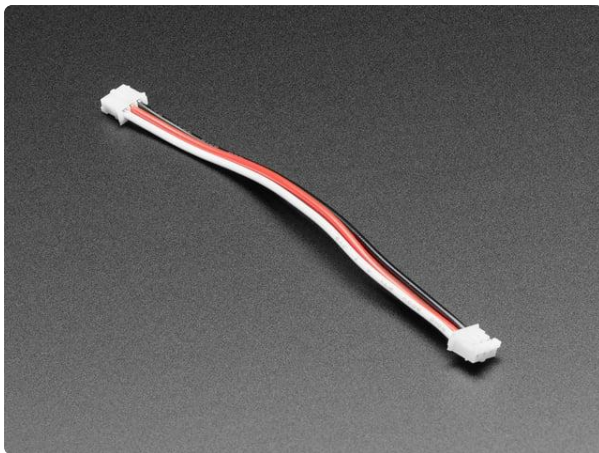
If you do not have the MEMENTO Camera Enclosure Kit (or if it is out of stock), you can build your own ring light for the MEMENTO using the following parts:



NeoPixel Ring - 12 x 5050 RGBW LEDs w/ Integrated Drivers

What is better than smart RGB LEDs? Smart RGB+White LEDs! These NeoPixel rings now have 4 LEDs in them (red, green, blue and white) for excellent lighting effects. Round and...

<https://www.adafruit.com/product/2852>



JST PH 2mm 3-pin Plug-Plug Cable - 100mm long

This cable is a little over 100mm / 4" long and fitted with JST-PH 3-pin connectors on either end. We dig the solid and compact nature of these connectors and the...

<https://www.adafruit.com/product/4336>

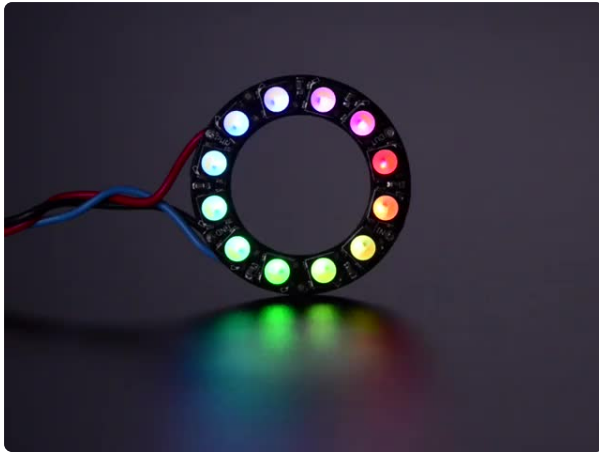
Wiring

Wiring using the MEMENTO Camera Enclosure Kit

If you are using the [Adafruit MEMENTO Camera Enclosure Kit \(http://adafru.it/5843\)](http://adafru.it/5843) with this guide, use the included JST cable to connect the Adafruit MEMENTO Camera LED Ring Front Plate 3-pin JST Connector and the 3-pin JST connector labeled A1 on the MEMENTO.

Wiring without the MEMENTO Camera Enclosure Kit

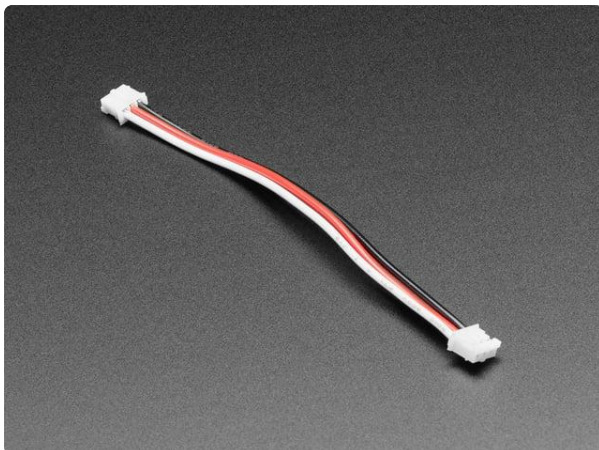
If you do not have the MEMENTO Camera Enclosure Kit (or if it is out of stock), you can build your own ring light for the MEMENTO using the following parts:



NeoPixel Ring - 12 x 5050 RGBW LEDs w/ Integrated Drivers

What is better than smart RGB LEDs? Smart RGB+White LEDs! These NeoPixel rings now have 4 LEDs in them (red, green, blue and white) for excellent lighting effects. Round and...

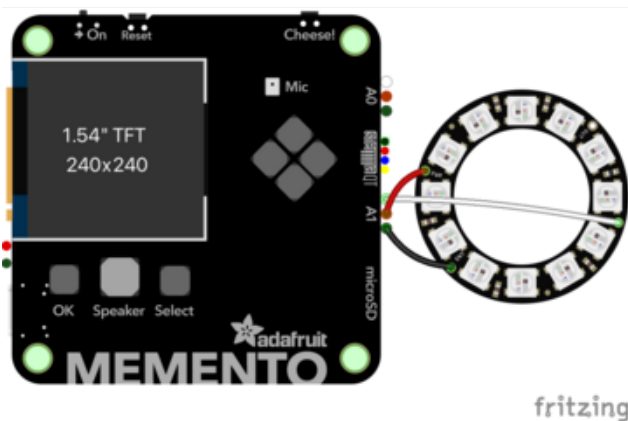
<https://www.adafruit.com/product/2852>



JST PH 2mm 3-pin Plug-Plug Cable - 100mm long

This cable is a little over 100mm / 4" long and fitted with JST-PH 3-pin connectors on either end. We dig the solid and compact nature of these connectors and the...

<https://www.adafruit.com/product/4336>



You must cut the JST PH 3-pin cable in half using a wire cutter. Using a soldering iron, make the following connections between the cable and the NeoPixel ring.

JST VCC (red wire) to NeoPixel PWR

JST GND (black wire) to NeoPixel GND

JST Data (white wire) to NeoPixel Data In

Then, plug the cable into the MEMENTO's A1 JST port.

Upload Code

Upload Code using Web Serial ESPTool

The WebSerial ESPTool was designed to be a web-capable option for programming Espressif ESP family microcontroller boards that have a serial-based ROM bootloader. It allows you to erase the contents of the microcontroller and program up to 4 files at different offsets.

For boards that lack native USB, like the ESP32 or ESP32-C3 microcontroller, this is how any firmware like CircuitPython .bin files can be loaded. **There is no drag-and-drop to a folder option for these boards.**

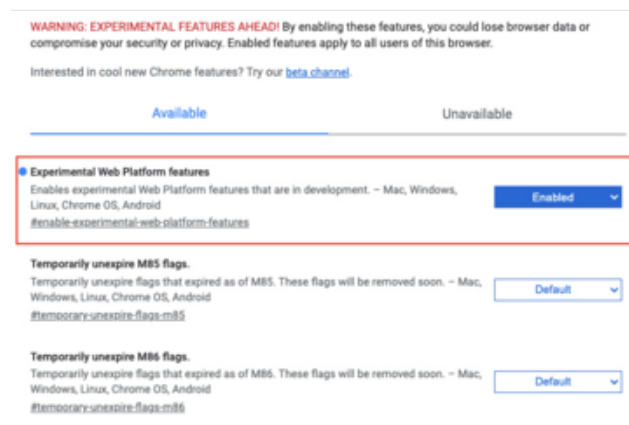
For boards with native USB, like ESP32-S2, -S3, etc. this is how the UF2 bootloader .bin file can be loaded. Once the UF2 bootloader is on, firmware like CircuitPython .uf2 files can be drag-and-dropped to a **BOOT** folder.

This tool is a good alternative for folks who cannot run Python **esptool.py** on their computer or are having difficulty installing or using **esptool.py**.

Enable Web Serial



You will have to use the Chrome or Chromium-based browser for this to work. [For example, Edge and Opera are Chromium \(https://adafru.it/10BL\)](https://adafru.it/10BL). Safari and Firefox, etc are not supported - [they have not implemented Web Serial \(https://adafru.it/10BM\)](https://adafru.it/10BM)!



If you have an ancient version of Chrome, you'll need to enable the Serial API, which is easy.

Visit <chrome://flags> from within Chrome. Find and enable the **Experimental Web Platform features**

Restart Chrome

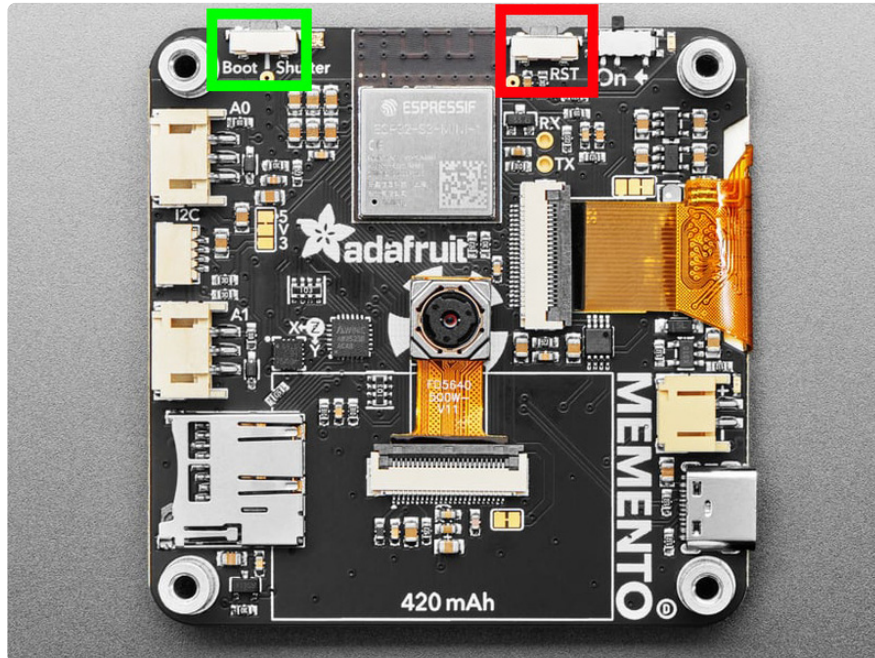
Enter Bootloader Mode

Before you can use the tool, you will need to put your board in bootloader mode. **Before you start, make sure your ESP32-S2/S3 is plugged into a USB port to your computer using a data/sync cable.** Charge-only cables will not work!

Turn on the On/Off switch - check that you see the green power light on so you know the board is powered, a prerequisite!

To enter the bootloader:

1. Press and hold the **BOOT/DFU** button down (green box). Don't let go of it yet!
2. Press and release the **Reset** button (red box). You should still have the **BOOT/DFU** button pressed while you do this.
3. Now you can release the **BOOT/DFU** button.



No USB drive will appear when you've entered the ROM bootloader. This is normal!

Download the application for your board

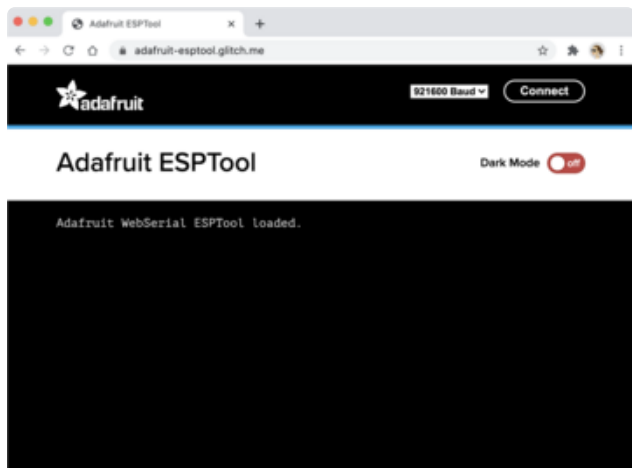
Click the button below to download the application binary file for your board.

**Download Firmware for MEMENTO
Face Detection and Recognition**

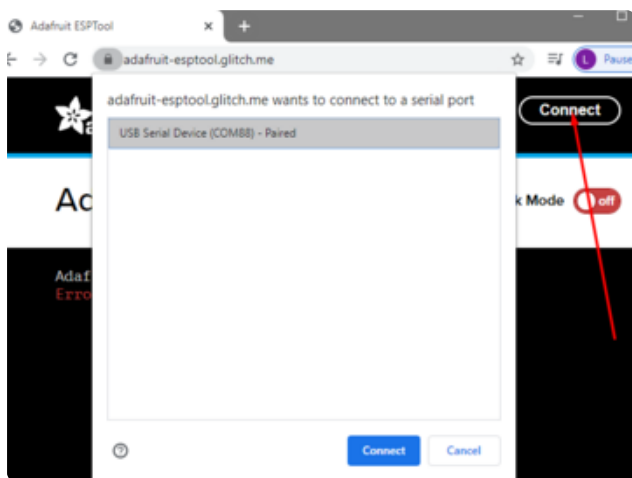
<https://adafru.it/1a8Y>

Connect and Upload

You should have plugged in **only** the **MEMENTO** board that you intend to flash. That way there's no confusion in picking the proper port when it's time!



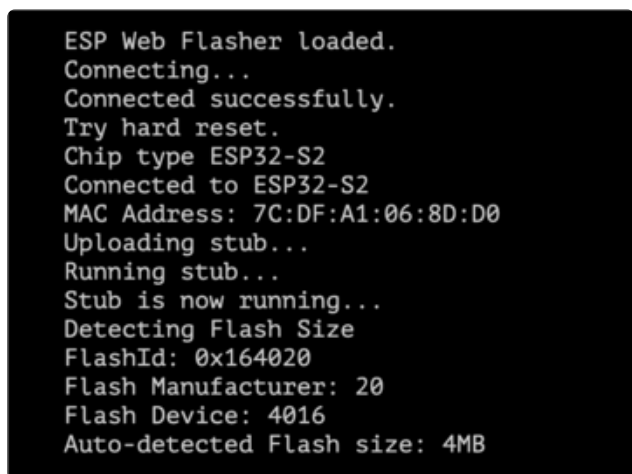
In the Chrome browser visit https://adafruit.github.io/Adafruit_WebSerial_ESPTool/ (<https://adafru.it/PMB>). You should see something like the image shown.



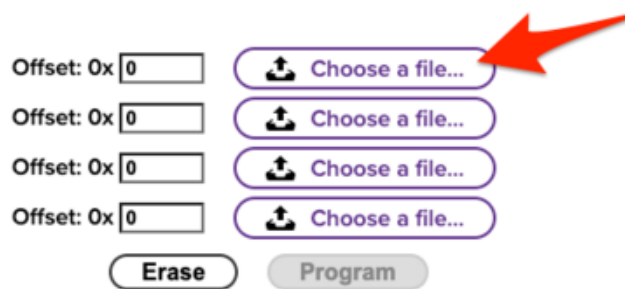
Press the **Connect** button in the top right of the web browser. You will get a pop-up asking you to select the COM or Serial port.

Remember, you should remove all other USB devices so only the ESP32-S2/S3 board is attached, that way there's no confusion over multiple ports!

On some systems, such as MacOS, there may be additional system ports that appear in the list.

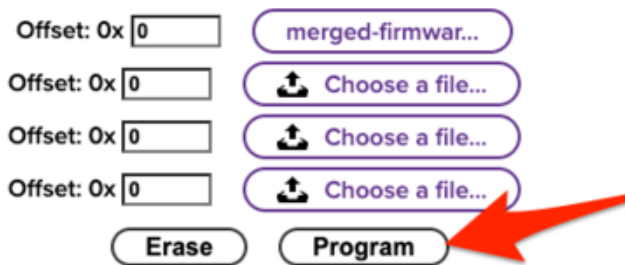


The JavaScript code will now try to connect to the ROM bootloader. It may time out for a bit until it succeeds. On success, you will see that it is **Connected** and will print out a unique **MAC address** identifying the board along with other information that was detected.



On the top of the tool, ensure the offset is set to `0x0` and click "Choose a File..."

From the file browser, select the .bin file you downloaded earlier.



Click Program and the binary will be uploaded to your MEMENTO board.



After the binary is uploaded to the MEMENTO, the console will instruct you to reset your device.

Press the **RESET** button on the **MEMENTO**. You should see the MEMENTO's screen briefly glow green, then show a preview of what the camera is seeing.

Usage

This application switches between two modes, face detection and face recognition.

What's the difference?

Face detection identifies when any face is present whereas face recognition can identify a specific face. Performing face detection is faster than performing face

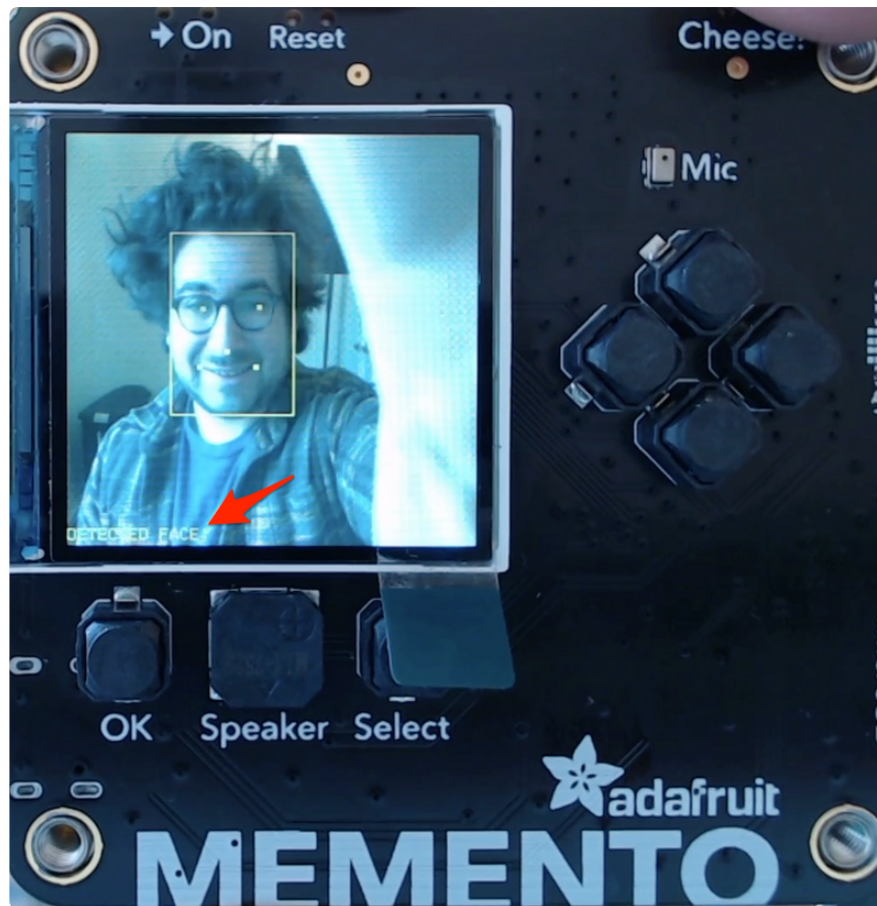
recognition. In face recognition, algorithms not only detect the face but also attempt to identify similarities between faces.

Face Detection

After uploading the code and pressing RESET, the MEMENTO's screen displays what the camera module sees.



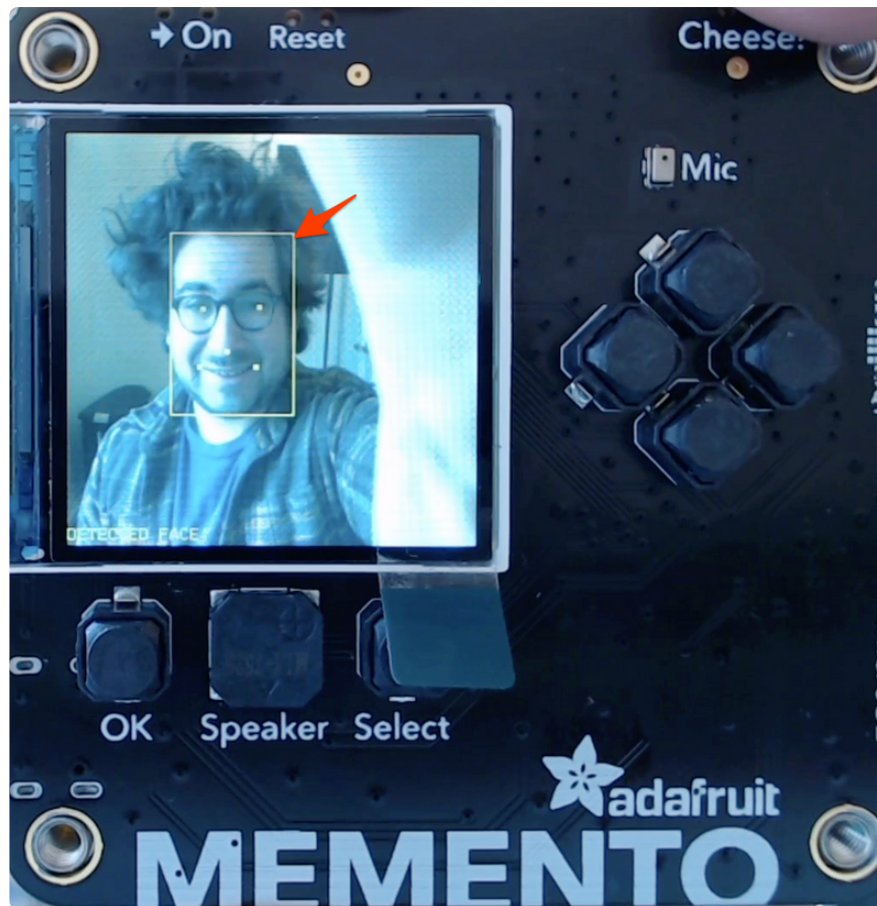
Move your face in front of the camera. If a face is detected by the MEMENTO, the screen displays the text, "DETECTED FACE!". A bounding box is drawn around the face and contains facial landmarks within it.



Face Detection Explanation

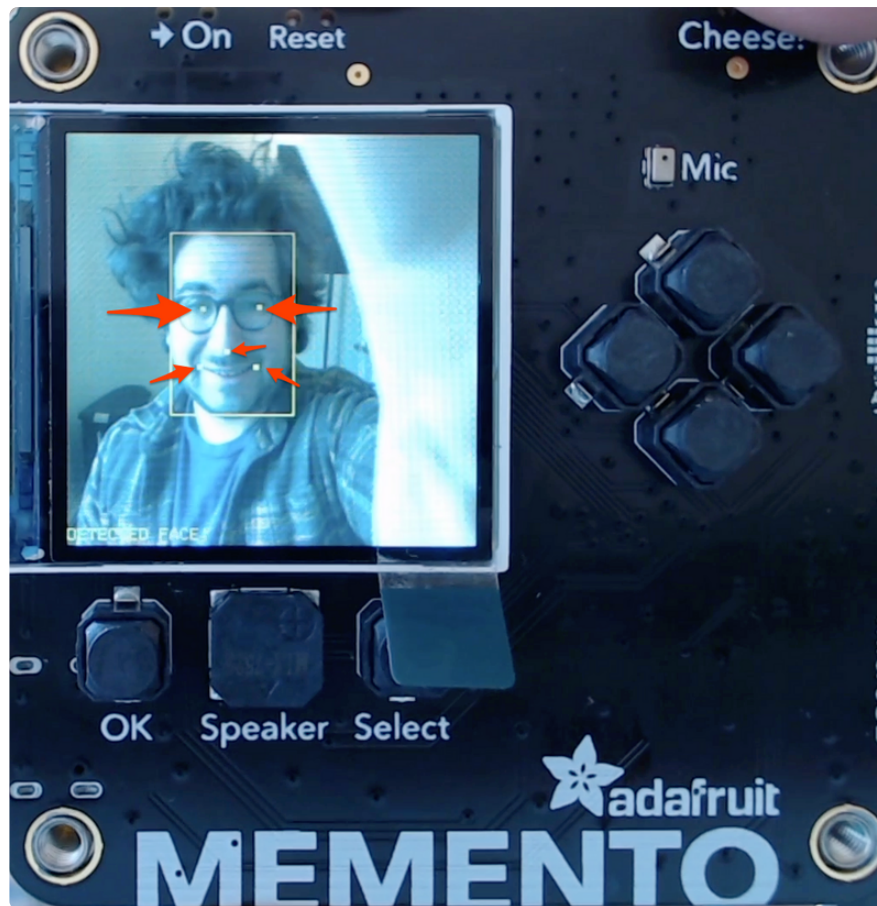
To understand what's happening in this demo, let's define a few terms first:

What is the green box around my face?



In computer vision, the green box around an object is called a [bounding box](https://adafru.it/19fp) (<https://adafru.it/19fp>). This box contains the x and y coordinates (the boundaries) of the detected object.

What are the points placed over my face?

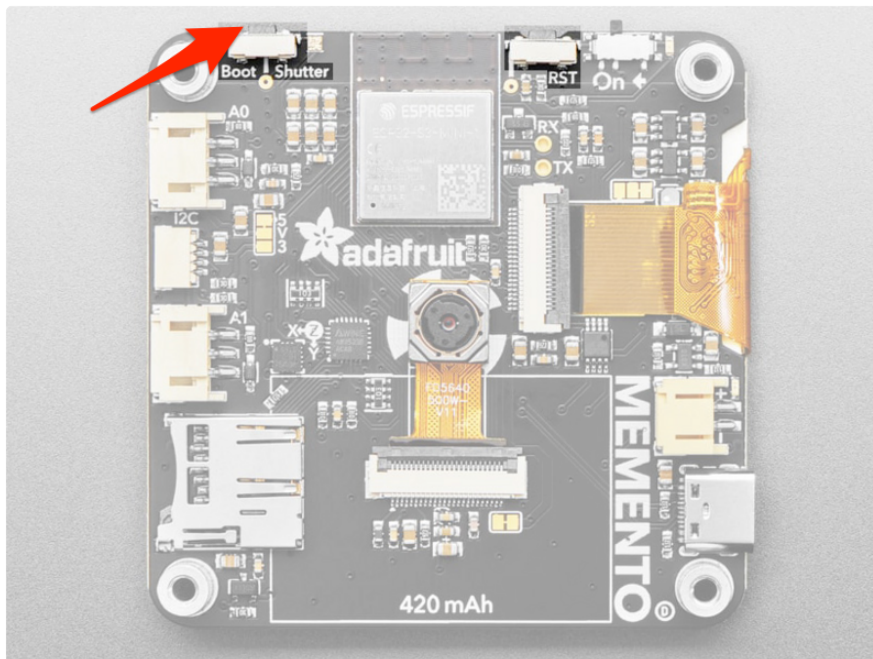


These points are called [landmarks](https://adafru.it/19fq) (<https://adafru.it/19fq>). The landmarks identified by this demo are facial landmarks, such as your left eye, mouth (left corner), nose, right eye, and mouth (right corner). Landmarks are primarily used for facial recognition but also can be used for emotional/mood analysis or pose detection.

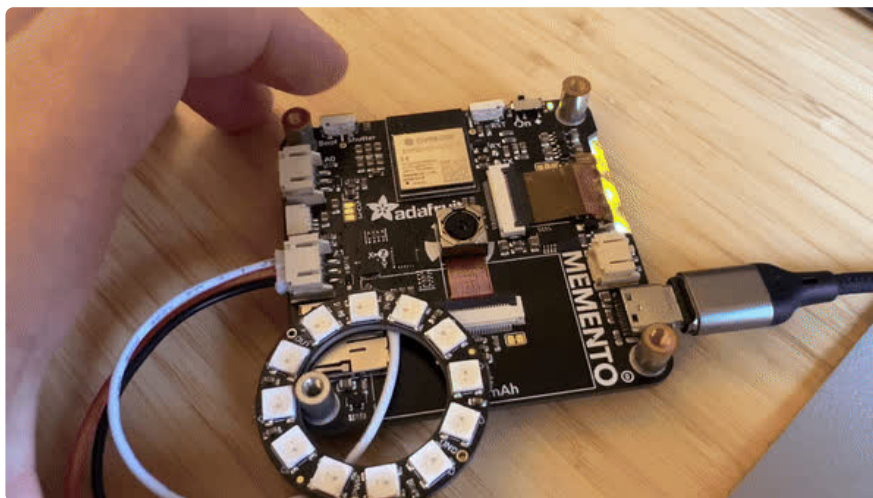
Face Recognition

Unlike the face detection demo which runs automatically, face recognition has a few steps. Before we can detect a face, the face needs to be enrolled first.

To enroll a new face, press the shutter button on the MEMENTO.



After pressing the shutter, the NeoPixel ring emits a white light to illuminate a face. The light indicates the MEMENTO is in "face enroll mode" and is attempting to detect a face.



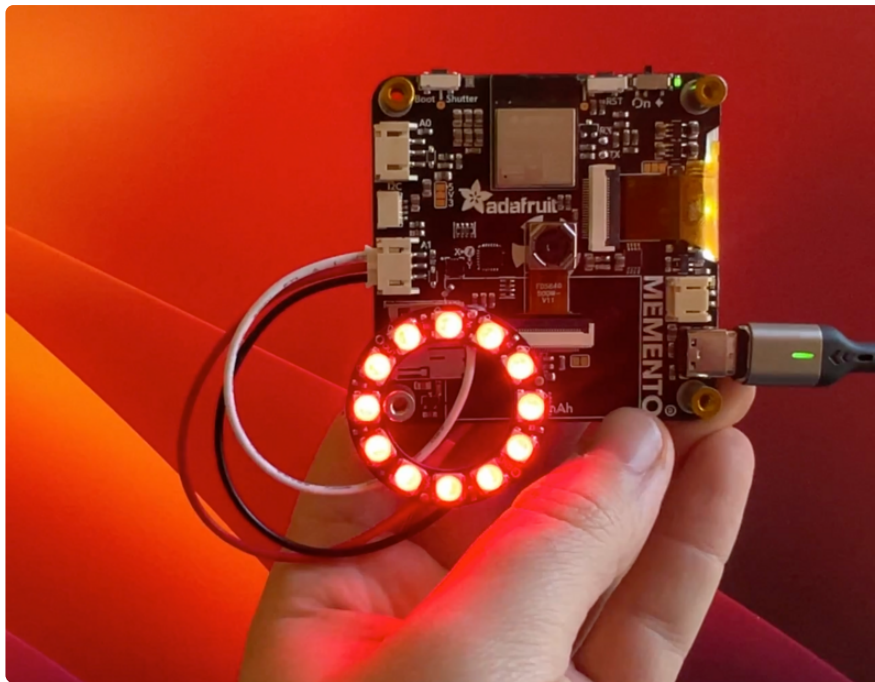
Move your face in front of the camera. If a face is detected, the screen will briefly display Enrolled a new face with ID #0 and the NeoPixel ring will turn off.



The next time the same face is moved in front of the camera, it will recognize it and the NeoPixel ring will turn green.

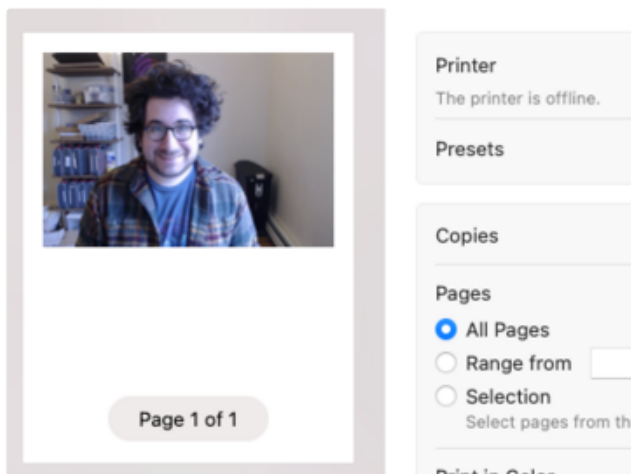


When you point the camera at a different, unrelated, face, the NeoPixel ring will glow red to indicate that the camera has detected a face but it is not one of the enrolled faces.

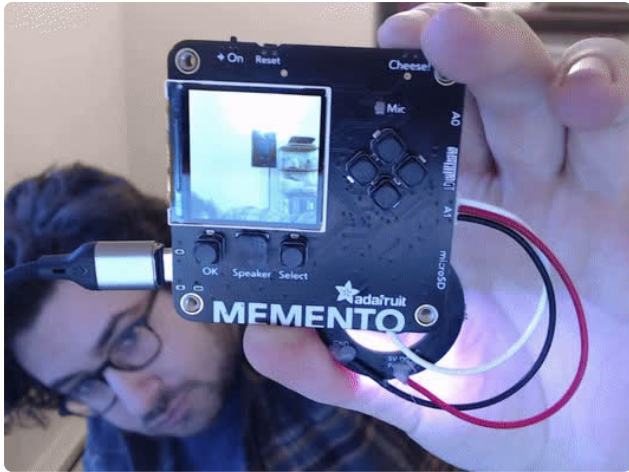


Going Further - "Tricking" the Facial Recognition

Since the facial recognition demo runs on an embedded system, Ladyada asked me to try out both the recognition performed by the MEMENTO's ESP32x and my iPhone's FaceID.



To attempt to test the face recognition code further, I went with something basic. I took a photo of my face and printed it out on printer paper.



Then, I placed the MEMENTO into the enrollment mode and enrolled my face.



After enrolling my face, I placed the photo of my face in front of the camera. It was recognized by the MEMENTO and lit up the NeoPixels with a green light.

However, my iPhone did not recognize the photo of my face. Even holding the photo in different lighting did not help.

Why?

The iPhone uses a more advanced version of facial recognition called [Face ID \(https://adafru.it/19fr\)](https://adafru.it/19fr). This system doesn't identify a face using a camera module like the MEMENTO code does, it performs recognition on the depth of the image.

Face ID begins by projecting over 30,000 infrared (invisible) "dots" onto your face. A module then projects infrared light on your face and an infrared camera takes a picture of the dots.



The pattern is then securely stored as a "map" (shown by the image above) in the iPhone's storage. The next time you put your iPhone up to your face, the camera will generate a new map and try to match it to the saved map.

Earlier on this page, we discussed landmarks as the method of identifying features for facial recognition. This is similar, except Face ID identifies over 30,000 landmark points compared to the MEMENTO which identifies 6 landmarks.

Code Walkthrough

Example/Demo Code History and Explanation

This project uses an example written by Me-No-Dev for Espressif Systems named [CameraWebServer.ino](https://adafru.it/19fj) (<https://adafru.it/19fj>). The example was modified by the author of this guide for Adafruit Industries to reduce the flash size overhead (we removed the web server functionality, isolated the face detection/recognition calls, brought this overhead into **main.cpp** and **ra_filter.h**), add compatibility for the Adafruit MEMENTO development board (added camera compatibility and added "blitting" the camera's raw image to the MEMENTO's TFT instead of to a webpage), and build an interactive demo around it.

So, since this is a larger codebase than a typical learn project and we only modified the code, this page won't explain everything that CameraWebServer does. It will explain the important and modifiable code segments within **main.cpp**.

Capturing a photo

Most applications for a digital camera like the MEMENTO require the camera to save photos in a compressed file format (like JPEG) to save space on the SD card and a

large resolution, we found the facial detection code runs fastest with a smaller frame size (240x240px) and the RGB565 raw bitmap.

Within `main.cpp`, the `initCamera()` function handles initializing the MEMENTO's camera. This code segment configures the camera's frame size to 240x240px and its pixel format to RGB565.

```
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.frame_size = FRAMESIZE_240X240;
config.pixel_format = PIXFORMAT_RGB565;
config.fb_count = 2;
```

Within the `loop()`, we don't need to perform conversion from RGB565 to another format or resolution. The code tells the camera to take a picture and then stores it in a frame buffer.

```
// capture from the camera into the frame buffer
Serial.printf("Capturing frame...\n");
fb = esp_camera_fb_get();
if (!fb) {
  Serial.printf("ERROR: Camera capture failed\n");
} else {
  Serial.printf("Frame capture successful!\n");
  ...
}
```

Face Detection

After a frame (photo) is successfully captured, the code performs face detection. In this example, the code runs two stages of inference.

In the first stage, inference on a model (`s1`) to detect objects is performed. The `s1.infer()` function call performs inference on the image, stored in the framebuffer (`fb`). If any objects are detected, they're stored in `candidates`.

```
Serial.printf("Frame capture successful!\n");
// Face detection
std::list<dl::detect::result_t> &candidates = s1.infer((uint16_t *)fb->buf,
{(int)fb->height, (int)fb->width, 3});
```

The second line runs another inference on a different model, `s2`. This call uses the objects detected in the first stage to then attempt to detect faces. Faces are stored in the `results` list.

```
std::list<dl::detect::result_t> &results = s2.infer((uint16_t *)fb->buf, {(int)fb->
height, (int)fb->width, 3}, candidates);
```

When a face is detected, the list of results will be non-zero. The code prints that a face has been detected. The face detection boxes and landmarks are drawn to the TFT in the `draw_face_boxes` function.

```
if (results.size() > 0) {
    Serial.println("Detected face!");
    ...
    // Draw face detection boxes and landmarks on the framebuffer
    draw_face_boxes(&rfb, &results, face_id);
    ...
}
```

Finally, the 240x240px frame buffer is drawn to the TFT display. Since the next iteration of `loop()` requires the use of the frame buffer, `fb`, to take a photo, we release it.

```
// Blit framebuffer to TFT
uint8_t temp;
for (uint32_t i = 0; i < fb->len; i += 2) {
    temp = fb->buf[i + 0];
    fb->buf[i + 0] = fb->buf[i + 1];
    fb->buf[i + 1] = temp;
}
pyCameraFb->setFB((uint16_t *)fb->buf);
tft.drawRGBBitmap(0, 0, (uint16_t *)pyCameraFb->getBuffer(), 240, 240);
// Release the framebuffer
esp_camera_fb_return(fb);
```

Face Recognition

To recognize a face, the code takes a picture and performs face detection (explained above) on the frame.

```
...
Serial.printf("Frame capture successful!\n");
// Face detection
std::list<dl::detect::result_t> &candidates = s1.infer((uint16_t *)fb->buf,
{(int)fb->height, (int)fb->width, 3});
std::list<dl::detect::result_t> &results = s2.infer((uint16_t *)fb->buf, {(int)fb->
height, (int)fb->width, 3}, candidates);
if (results.size() > 0) {
    Serial.println("Detected face!");
    ...
}
```

A structure holding the frame buffer data, `rfb`, is created and data is copied from the frame buffer (`fb`) to the new structure.

```
int face_id = 0;
fb_data_t rfb;
rfb.width = fb->width;
rfb.height = fb->height;
rfb.data = fb->buf;
rfb.bytes_per_pixel = 2;
rfb.format = FB_RGB565;
...
```

Since face recognition is a slow operation to perform. The code only attempts it if it detected a face and is enrolling the face, or if a face was previously enrolled. The `run_face_recognition` method is called with the copy of frame buffer data and the `results` from the second inference function.

```
if (recognizer.get_enrolled_id_num() > 0 || is_enrolling) {  
    face_id = run_face_recognition(&rfb, &results);  
}  
...
```

Within the `run_face_recognition` function, a vector of landmarks (discussed on the "Usage" page of this guide) is created from the inference results. Then, a `tensor` multi-dimensional array is created and shaped to fit the framebuffer's data.

```
std::vector<int> landmarks = results->front().keypoint;  
int id = -1;  
(void)id;  
  
Tensor<uint8_t> tensor;  
tensor.set_element((uint8_t *)fb->data)  
    .set_shape({fb->height, fb->width, 3})  
    .set_auto_free(false);
```

Enrolling a New Face

The function obtains the amount of faces which are currently enrolled. Then, it verifies if the MEMENTO is in enroll-mode and if the number of faces enrolled is less than the maximum.

If that condition is true, the code proceeds to enroll a new face and assign it a face identifier number. The serial and TFT print the new enrolled identifier and disables the enroll mode (`is_enrolling = false`).

```
int enrolled_count = recognizer.get_enrolled_id_num();  
if (enrolled_count < FACE_ID_SAVE_NUMBER && is_enrolling) {  
    int id = recognizer.enroll_id(tensor, landmarks, "", true);  
    Serial.printf("Enrolled ID: %d", id);  
    tft.setCursor(0, 230);  
    tft.setTextColor(ST77XX_CYAN);  
    tft.print("Enrolled a new face with ID #");  
    tft.print(id);  
    is_enrolling = false;
```

Recognizing a Face

The results of the facial recognition model, `recognizer` are stored in `recognize`, a structure containing the face similarity value (measuring the similarity between the frame's landmarks and the landmarks on the saved face).

```
face_info_t recognize = recognizer.recognize(tensor, landmarks);
```

The `recognize` similarity value weighs the image's assessed similarity value against a saved face's similarity value.

The similarity values range from `0.0` to `1.0`, where a value of `1.0` is a "confident positive match". In order to avoid a false positive (the code "recognizes" an invalid face), we added a confidence threshold. This threshold,

`FR_CONFIDENCE_THRESHOLD`, is used to detect how "confident" the match is by comparing it to the similarity value.

If the code did not have a confidence threshold, the code recognizes a face with much lower accuracy and possibly raises false positive or false negative matches.

In the code segment below, if the face is recognized (and its similarity value is larger than the `FR_CONFIDENCE_THRESHOLD` value), the MEMENTO's display shows that a face has been recognized and the NeoPixel ring lights up green.

```
if (recognize.id >= 0 && recognize.similarity >= FR_CONFIDENCE_THRESHOLD) {
    // Face was recognized, print out to serial and TFT
    Serial.printf("Recognized ID: %d", recognize.id);
    Serial.printf("with similarity of: %0.2f", recognize.similarity);
    tft.setCursor(0, 220);
    tft.setTextColor(ST77XX_CYAN);
    tft.print("Recognized Face ID #");
    tft.print(id);
    tft.print("\nSimilarity: ");
    tft.print(recognize.similarity);
    // Set pixel ring to green to indicate a recognized face
    for (int i = 0; i <= NUMPIXELS; i++) {
        pixels.setPixelColor(i, pixels.Color(0, 255, 0));
    }
    pixels.show();
    delay(2500);
}
```

If the code did not recognize a face, but other faces are enrolled, the TFT prints "Intruder alert" and the NeoPixel ring shows a red color.

```
} else if (recognizer.get_enrolled_id_num() > 0) {
    // Face was not recognized but we have faces enrolled
    Serial.println("Intruder alert - face not recognized as an enrolled face!");
    Serial.printf("This face has a similarity of: %0.2f\n",
        recognize.similarity);
    // Set pixel ring to red to indicate a unrecognized face
    for (int i = 0; i <= NUMPIXELS; i++) {
        pixels.setPixelColor(i, pixels.Color(255, 0, 0));
    }
    pixels.show();
    delay(1000);
}
...
```

Adjust the Confidence Threshold

The confidence threshold, `FR_CONFIDENCE_THRESHOLD`, determines how "confident" the face match is by comparing it to a similarity value.

If the confidence threshold value is too low, the code may incorrectly recognize a face. If the value is too high, the code may not recognize a face unless there is an exact match between the enrolled face and the image you just took.

To adjust this value in code, you will need to adjust the value of `FR_CONFIDENCE_THRESHOLD` to a number between 0.0 and 1.0, where 1.0 is an exact match:

```
// Threshold (0.0 - 1.0) to determine whether the face detected is a positive
// match NOTE - This value is adjustable, you may "tune" it for either a more
// confident match
#define FR_CONFIDENCE_THRESHOLD 0.7
```

Then, you will need to build and upload the code. This is discussed on the Modify Code using PlatformIO page of this guide.

Save Faces to Flash Memory

This code does not save the enrolled faces if the MEMENTO is rebooted. However, if you want to "lock" the code to save the enrolled faces between device reboots, you will need to set the following variable to **true**.

```
// True if you want to save faces to flash memory and load them on boot, False
// otherwise
#define SAVE_FACES_TO_FLASH false
```

Then, you will need to build and upload the code. This is discussed on the Modify Code using PlatformIO page of this guide.

Adjust the Number of Recognized Faces

The demo provided in this guide recognizes a maximum of four faces. If you want to recognize more faces, the following variable will need to be modified.

```
// The number of faces to save
// NOTE - these faces are saved to the ESP32's flash memory and survive between
// reboots
#define FACE_ID_SAVE_NUMBER 4
```

Increasing the `FACE_ID_SAVE_NUMBER` increases the amount of time required to recognize a new face. After adjusting this number, you will need to build and upload the code to the MEMENTO. This is discussed on the Modify Code using PlatformIO page of this guide.

Restoring MEMENTO Demo and UF2 Bootloader

You're probably used to seeing the **CAMERABOOT** drive when loading CircuitPython or Arduino. The **CAMERABOOT** drive is part of the UF2 bootloader and allows you to drag and drop files, such as CircuitPython. However, the application in this guide uses a large partition size, which overwrites the UF2 bootloader present on the MEMENTO.

This means that after installing the facial detection application, double-tapping the RESET button will not put your MEMENTO into bootloader mode.

However, you can get your MEMENTO back to "normal" by writing a new binary application to the board that repairs your board's UF2 bootloader and performs a factory reset:

Instructions for MEMENTO Factory Reset and Bootloader Repair

<https://adafru.it/19ft>

Modify Code using PlatformIO

Modifying this code using PlatformIO is for advanced users only.

This project library brings in a considerable number of dependencies and takes a looong time to compile using Arduino IDE. If you want to modify the code in this guide, you'll want to compile the project using PlatformIO rather than Arduino IDE.

Install PlatformIO

Follow this page's instructions to install PlatformIO and Visual Studio Code (the IDE of choice for using PlatformIO).

Download and Install PlatformIO

<https://adafru.it/19cu>

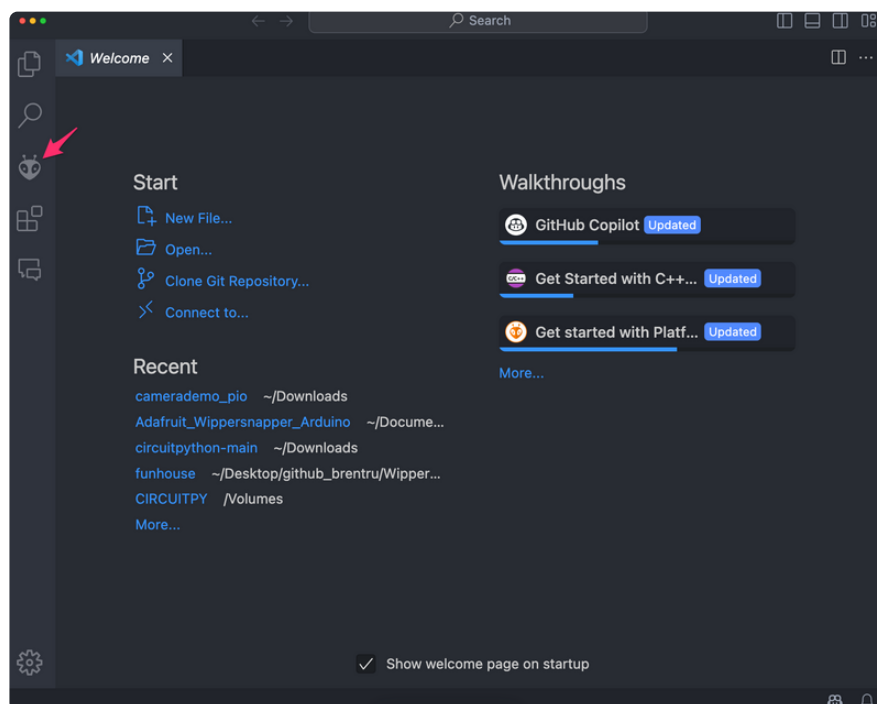
Configure Your Workspace

The ZIP file below includes a pre-configured workspace for using PlatformIO. **Download and unzip this file.** Then, save it somewhere safe, like your desktop.

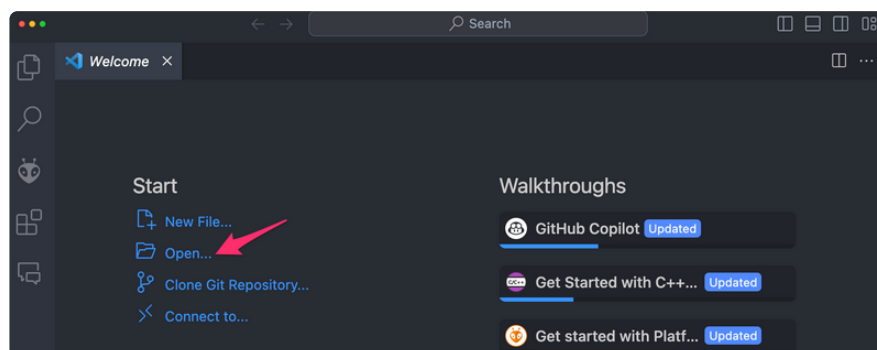
Download Project Code and PlatformIO Environment

<https://adafru.it/19fs>

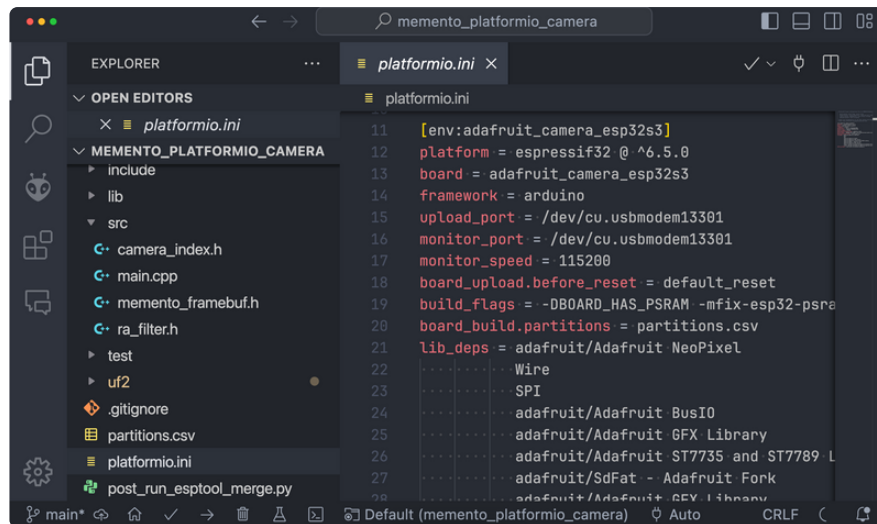
Open Visual Studio Code (VSCode). To ensure you have installed the PlatformIO extension properly, **look for the alien symbol in your VSCode sidebar.**



Underneath Start, click **Open...**



Navigate to the folder created when you unzipped the zip file. Then, Click Open to open the workspace.



A large amount of configuration files and directories will appear in your VSCode instance.

To compile this code, we are only going to discuss the following files and directories:

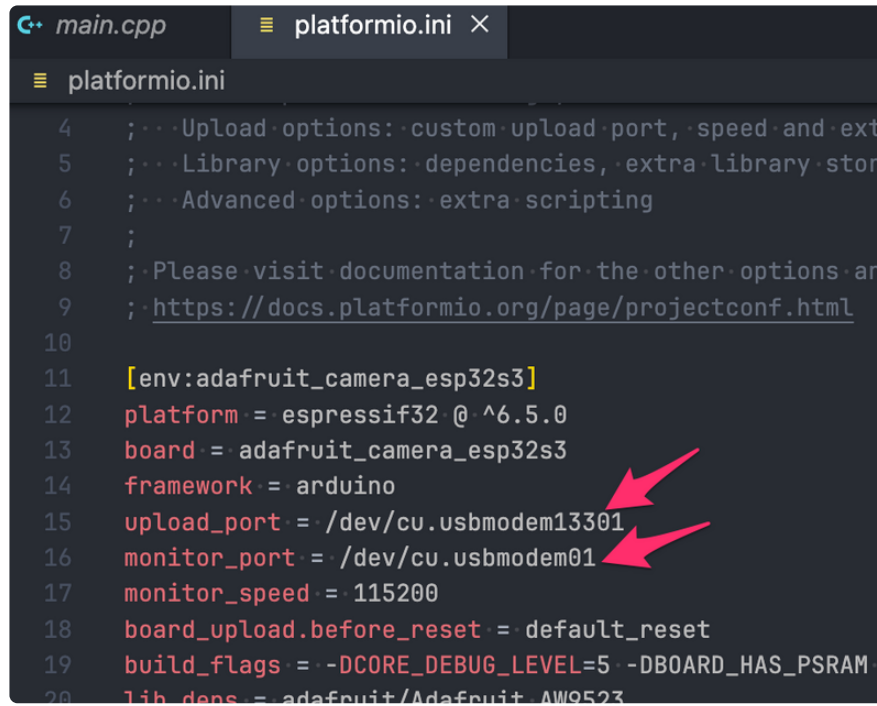
- **platformio.ini** - This is the project configuration file used to build the demo code. More [documentation about this file is located here \(https://adafru.it/19cw\)](https://adafru.it/19cw).
- **lib** directory - This directory is intended for project-specific (private) libraries. PlatformIO will compile them to static libraries and link them into executable files.
 - For our project, the specific library within this directory is the [Adafruit_PyCamera \(https://adafru.it/19cx\)](https://adafru.it/19cx) library.
- **src** directory - The directory where the project's source code, **main.cpp**, is located as well as the included headers (such as **ra_filter.h** which stores the facial recognition/detection overhead).

Build and Upload with PlatformIO

Before the code is built, you'll need to make two changes to the platformio.ini file:

- Change **upload_port** to reflect the MEMENTO's upload port.
 - Don't know the desired port? We have steps to find them for [Windows \(http://adafru.it/19cy\)](http://adafru.it/19cy), [MacOS \(https://adafru.it/19cz\)](https://adafru.it/19cz), and [Linux \(https://adafru.it/19cA\)](https://adafru.it/19cA).

- The `monitor_port` is different from the `upload_port`, and will only appear on your computer when you've uploaded the test code. For now, leave this alone.
 - After uploading the test code, change `monitor_port` to reflect the MEMENTO's monitor/serial port.

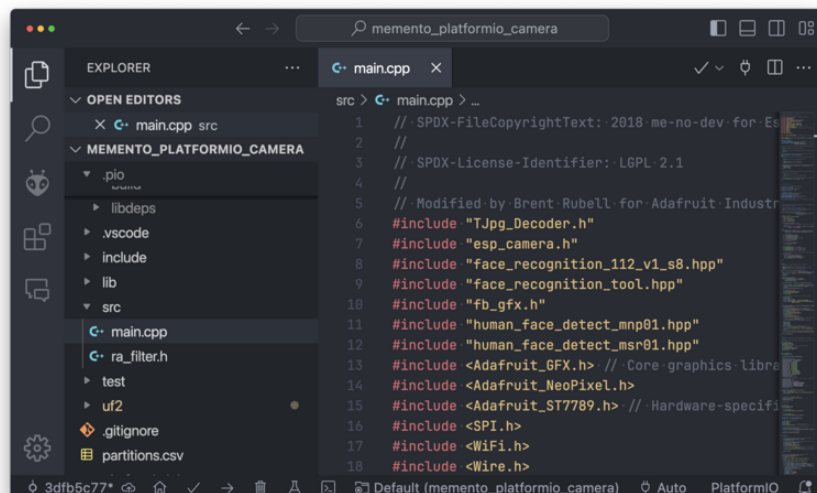


```

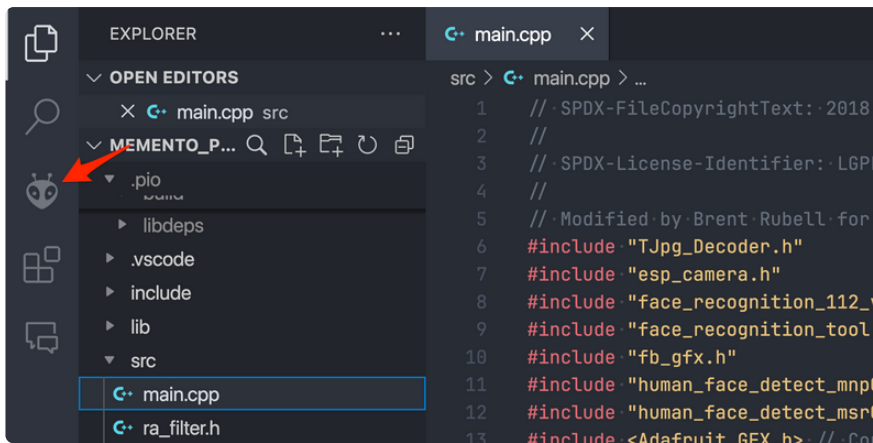
platformio.ini

4 ;... Upload options: custom upload port, speed and extra
5 ;... Library options: dependencies, extra library storage
6 ;... Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:adafruit_camera_esp32s3]
12 platform = espressif32 @ ^6.5.0
13 board = adafruit_camera_esp32s3
14 framework = arduino
15 upload_port = /dev/cu.usbmodem13301
16 monitor_port = /dev/cu.usbmodem01
17 monitor_speed = 115200
18 board_upload.before_reset = default_reset
19 build_flags = -DCORE_DEBUG_LEVEL=5 -DBOARD_HAS_PSRAM
20 lib_deps = adafruit/Adafruit_AW9523
  
```

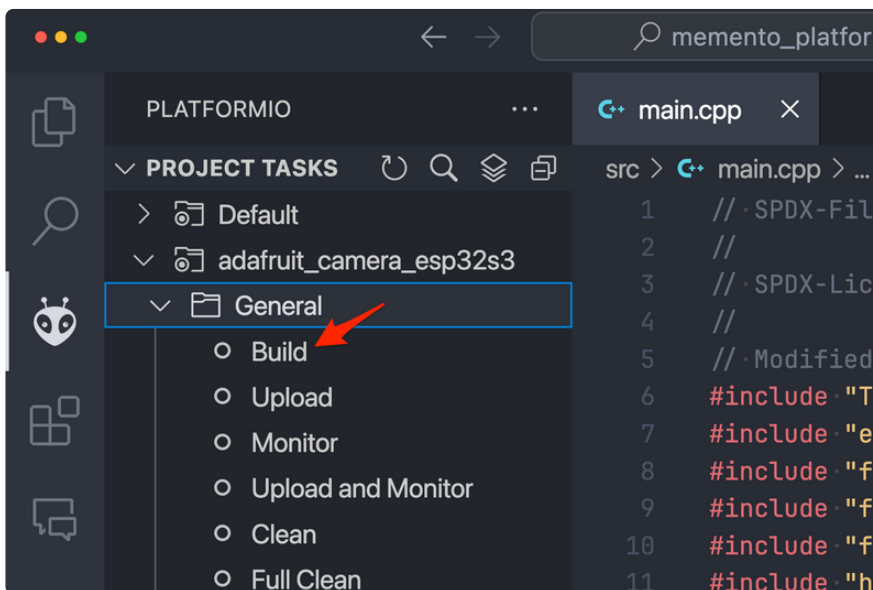
Navigate to `src/main.cpp` to open the example code.



With this file open, click the Alien symbol on the VSCode sidebar to open the PlatformIO Project Explorer.



Underneath PlatformIO's Project Tasks, click **Build**.

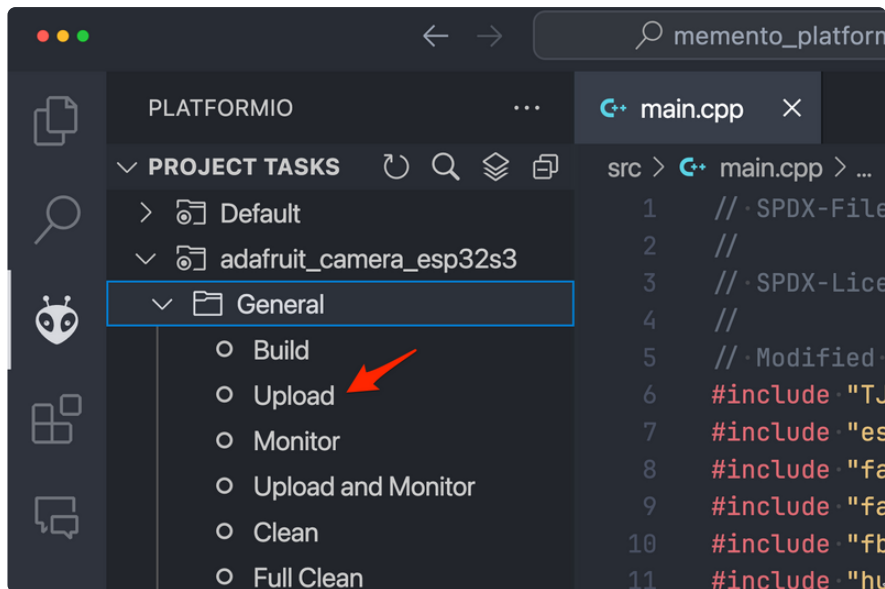


Once the build task is completed, the terminal will show **SUCCESS** along with the time it took to compile the project.

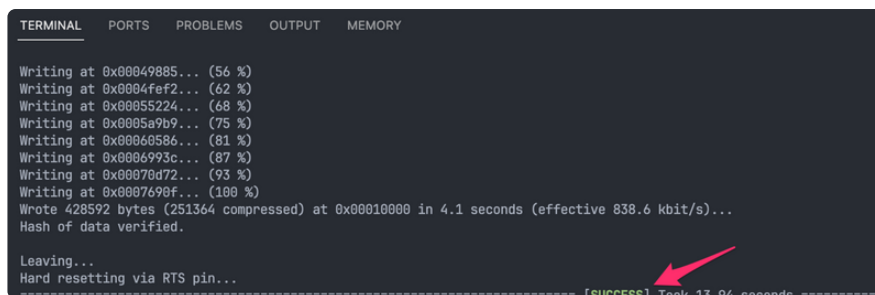
```
Checking size .pio/build/adafruit_camera_esp32s3/firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [==          ] 15.5% (used 50720 bytes from 327680 bytes)
Flash: [=====    ] 62.8% (used 2511073 bytes from 3997696 bytes)
===== [SUCCESS] Took 4.07 seconds =====
```

Before uploading this project to the board, [put the board into ROM Bootloader Mode \(https://adafru.it/19cB\)](https://adafru.it/19cB).

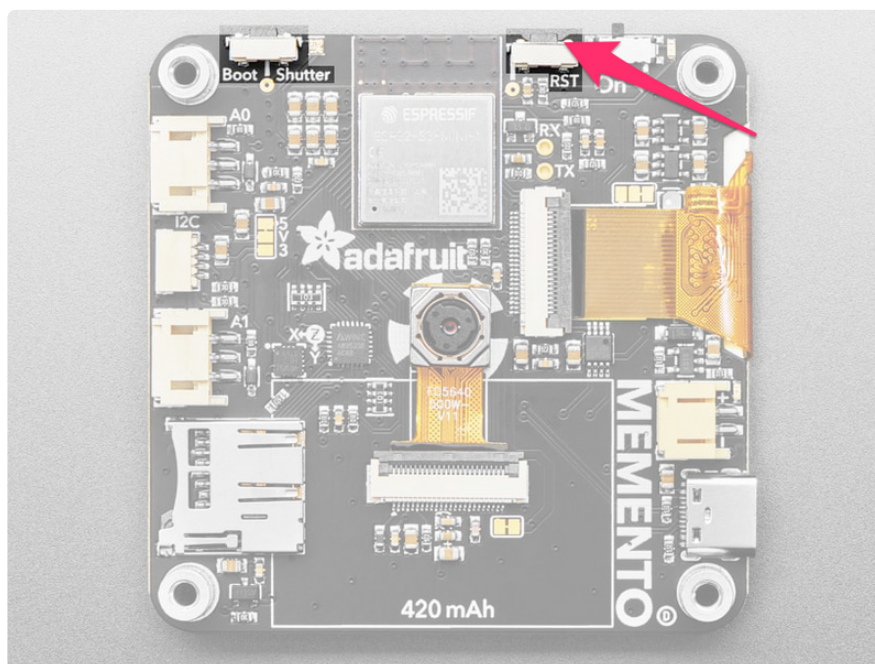
From the PlatformIO Project Tasks menu, click **Upload**.



Once the upload completes, the terminal should look like the following screenshot and show **SUCCESS**.



Press the RST (Reset) button on the MEMENTO to run the uploaded code.



After the board resets, you'll see a preview of what the camera module is seeing on the MEMENTO display. Follow the "Usage" page in this guide for detailed usage instructions.

Congrats - you have successfully compiled and uploaded the example code. You may make any modifications or extensions to this code that you'd like!