Face Tracking Robot with MEMENTO

Created by Brent Rubell

https://learn.adafruit.com/face-tracking-robot-with-memento

Last updated on 2024-06-03 03:58:57 PM EDT

Table of Contents

Overview

In a previous guide, we created a computer vision system to detect and recognize faces using the MEMENTO Camera (https://adafru.it/19fv).

This guide builds upon the computer vision system from the previous guide to build a playful robot that uses the Adafruit MEMENTO to capture an image, detect a face, and track it until the face is no longer in the frame.

About the code used in this guide

This project uses an example written by Me-No-Dev for Espressif Systems, CameraWebServer.ino (https://adafru.it/19fj). This example was modified by the author of this guide to reduce the flash size overhead.

To do so, this project removed the web functionality, isolated the face detection / recognition calls, brought the overhead into **main.cpp** and **ra_filter.h.** It adds compatibility for the Adafruit MEMENTO development board (added camera

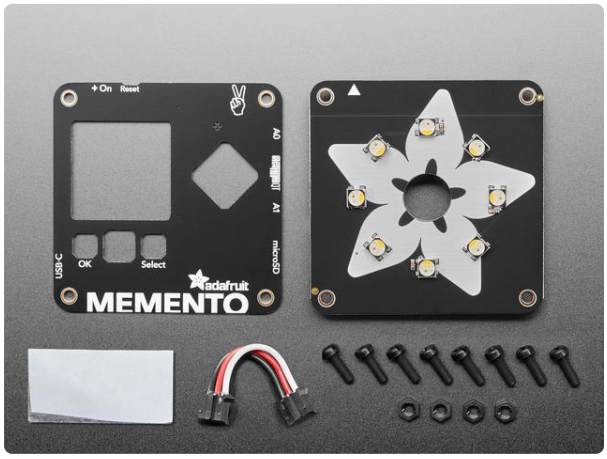
compatibility and added "blitting" the camera's raw image to the MEMENTO's TFT instead of to a webpage), sets up a PlatformIO build environment to decrease compile time, and creates an interactive robotics demo around the code.

Parts

MEMENTO - Python Programmable DIY Camera - Bare Board

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...

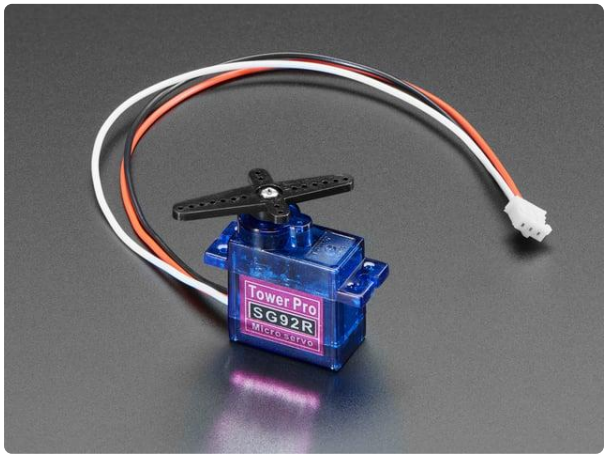
https://www.adafruit.com/product/5420



Adafruit MEMENTO Camera Enclosure & Hardware Kit

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



Micro Servo with 3-pin JST PH 2mm Cable - TowerPro SG92R

This tiny little servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds you're used to but smaller. You can use any...

https://www.adafruit.com/product/4326

1 x USB A to USB C Cable https://www.adafruit.com/product/5153

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

256MB Micro SD Memory Card

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

256MB Micro SD Memory Card

---

1 x 3.7V 420mAh Lithium Ion Polymer Battery https://www.adafruit.com/product/4236

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

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:

1 x NeoPixel Ring with 12x RGBW LEDs https://www.adafruit.com/product/2852

NeoPixel Ring - 12 x 5050 RGBW LEDs w/ Integrated Drivers - Natural White - ~4500K

---

1 x JST PH 3-pin Plug-Plug Cable https://www.adafruit.com/product/4336

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

---

1 x Magnetic Pin Back https://www.adafruit.com/product/1170

Magnetic Pin Back

---

1 x Jumper Wires Socket and Plug-Plug https://www.adafruit.com/product/4635

Jumper Wires Socket and Plug-Plug

---

1 x Camera and Tripod 3/8" to 1/4" Adapter Screw https://www.adafruit.com/product/2392

Camera and Tripod 3/8" to 1/4" Adapter Screw

---

4 x M3x6mm Screws

M3x6mm Screws

---

4 x M2.5x6mm Screws

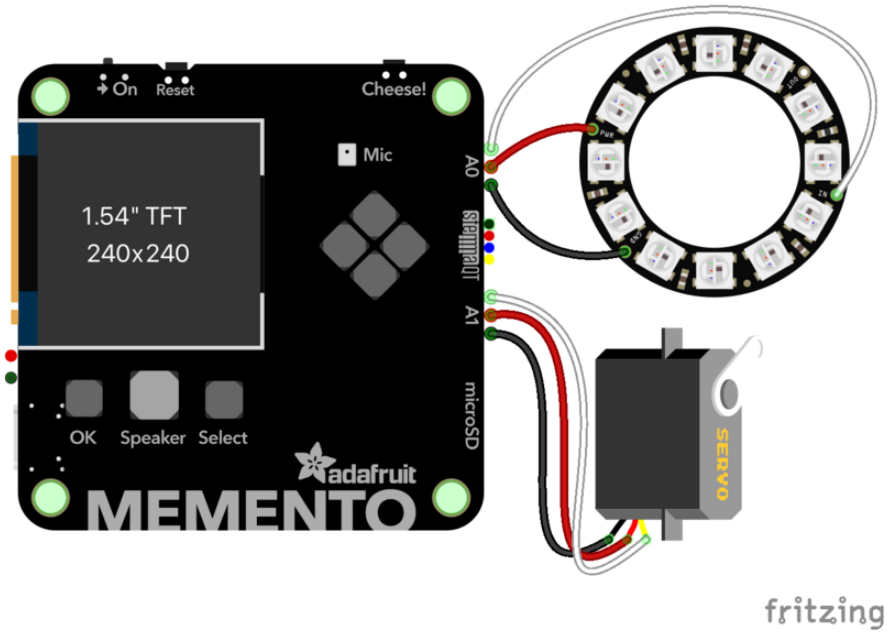
M2.5x6mm Screws

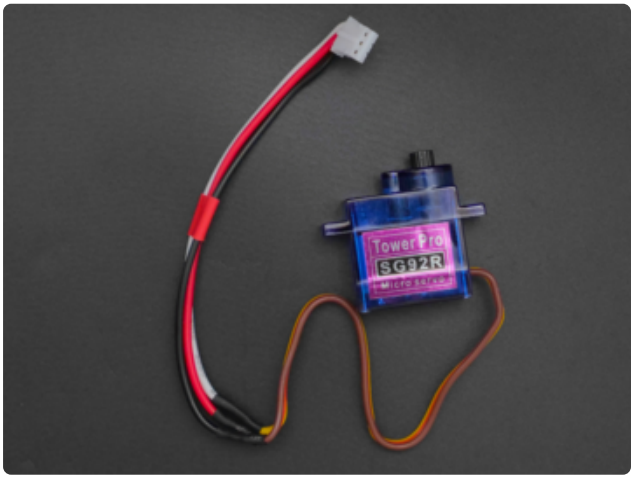
---

3 x M3x20mm Screws

M3x20mm Screws

Wiring





Wiring the Servo

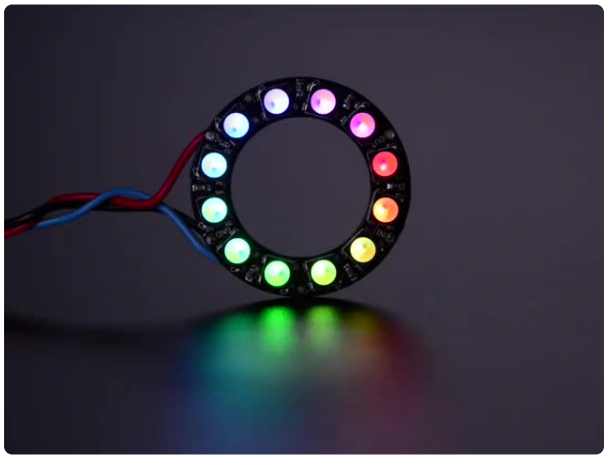
For wiring the Micro Servo with 3-pin JST PH (http://adafru.it/4326), connect the servo's 3-PIN JST connector to the receptacle labeled A1 on the MEMENTO.

Wiring the NeoPixel Ring using the MEMENTO Camera Enclosure Kit

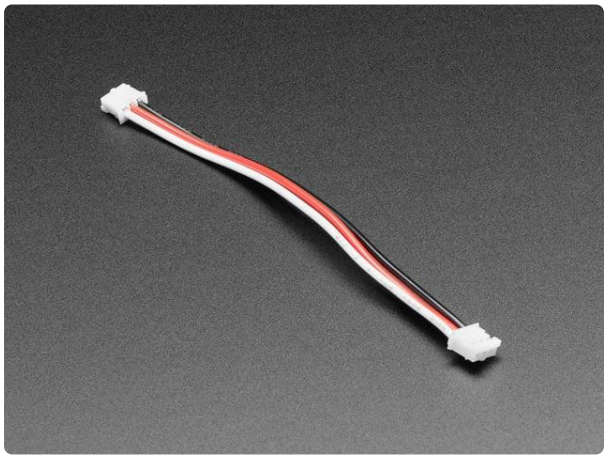
If you are using the Adafruit MEMENTO Camera Enclosure Kit (http://adafru.it/5843) with this guide, use the included JST cable to connect the Adafruit MEMENTO Camera LED Ring Front Plate 3-pin JST Connector and the 3-pin JST connector labeled **A0** on the MEMENTO.

Wiring the NeoPixel Ring without the MEMENTO Camera Enclosure Kit

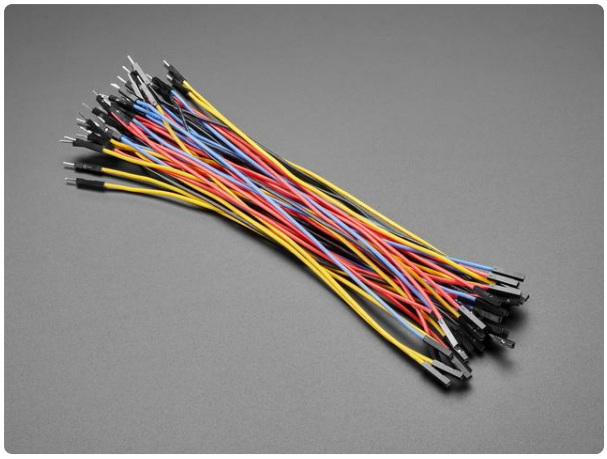
If you do not have the MEMENTO Camera Enclosure Kit (or if it is out of stock), you can build your ring light for the MEMENTO using the following parts:



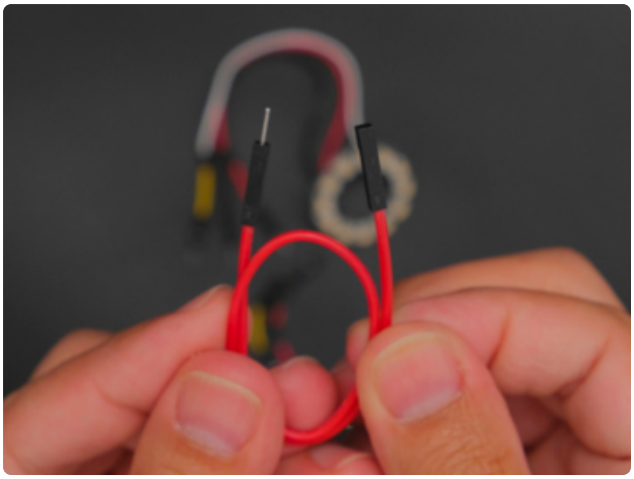
NeoPixel Ring - 12 x 5050 RGBW LEDs w/ Integrated Drivers
What is better than smart RGB LEDs? Smart RGB+White LEDs! These NeoPixel rings now have 4 LEDs in them (red, green, blue and white) for excellent lighting effects. Round and...
https://www.adafruit.com/product/2852



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



[Premium Silicone Covered Extension Jumper Wires - 200mm x 40]
These premium extension jumper wires are handy for making wire harnesses or jumpering between headers on PCBs. They're 200mm (~7.8") long and come loose as a pack of...
https://www.adafruit.com/product/4635



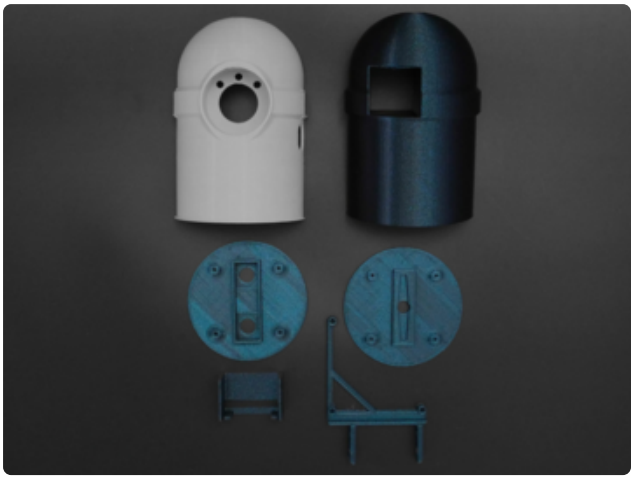
Use socket and plug Jumper wires to easily connect the Neopixel ring when mounting to the robot enclosure.

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:

Make the following connections between the MEMENTO and the NeoPixel ring:

- MEMENTO **A0 VCC** to NeoPixel ring **PWR**
- MEMENTO **A0 GND** to NeoPixel ring **GND**
- MEMENTO **A0 Data** to NeoPixel ring data **IN**

3D Printing



3D Printed Parts

STL files for 3D printing are oriented to print "as-is" on FDM style machines.

Parts are designed to 3D print without any support material using PLA filament.

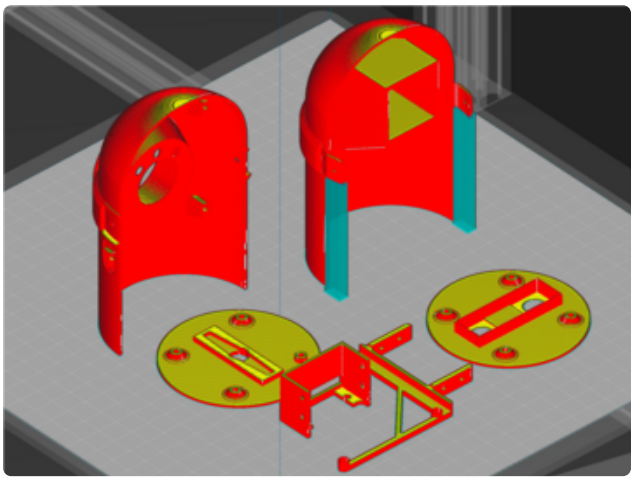
Original design source files may be downloaded using the links below.

Download STLs

https://adafru.it/19mc

Edit Design

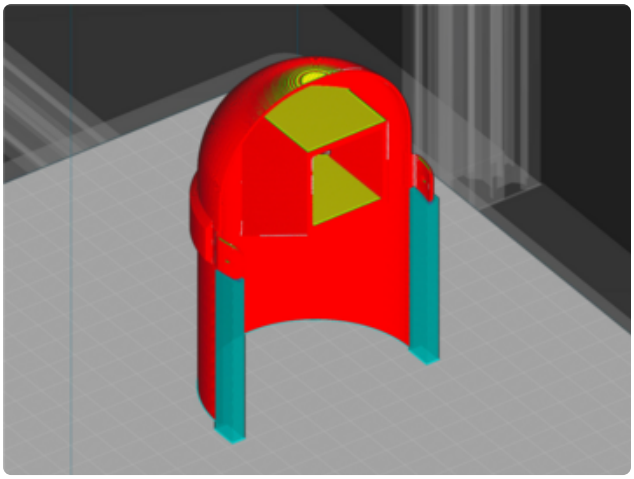
https://adafru.it/19md



Slice with Settings for PLA Material

The parts were sliced using CURA with the slice settings below.

PLA filament 200c extruder
0.25 layer height
10% gyroid infill
60mm/s print speed
60c heated bed
Brim line count 2



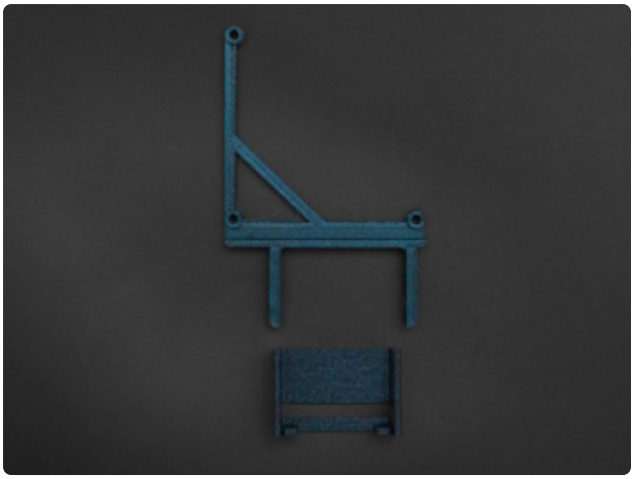
Supports

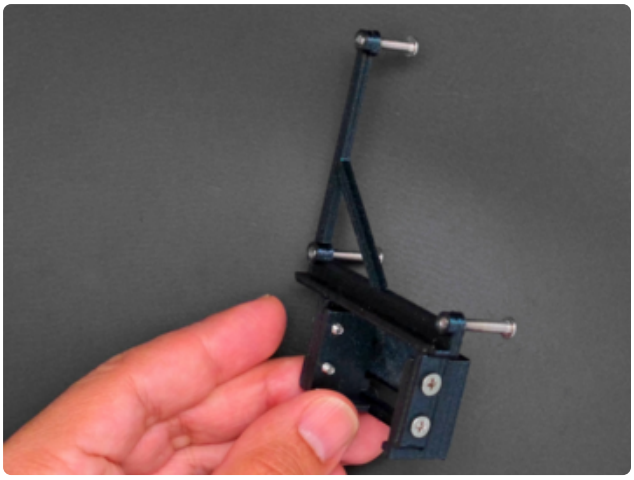
Support Overhang Angle: 50
Support Destiny: 6%
Enable Support Interface
Enable Support Roof
Support Z Distance: .21

Assembly

Mount LED ring

Thread the jumper wires through the cut outs in the circular mount. The LED ring PCB press fits inside the body part.

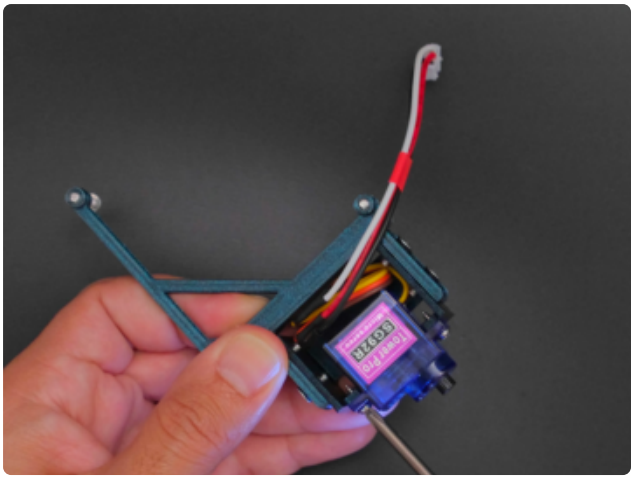




Frame assembly

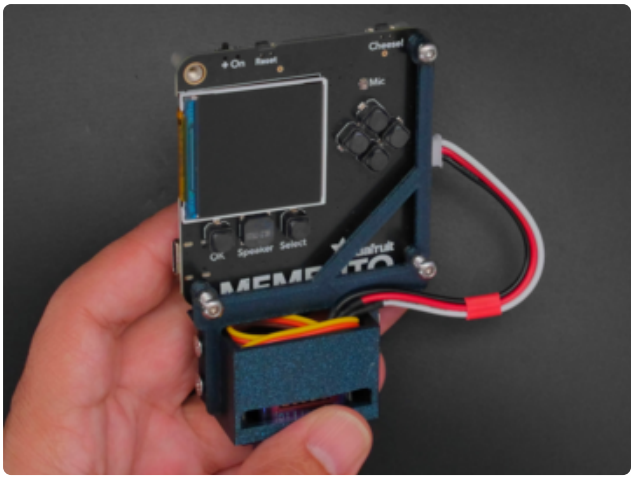
Use four M3x6mm to connect the servo holder part to the frame part.

The frame mounts to the MOMENTO board with three M3x20mm long screws.



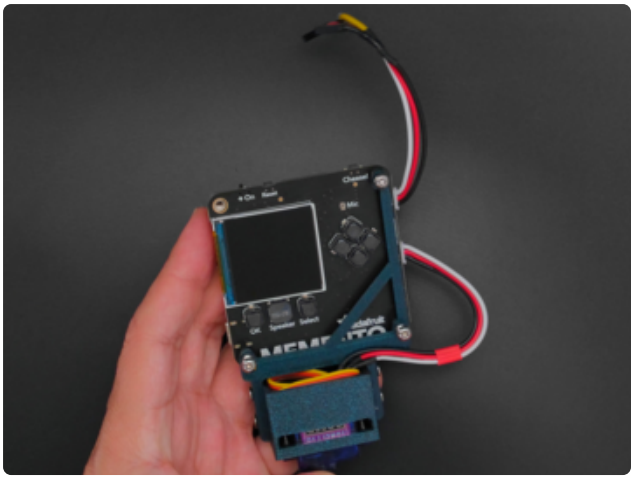
Mount Servo

Use two screws to mount the servo to the holder.



Attach frame assembly

The frame aligns and mounts to the three mounts on the MEMENTO PCB.

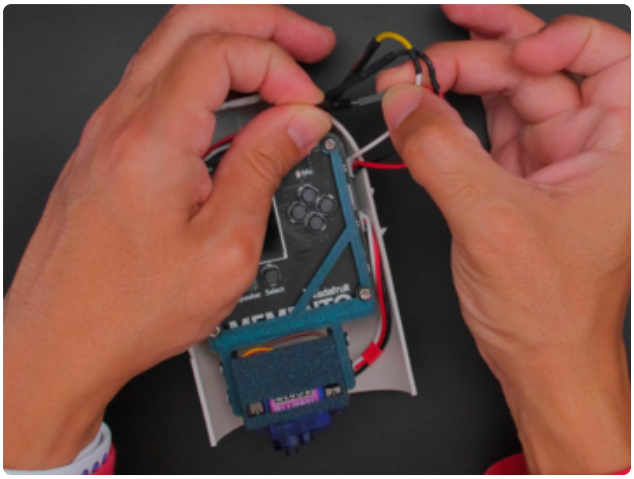


Connect JST cables

Connect the servo and LED ring JST cables to the ports on the MEMENTO.

Mount frame assembly

The frame assembly mounts to the standoffs on the body print.

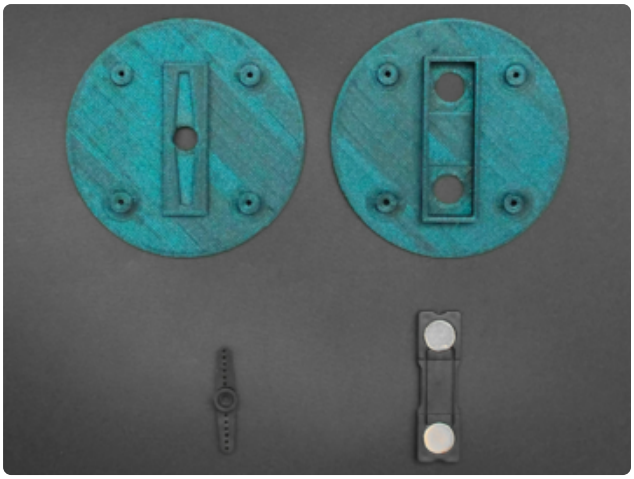


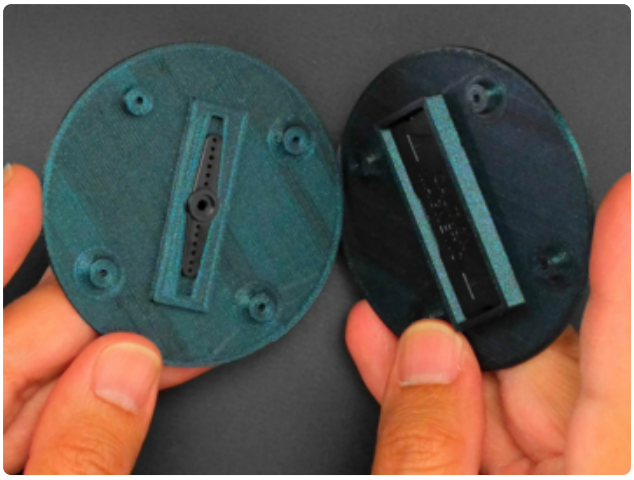
Connect jumper wires

The LED jumper socket and pin wires connect and then are tucked inside the body.

Attach back body

Align the back body print with the tab holes on the sides.



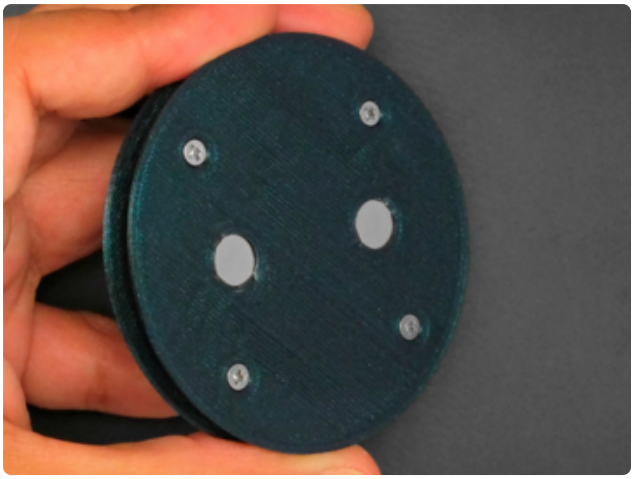


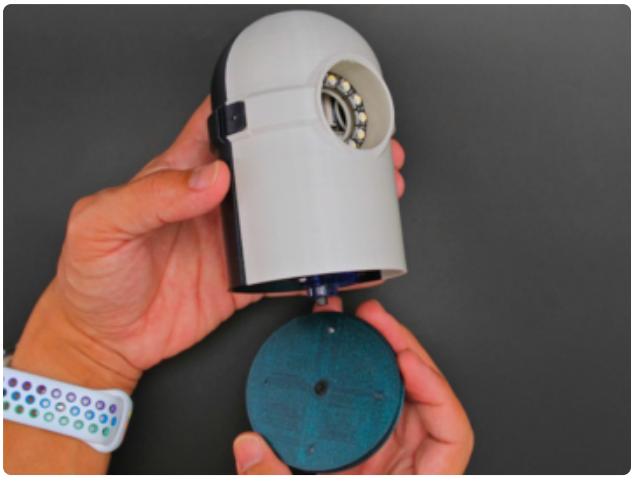
Base assemble

The base plate houses the servo horn and magnet clips.

Press fit the included servo horn into the cutout on the base plate.

Align the magnet bar to the cutouts on the rectangular base plate. The magnet bar part press fits into place so the magnets are flush with the cutouts.



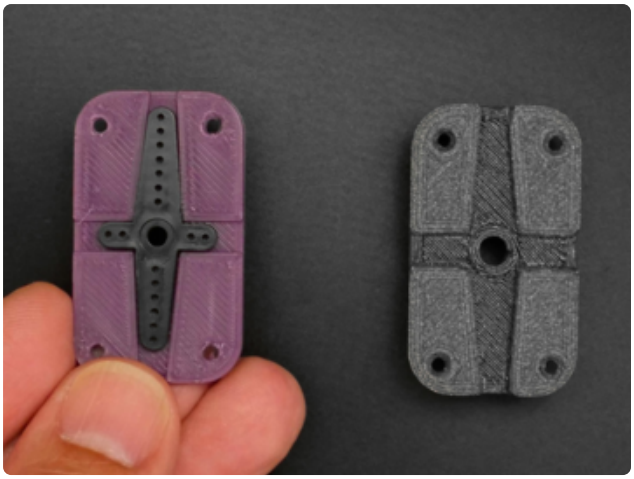


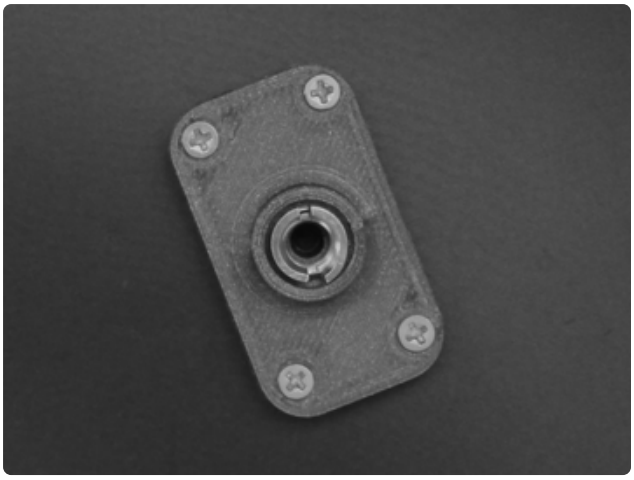
Combine base plates

Use four M2.5x6mm long screws to attach both base plates.

Mount with magnets

The magnets are strong enough to hold the robot on metal surfaces. A ferromagnetic bar is used to help mount to a shoulder.



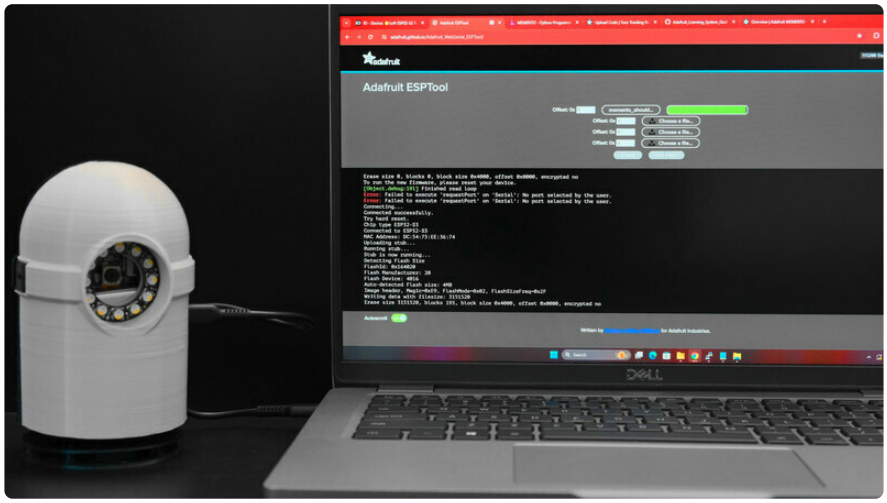


Tripod attachment

The tripod attachment uses the included servo horn and 3/8 to 1/4-20 tripod thread.

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.

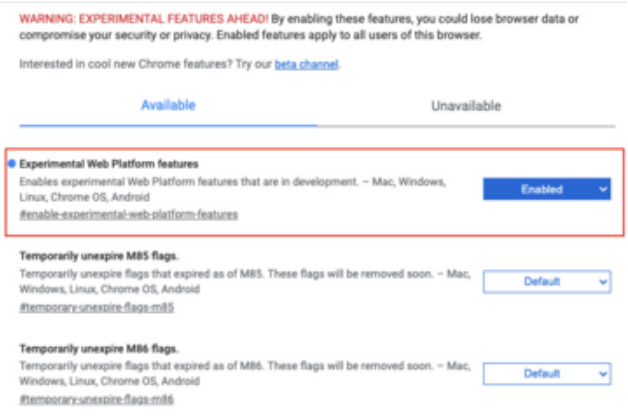
For boards that lack native USB, like the ESP32 or ESP32-C3 microcontroller, this is how firmware like CircuitPython **.bin** files can be loaded. **There is no drag-and-drop to a folder option for these boards.**

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). Safari and Firefox, etc are not supported - they have not implemented Web Serial (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

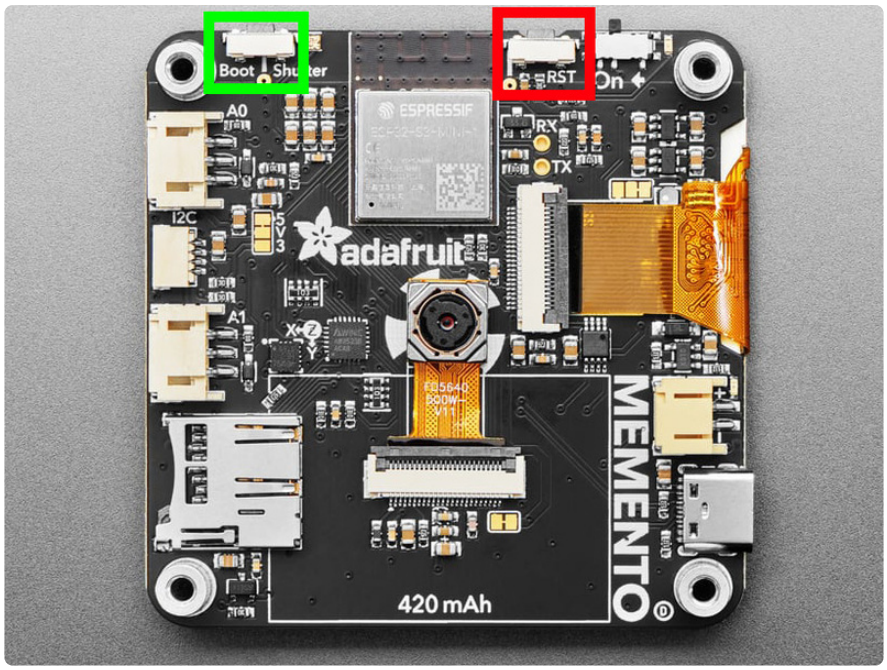
Enter Bootloader Mode

Before you can use the tool, you will need to put your board in bootloader mode. **Before you start, make sure your MEMENTO 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

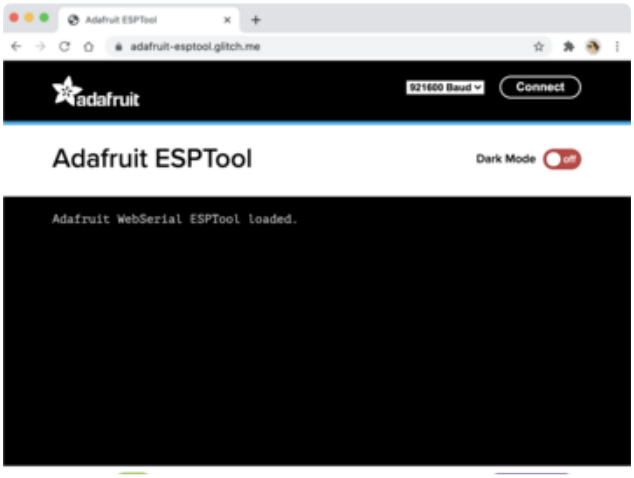
Click the button below to download the application binary file for your board.

Download firmware for MEMENTO Face Tracking Robot

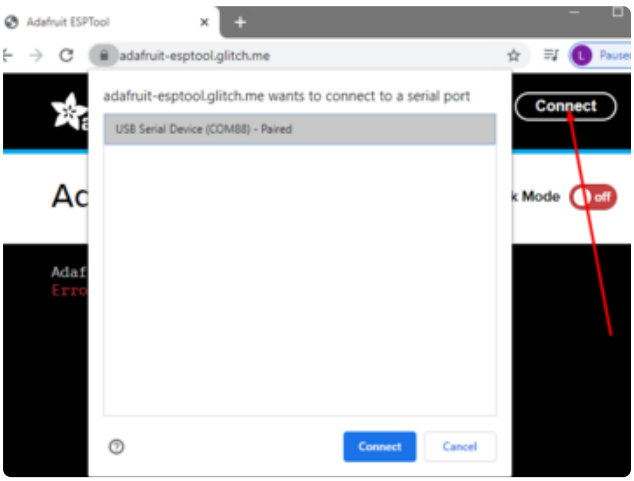
https://adafru.it/19me

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.



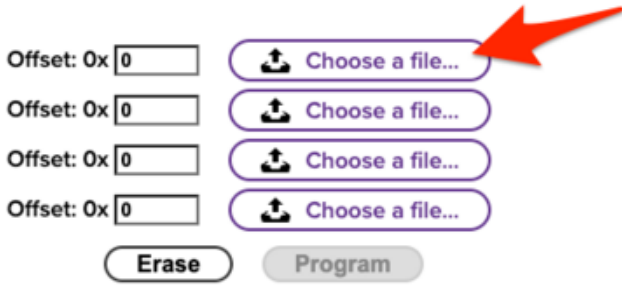
Press the **Connect** button in the top right of the web browser. You will get a pop-up asking you to select the COM or Serial port.

Remember, you should remove all other USB devices so only the MEMENTO 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.

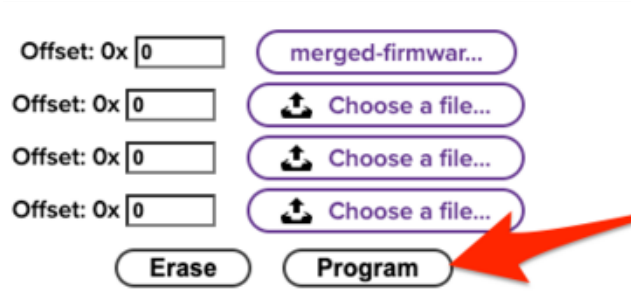
```
ESP Web Flasher loaded.
Connecting...
Connected successfully.
Try hard reset.
Chip type ESP32-S2
Connected to ESP32-S2
MAC Address: 7C:DF:A1:06:8D:D0
Uploading stub...
Running stub...
Stub is now running...
Detecting Flash Size
FlashId: 0x164020
Flash Manufacturer: 20
Flash Device: 4016
Auto-detected Flash size: 4MB
```

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

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

The display shows the overlay when a face is detected. Note robots do not work!

Move in front of the camera and make sure your face is fully visible.

When the robot detects your face, the NeoPixel ring will light up.

The robot will track your face, following it left and right.

When a face is no longer detected, the robot will move its head back to a resting position and turn off its ring light.

What am I seeing? How does Face Detection Work?

For more details about how the MEMENTO's TFT display illustrates how the computer vision for this project works, read through this section of a previous guide (https://adafru.it/19mf).

Troubleshooting

Uneven Lighting

One area we noticed the camera had issues with is uneven lighting. The author of this guide has a window parallel to his desk which caused uneven lighting conditions. The camera would track halfway, but stop when the face became overly saturated by light from the window.

We attempted to correct this by gently illuminating the face with a NeoPixel ring. However, you may find this is not enough. Putting a light source behind the camera will even out the lighting in your room/office/workshop.

Increasing Detection Accuracy

One of the ways to increase this project's face detection accuracy is to find a "sweet spot" in the bounding boxes width. When the robot detects a face, it prints the dimensions of the bounding box, in pixels, to the Serial port (you can read this using the Arduino Serial Monitor or a similar tool). Since the width of the frame is 240 pixels, you may find moving further (or closer) to the camera increases detection accuracy.

Once you have the robot accurately tracking your face, write down the dimensions of the bounding box. You can also put a piece of painter's tape on the floor to mark the ideal distance between you and the robot.

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). 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**), added 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 robotics 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** as they pertain to this project.

Capturing and Detecting a Photo

The `loop` function's first call is to `performFaceDetection()`, a function that captures a frame from the camera and performs face detection on it.

Within `performFaceDetection()`, a photo (aka, a "frame") is captured from the camera and stored into a buffer (`fb = esp_camera_fb_get()`). Then, two-stages of inference are run on this frame to attempt detecting a face.

Within 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 a list, `candidates` .

```
std::list<dl::detect::result_t> &candidates = s1.infer((uint16_t *)fb->buf,
{(int)fb->height, (int)fb->width, 3});
```

The second line runs inference on a different model, `s2` . This inference attempts to differentiate unique faces from the generic objects detected in `s1` . Then, the faces are stored in a list, `results` , which is returned back to the `loop()` function.

```
std::list<dl::detect::result_t> performFaceDetection() {
  // Capture a frame from the camera into the frame buffer
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.printf("ERROR: Camera capture failed\n");
    return std::list<dl::detect::result_t>();
  }

  // Perform 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);
  return results;
}
```

Did it Detect a Face?

If `performFaceDetection()` detected a face, the `detectionResults` list will have a non-zero size. In the `loop()` , upon successful detection of a face, the NeoPixel ring will fill with a blue color to indicate to the user that the robot is tracking a face.

```
if (detectionResults.size() > 0) {
    // Fill NeoPixel ring with a blue color while tracking
    pixels.fill(pixels.Color(0, 0, 255), 0, pixels.numPixels());
    pixels.show();
    ....
```

If a face was not detected in the past 3 seconds (the value of `DELAY_SERVO_CENTER` , in seconds), the robot "resets" itself by returning the head servo to its center position, adjusting the bounding box location to point to the center of the frame, and turning of fthe blue NeoPixel ring.

```
} else {
    // We aren't tracking a face anymore
    isTrackingFace = false;

    // No face has been detected for DELAY_SERVO_CENTER seconds, re-center the
    // servo
    if ((millis() - prvDetectTime) > DELAY_SERVO_CENTER) {
      Serial.println("Lost track of face, moving servo to center position!");
```

```
      curServoPos = SERVO_CENTER;
      headServo.write(curServoPos);

      // Reset the previous detection time
      prvDetectTime = millis();

      // Re-center the bounding box at the middle of the TFT
      prv_face_box_x_midpoint = 120;

      // Clear the NeoPixels while we're not tracking a face
      pixels.clear();
      pixels.show();
    }
```

"Debouncing" the Face Detection

The `performFaceDetection()` function executes very quickly and we want to detect and track a new face. To do this, the code segment below allows the code to avoid re-triggering on every face by implementing a time delay of `DELAY_DETECTION` seconds.

```
if ((!isTrackingFace) && ((millis() - prvDetectTime) > DELAY_DETECTION)) {
  Serial.println("Face Detected!\nTracking new face...");
  isTrackingFace = true;
}
```

Drawing to the TFT

The code prints text to the TFT indiciating that a face is tracked by the robot. Then, the `draw_face_boxes()` function is called with the frame buffer, its meta-data, and the `detectionResults` list.

```
/// Write to TFT
tft.setCursor(0, 230);
tft.setTextColor(ST77XX_GREEN);
tft.print("TRACKING FACE");

// Draw face detection boxes and landmarks on the framebuffer
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;
draw_face_boxes(&rfb, &detectionResults);
...
```

Within `draw_face_boxes()`, a bounding box is drawn around the face to illustrate to the user what the code "sees".

```
// draw a bounding box around the face
tft.drawFastHLine(x, y, w, color);
tft.drawFastHLine(x, y + h - 1, w, color);
tft.drawFastVLine(x, y, h, color);
```

```
tft.drawFastVLine(x + w - 1, y, h, color);
Serial.printf("Bounding box width: %d px\n", w);
Serial.printf("Bounding box height: %d px\n", h);
```

The center/midpoint of the bounding box is then calculated. The code uses this new center point to compare against the previous center point, as a way to measure how much the face moved (and in what direction on the X-axis).

```
// Calculate the current bounding box's x-midpoint so we can compare it
// against the previous midpoint
cur_face_box_x_midpoint = x + (w / 2);

// Draw a circle at the midpoint of the bounding box
tft.fillCircle(cur_face_box_x_midpoint, y + (h / 2), 5, ST77XX_BLUE);
```

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

```
// Blit out the framebuffer to the 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);
```

Tracking the Face's Movement and Moving the Robot's Head

The `trackFace()` function handles moving the robot's head servo. Specifically, it handles calculations to answer the following questions:

1) Should we move the servo at all?

2) How much to move the servo by

3) In what direction should we move the servo?

The `trackFace()` function first checks if the bounding box has moved by comparing the current bounding box's midpoint position on the x-axis against the previous bounding box's x-axis midpoint value.

```
void trackFace() {
  // Check if the bounding box has moved and if this is the first frame with a
  // face detected, just save the coordinates
```

```
  if ((cur_face_box_x_midpoint != prv_face_box_x_midpoint) &&
      (prv_face_box_x_midpoint != 0)) {
      ...
```

If the face has moved, the code calculates the difference between both bounding box midpoints, in pixels.

```
Serial.printf("x_midpoint (curr. face):      %d px\n",
cur_face_box_x_midpoint);
Serial.printf("x_midpoint (prv. face):       %d px\n",
prv_face_box_x_midpoint);

// Calculate the difference between the new bounding box midpoint and the
// previous bounding box midpoint, in pixels
int mp_diff_pixels = abs(cur_face_box_x_midpoint - prv_face_box_x_midpoint);
Serial.printf("Difference between midpoints: %d px\n", mp_diff_pixels);
```

If the servo moved whenever a difference between the midpoints was detected, it'd be very "jittery". To smooth its motion, the code only moves the servo if it's past an adjustable `SERVO_HYSTERESIS` value of 2 pixels.

Then, the proportional amount of degrees the servo should move is calculated by multiplying the difference between the midpoints, in pixels, with a `SERVO_MOVEMENT_FACTOR` value. The resulting `servoStepAmount` degrees is more accurate than moving the servo a fixed amount of steps and results in smoother motion.

We've also implemented the concept of "dead zones" to smooth the servo's motion even further. These `dead_zones` are calculated as +/-10px from the center of the bounding box. If the bounding box's x-midpoint value is past the deadzone on either the right or the left, the servo's postion is incremented (moving left) or decremented (moving left).

```
// Only move the servo if the magnitude of the difference between the
// midpoints is greater than SERVO_HYSTERESIS
// NOTE: This smooths the servo's motion to avoid the servo from
// "jittering".
if (mp_diff_pixels > SERVO_HYSTERESIS) {
  // Calculate how many steps, in degrees, the servo should move
  int servoStepAmount = mp_diff_pixels * SERVO_MOVEMENT_FACTOR;

  // Move the servo to the left or right, depending where x_midpoint is
  // located relative to the dead zones
  if (cur_face_box_x_midpoint < deadzoneStart)
    curServoPos += servoStepAmount;
  else if (cur_face_box_x_midpoint > deadzoneEnd)
    curServoPos -= servoStepAmount;
}
```

The absolute `curServoPos` value is calculated (so we don't step in a negative direction, past the physical limitation of the servo) and is written to the servo.

```
// Move the servo to the new position
if (curServoPos != prvServoPos) {
curServoPos = abs(curServoPos);
Serial.printf("Moving servo to new position: %d degrees\n", curServoPos);
headServo.write(curServoPos);
}
```

Finally, the values of `prv_face_box_x_midpoint` and `prvServoPos` are updated to reflect the current position of the servo and the frame.

```
}
  // Update the previous midpoint coordinates with the new coordinates
  prv_face_box_x_midpoint = cur_face_box_x_midpoint;
  // Save the current servo position for the next comparison
  prvServoPos = curServoPos;
}
```

Going Further - Tuning and Tweaking

This guide's code was tested in the guide author's office. The ML model was not trained on the guide author and it was not created by us. So, the accuracy of your robot's face detection may vary due to a large amount of factors such as room lighting, distance from the camera, camera's field of view in your environment, etc. You may find that modifying and fine-tuning the `SERVO_HYSTERESIS` and `SERVO_MOVEMENT_FACTOR` values results in smoother overall motion.

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 the 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

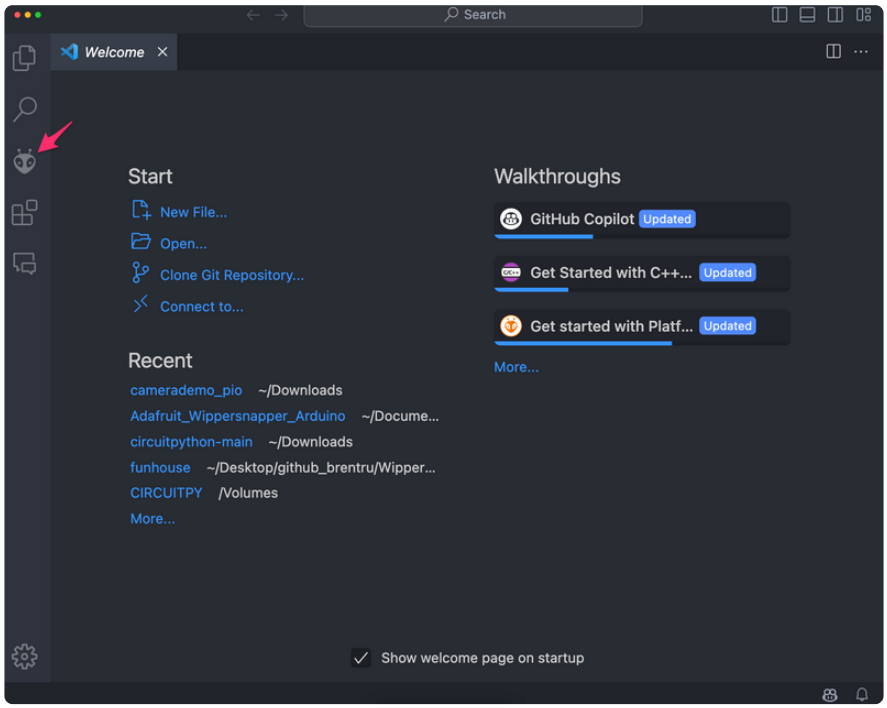
Configure Your Workspace

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.

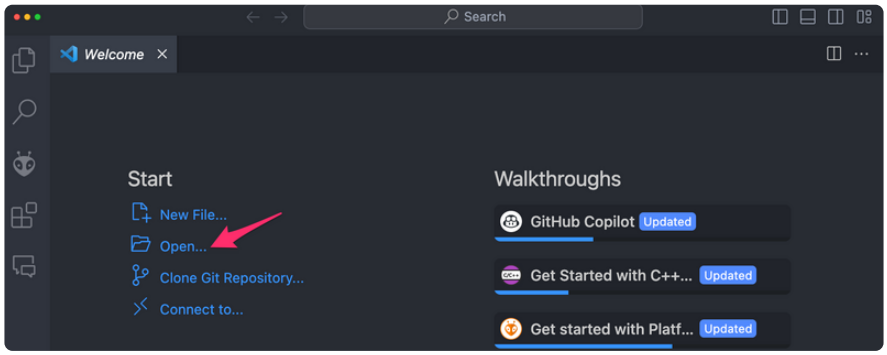
Download Project Code and PlatformIO Environment

https://adafru.it/19mA

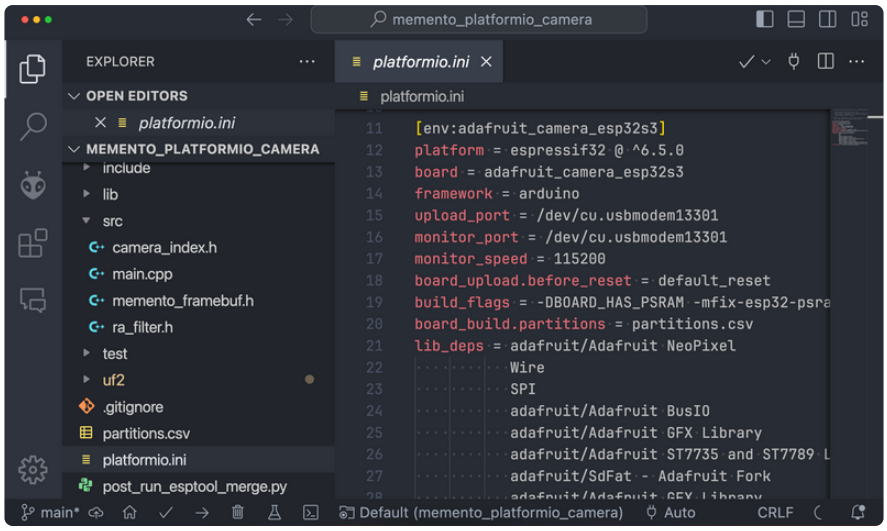
Open Visual Studio Code (VSCode). To ensure you have installed the PlatformIO extension properly, **look for the alien symbol in your VSCode sidebar.**



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, focus on the following files and directories:

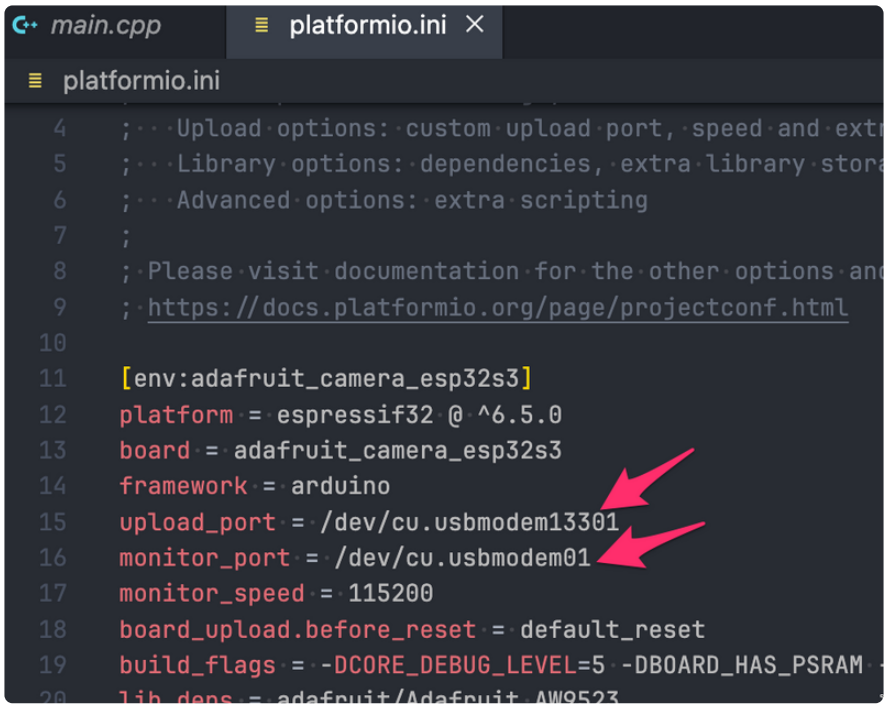
- **platformio.ini** - This is the project configuration file used to build the demo code. More documentation about this file is located here (https://adafru.it/19cw).
- **lib** directory - This directory is intended for project-specific (private) libraries. PlatformIO will compile them to static libraries and link them into executable files.
    - For this project, the specific library within this directory is the Adafruit_PyCamera (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

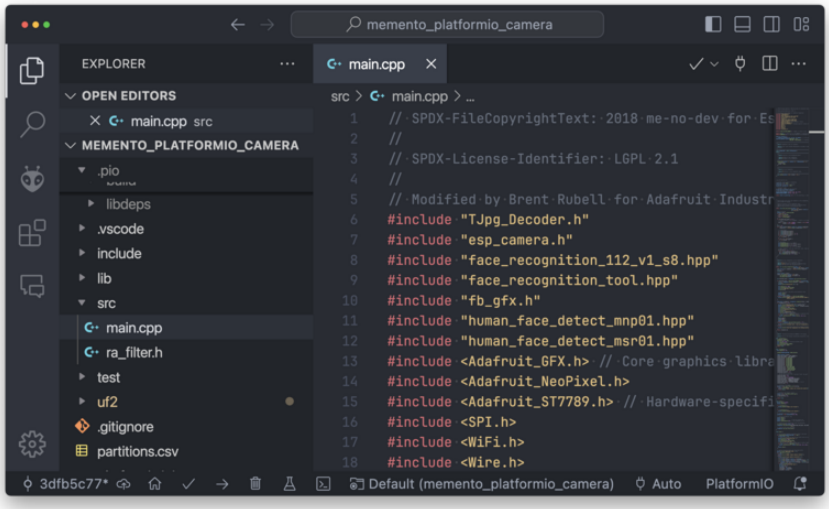
Before the code is built, you'll need to make two changes to the **platformio.ini** file:

- Change `upload_port` to reflect the MEMENTO upload port.
    - Don't know the desired port? There are steps to find them for Windows (https://adafru.it/19cy), MacOS (https://adafru.it/19cz), and Linux (https://adafru.it/19cA).

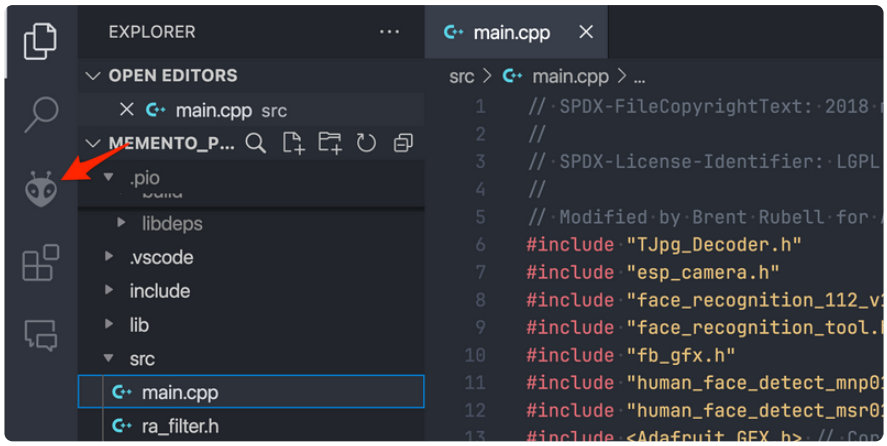
- The `monitor_port` is different from the `upload_port`, and will only appear on your computer when you've uploaded the test code. For now, leave this alone.
    - After uploading the test code, change `monitor_port` to reflect the MEMENTO's monitor/serial port.



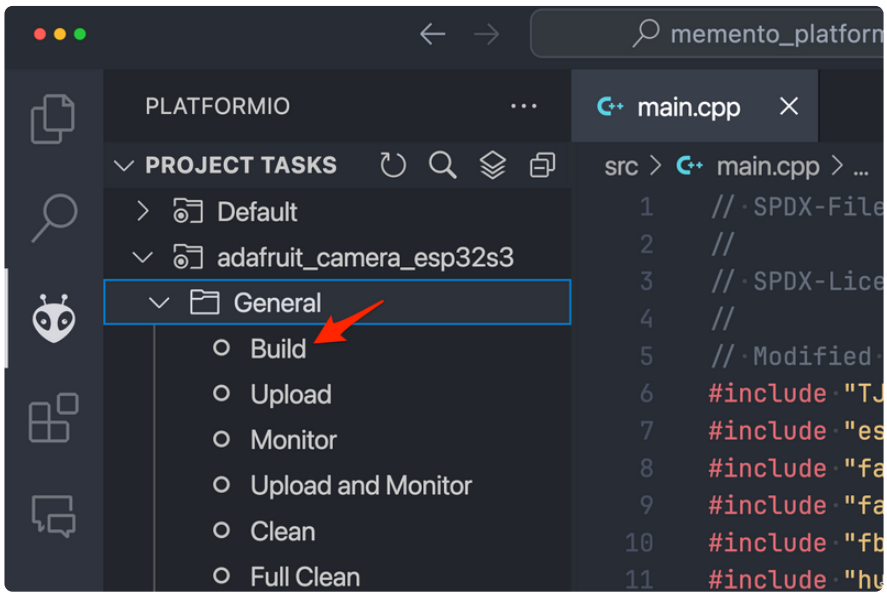
Navigate to **src/main.cpp** to open the example code.



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**.

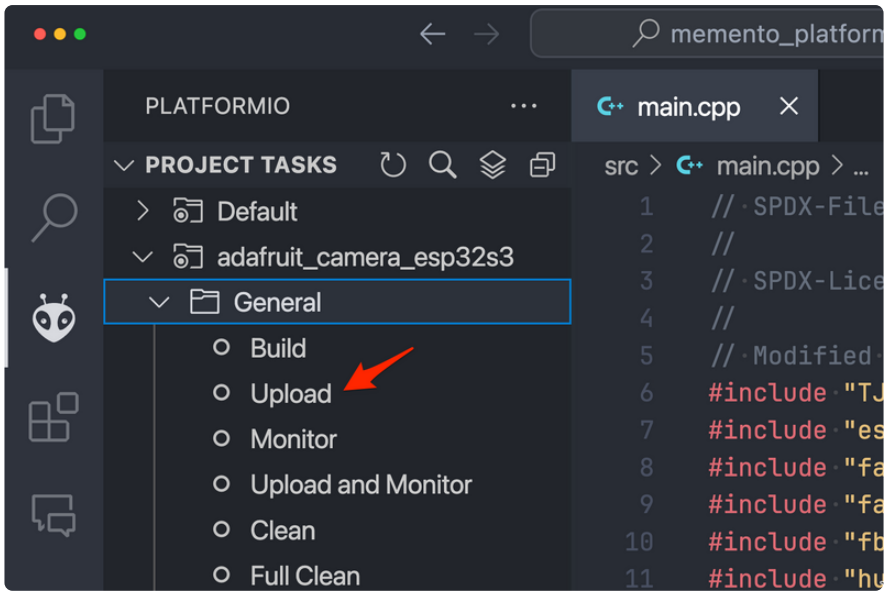


Once the build task is completed, the terminal will show `SUCCESS` along with the time it took to compile the project.

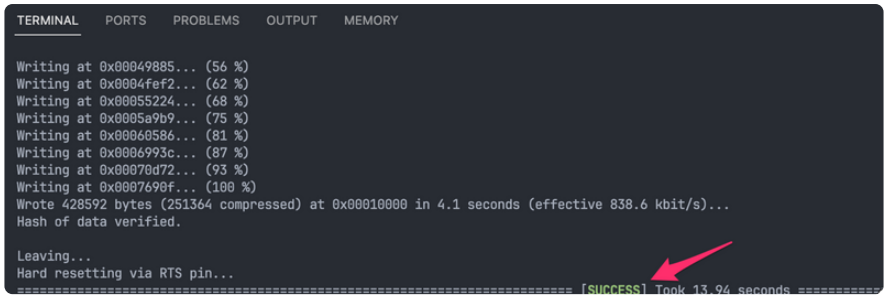
```
Checking size .pio/build/adafruit_camera_esp32s3/firmware.e
lf
Advanced Memory Usage is available via "PlatformIO Home > P
roject Inspect"
RAM:   [==        ]  15.5% (used 50720 bytes from 327680 by
tes)
Flash: [======    ]  62.8% (used 2511073 bytes from 3997696
 bytes)
=============== [SUCCESS] 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).

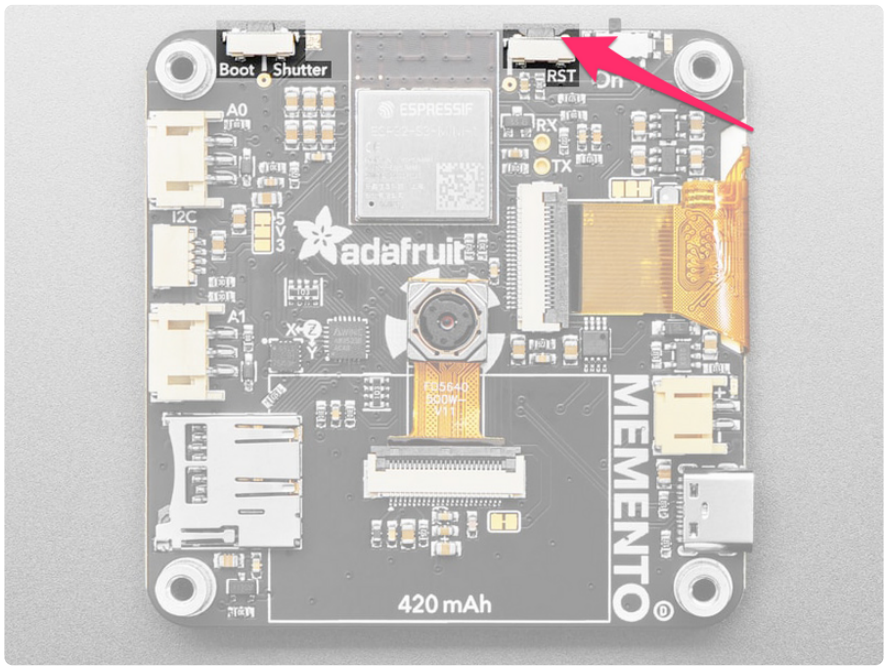
From the PlatformIO Project Tasks menu, click **Upload**.



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!