Adafruit MONSTER M4SK

Created by Phillip Burgess

https://learn.adafruit.com/adafruit-monster-m4sk-eyes

Last updated on 2024-04-09 10:47:38 AM EDT

Table of Contents

Overview 5

Update the UF2 Bootloader 8

- Updating Your Bootloader

Quickstart 11

Flash Filesystem 12

- Creating the SPI Flash Filesystem

M4 EYES Firmware 14

Ready-Made Graphics 15

Customization Basics 22

Preparing Graphics 24

- Image Storage
- Eyelids

Configurable Settings 28

Sizes and Shapes 29

Colors and Textures 31

Distinct Left/Right Eyes 32

Eyelids 33

Light Sensor 34

Voice Changer 34

- Usage
- Example Config.eye File
- Tips for using the Monoprice 5-Watt Guitar Amplifier

Building Eyes from Source Code 37

Arduino IDE Setup 38

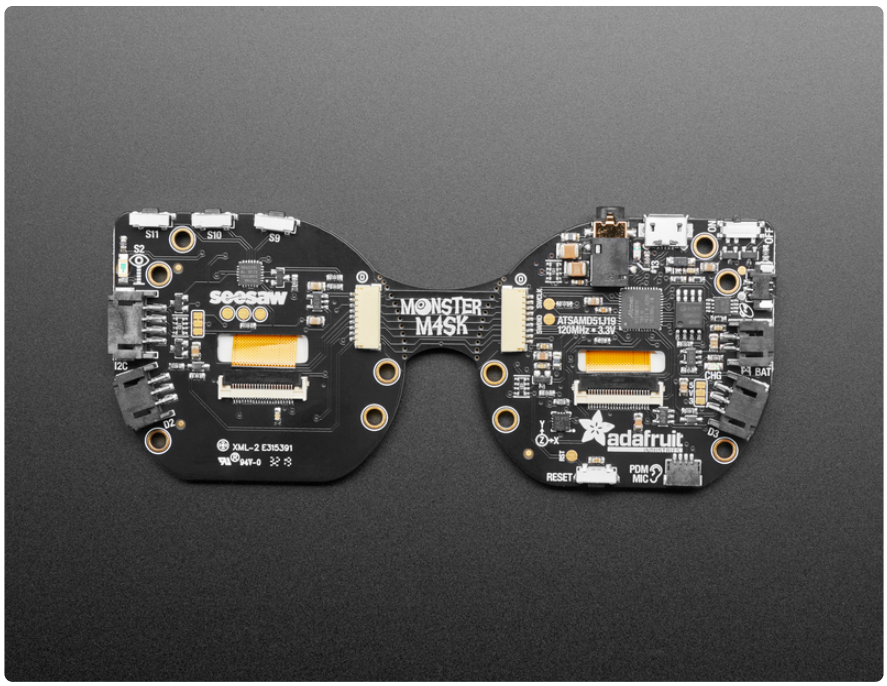
Using with Arduino IDE 40

- Install SAMD Support
- Install Adafruit SAMD
- Windows 7 and 8.1
- Blink
- Successful Upload
- Compilation Issues
- Manually bootloading
- Ubuntu & Linux Issue Fix

Source, Libraries and Settings 46
• Source Code
• Libraries
• Project Settings
Troubleshooting 48
• Board does not show up as drive when connected to computer
• Board Unresponsive; Flashing Red LED
• Ol' Blue Eyes
• Crossed Eyes
• Technicolor Snow
• Erase the Filesystem
• Diagnostics
• Screen(s) Coming Unstuck
Lens Holders 51
• Lens Holders
• 3D Printed Options
Voice Changer 54
Pinouts 54
• Displays and Display Connectors
• Microcontrollers and Flash
• Power Switch and Ports
• Audio
• Capacitive Touch Pad and Buttons
• Sensors
• I2C Connector
• Digital/Analog Connectors
• Reset Button
• LED
• SECTION TITLE HERE
Downloads 62
• Files:
• Schematic
• Fab Print

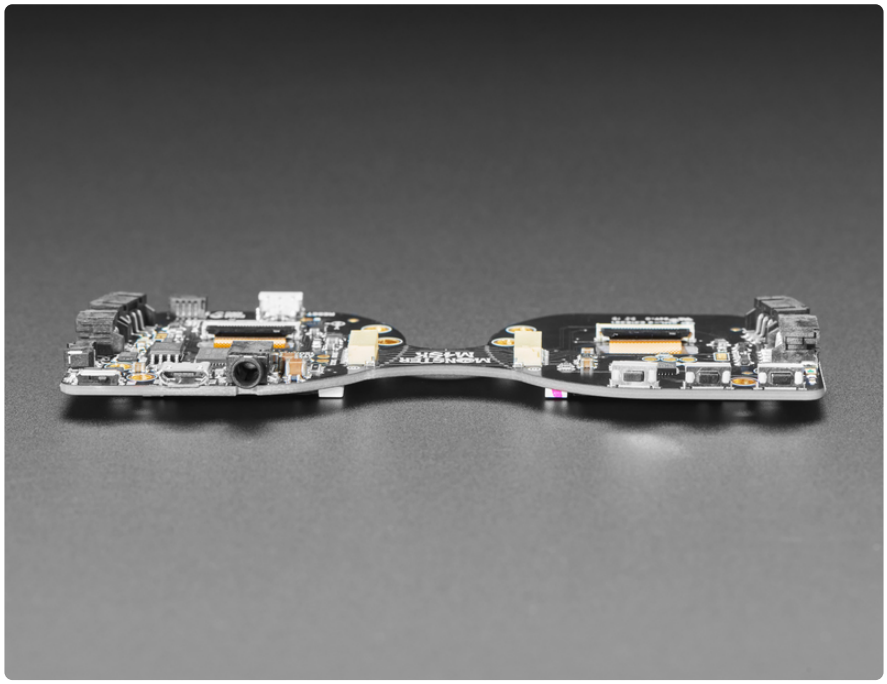
Overview

Peep dis! Have you always wanted to have another pair of eyes on the back of your head? Or outfit your costume with big beautiful orbs? The MONSTER M4SK is like the Hallowing (http://adafru.it/3900) but twice as good, with two gorgeous 240x240 pixel IPS TFT displays, driven by a 120MHZ Cortex M4 processor that can pump out those pixels super fast. You'll get the same quality display as our Raspberry Pi Eyes kit but without needing to tote around a full Linux computer (http://adafru.it/3356)

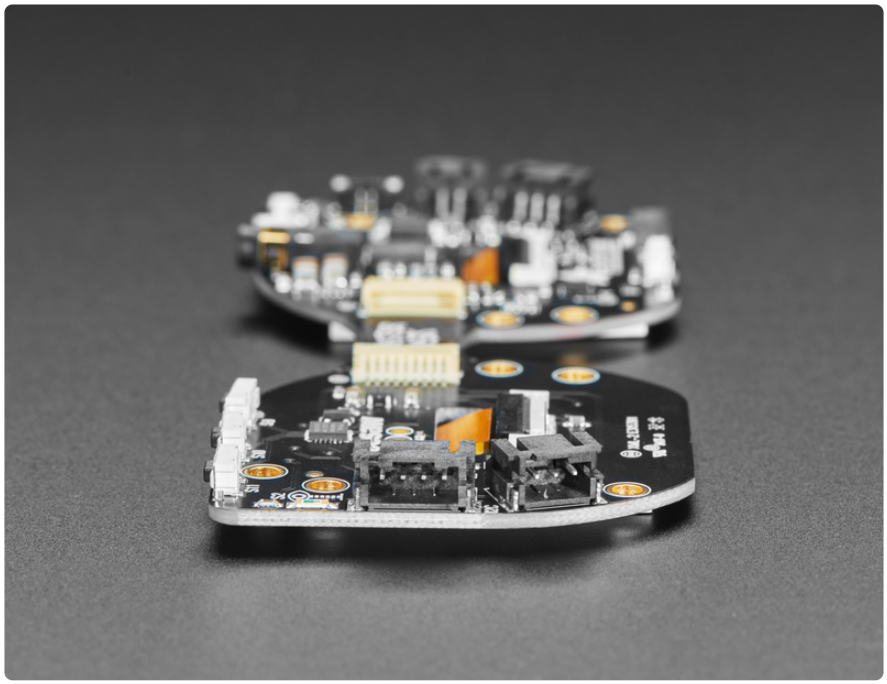


This unique design has the eyes at the same pupil-distance as a human (~63mm) but is designed so that the nose section can be broken apart with pliers/cutters and then

wired together with a 9-pin JST SH cable up to 100mm long (http://adafru.it/4350) so the eyes can be re-positioned or freely attached.



We wanted to make audio-effects easier so in addition to a class D audio amp, there's also a stereo headphone jack that is connected to the two DACs on the chip. Use it when you want an externally sound amplifier box for big effects. For small portable effects, the built-in amp can drive 8 ohm speakers up to 1 Watt.



On each side are JST-PH plugs for connecting external devices. The 3-pin JSTs connect to analog/timer pins on the SAMD51, so you can use them for sensors or GPIO devices. The 4-pin JST connector connects to the I2C port and you can fit Grove

connectors in it for additional hardware support. For the PDM mic port, you can use this cable (http://adafru.it/4209) to wire to a PDM mic (http://adafru.it/3492).

For power, you can either use a USB micro-B cable, or connect a li-poly battery to the 2-pin JST-PH “BAT” connector — either directly, or with an extension cable (http://adafru.it/1131) and the battery tucked elsewhere.

There's also plenty of sensors built in - light sensor, 3 tactile buttons, and a capacitive touch pad on the nose.

Speaking of that nose, the silkscreen is by the skillful Miss Monster, check out those fangs! (https://adafru.it/FB1)

This is by far the cutest, creepiest and most incredible development board we've made so far! Gaze upon these features:

- ATSAMD51G19 Cortex M4 microcontroller running at 120MHz with 512KB Flash, 192KB RAM (the board markings incorrectly say ATSAMD51J19)
- 8 MB QSPI flash for storing graphics and sound effects
- Two 240x240 IPS TFT displays each on their own SPI bus
- Beautiful silkscreen with a boop-able nose that is a capacitive touch pad
- Lipoly battery charge circuit for portable use
- Stereo headphone jack out, for sound effects via an amplifier
- Mono speaker driver for smaller 8 ohm 1W speakers
- One 4 pin STEMMA JST connector for I2C connection (also Grove compatible)
- Two 3 pin STEMMA JST connectors with digital/analog/PWM for servos, sensors, etc

- One 4 pin JST SH port for connecting an optional PDM microphone
- Backlight controls
- Three tactile buttons
- Light sensor
- On/Off Switch and reset button

And as you can expect, we've got some great new eyeball code, which does 2 eyes with user-configurable graphics. Right now our code support is only for Arduino - CircuitPython isn't quite fast enough to do the 3D animation techniques we use to draw the eyeballs. The eyes look even better if you pair them with these 40mm glass or plastic lenses. (You'll need two of course) (https://adafru.it/FB2)

Update the UF2 Bootloader

Update the Bootloader on your SAMD51 M4 board to prevent a somewhat rare problem of parts of internal flash being overwritten on power-up.

Your SAMD51 M4 board bootloader may need to be updated to fix an intermittent bug that can erase parts of internal flash.

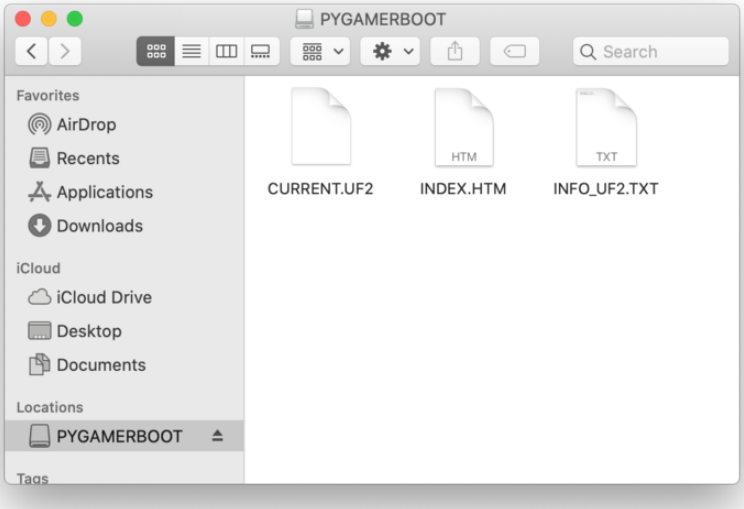
Updating Your Bootloader

To see if you need to update your bootloader, get the UF2 boot drive to appear as a mounted drive on your computer, in a file browser window. If you're running

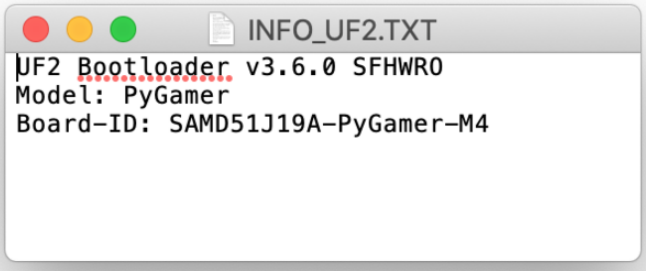
MakeCode, click the reset button once. If you're running CircuitPython or an Arduino program, double-click the reset button.

When you see the **...BOOT** drive (**FEATHERBOOT**, **METROM4BOOT**, **ITSYM4BOOT**, **PORTALBOOT**, etc.) , click the drive in the file browser window and then double-click the **INFO_UF2.TXT** file to see what's inside.

The example screenshots below are for a PyGamer. What you see for your board will be largely the same except for the board name and the **BOOT** drive name.

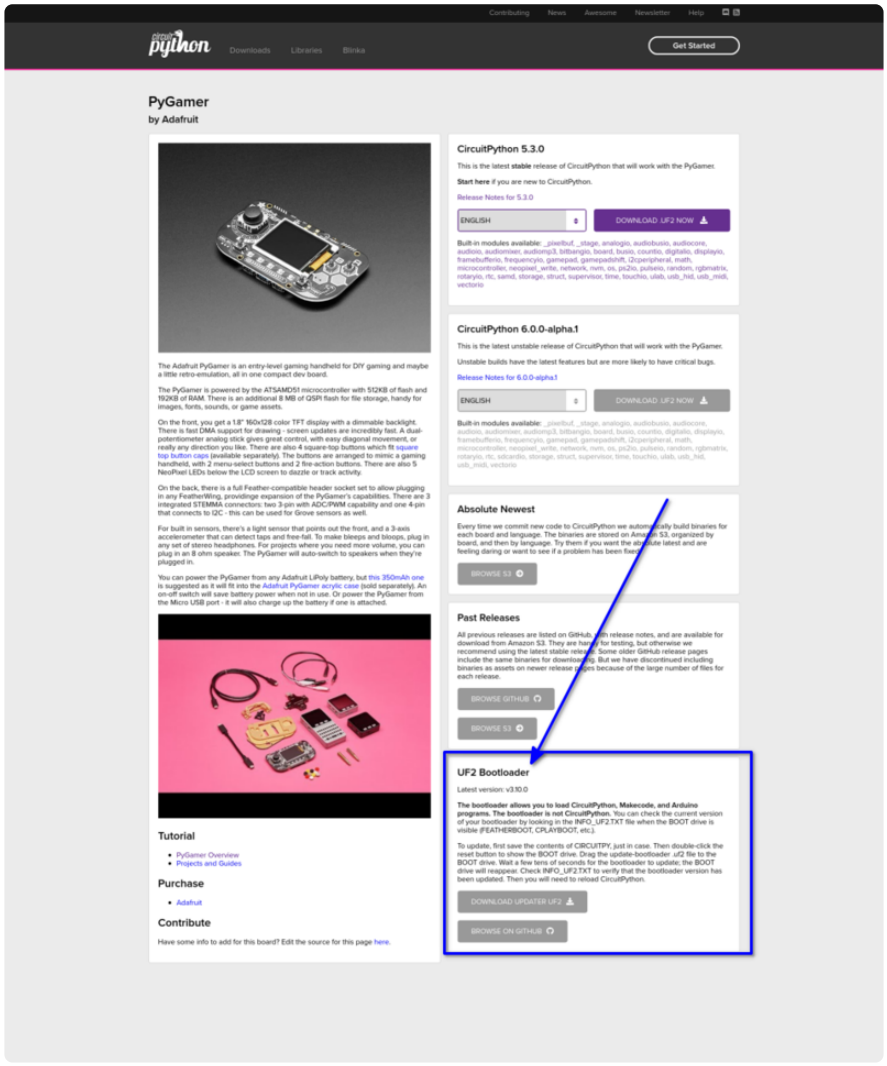


The bootloader version is listed in **INFO_UF2.TXT**. In this example, the version is **v3.6.0**.



If the bootloader version you see is older than v3.9.0, you need to update. For instance, the bootloader above needs to be upgraded.

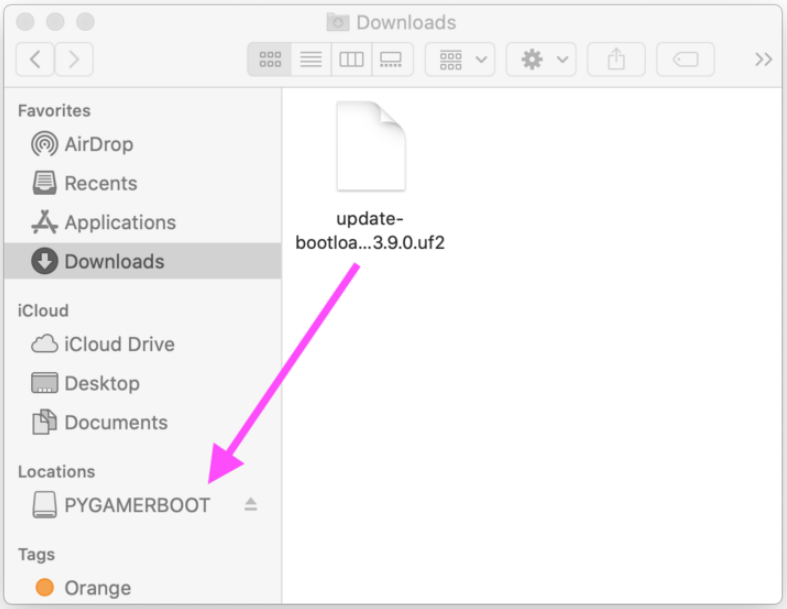
Download the latest version of the bootloader updater from the circuitpython.org Downloads page for your board.



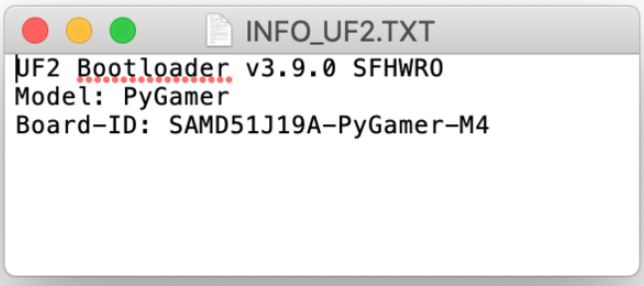
Latest board downloads

https://adafru.it/Em8

- The bootloader updater will be named **update-bootloader-name_of_your_board-v3.9.0.uf2** or some later version. Drag that file from your **Downloads** folder onto the **BOOT** drive:



After you drag the updater onto the boot drive, the red LED on the board will flicker and then blink slowly about five times. A few seconds later, the **BOOT** will appear in the Finder. After that, you can click on the **BOOT** drive and double-click **INFO_UF2.TXT** again to confirm you've updated the bootloader.



Quickstart

The MONSTER M4SK board should arrive **ready-to-scare**. Connect a battery or USB cable, move the power switch to the “on” position, and after a few seconds you should get **blinking eyes**.

The next few pages cover...

- How to reformat the flash storage. **Most readers can skip this**, unless you got one of the very earliest MONSTER M4SK boards, or if it just needs a complete wipe.
- How to **install or update the eye firmware** for the **latest features** and **bug fixes**.
- Loading **different eye designs**.

Flash Filesystem

If plugging your MONSTER M4SK board into a computer over USB and it does not appear as a small flash drive named **CIRCUITPY**, it's usually one of three reasons:

- Verify that you're using a **USB "charge and sync" cable**, not a charge only cable.
- The **first batch** of MONSTER M4SK boards **did not have the flash filesystem initialized**...we'll cover that below.
- Something happened...maybe static, maybe a bug in the software...and **corrupted** the flash drive contents.

If you **do** see the **CIRCUITPY** flash drive, everything's good and you can skip ahead to the next page.

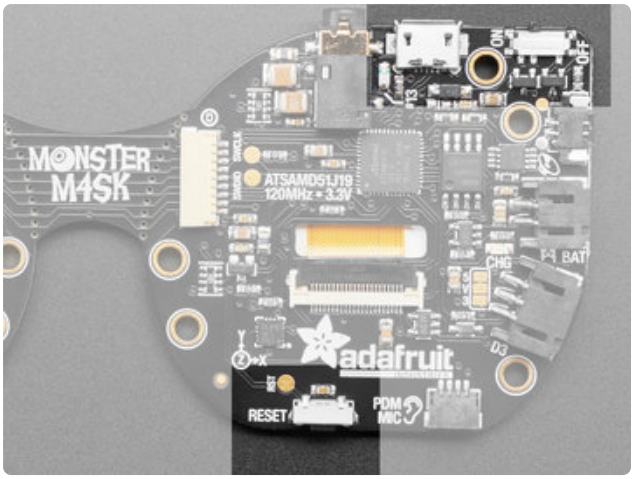
Creating the SPI Flash Filesystem

If the CIRCUITPY drive does not appear, this just involves **temporarily** installing a version of **CircuitPython**. The eye code isn't written in CircuitPython, we'll just use it here to get the flash initialized!

Download that .UF2 file here:

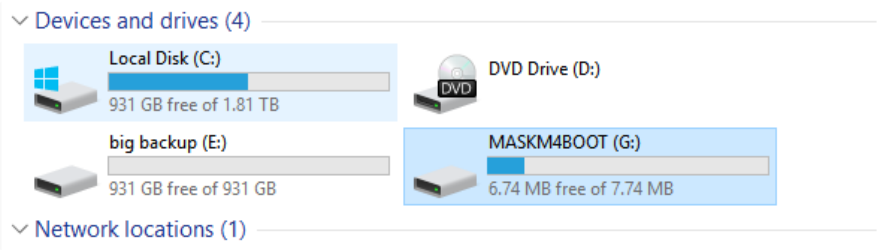
Download CircuitPython for MONSTER M4SK

https://adafru.it/FPn



Connect MONSTER M4SK to your computer with a USB cable, set the power switch to the ON position and double-tap the reset button to enter bootloader mode.

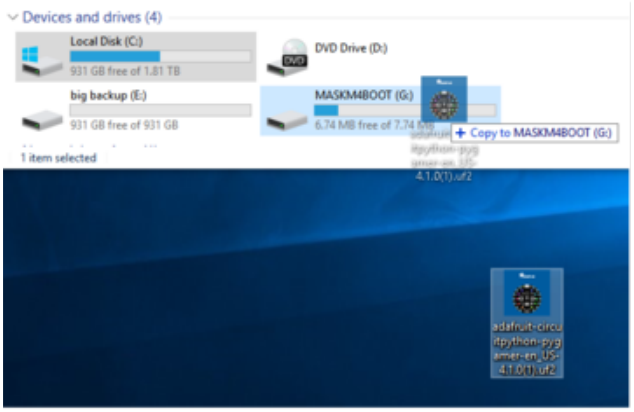
After a moment, a drive called MASKM4BOOT should appear.



I'm not seeing the MASKM4BOOT drive!

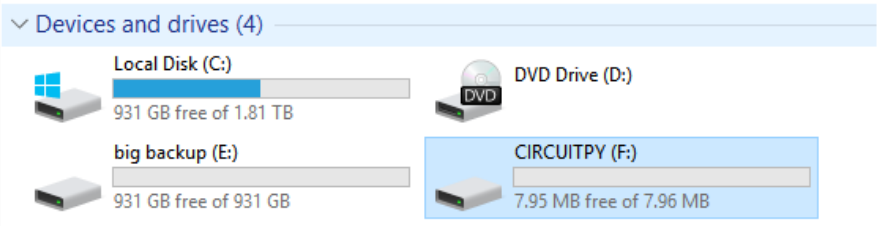
Things to check:

- Confirm the **power switch** is in the **ON** position.
- **Double-tap** the **reset** button again. The red LED on the back should do a slow "breathing" cycle. If not, try again.
- Make sure you're using a good USB "**charge and sync**" cable, NOT a "charge only" cable.



Drag the CircuitPython .UF2 file to this drive and wait a few seconds.

After CircuitPython installs, and with the MONSTER M4SK board connected over USB, it should appear on your computer as an 8 MB flash drive called CIRCUITPY.



In rare situations the flash memory might need a complete erasing, before even setting up the flash filesystem. The Troubleshooting page of this guide includes a utility for this. Once erased, you can then start again with the filesystem setup as described above.

Initializing the filesystem is a one-time operation. You should not need to repeat this process, unless the flash drive contents somehow get corrupted.

M4 EYES Firmware

To **update firmware** on your **MONSTER M4SK** board with the latest **eye animation** software, start by downloading this .UF2 file:

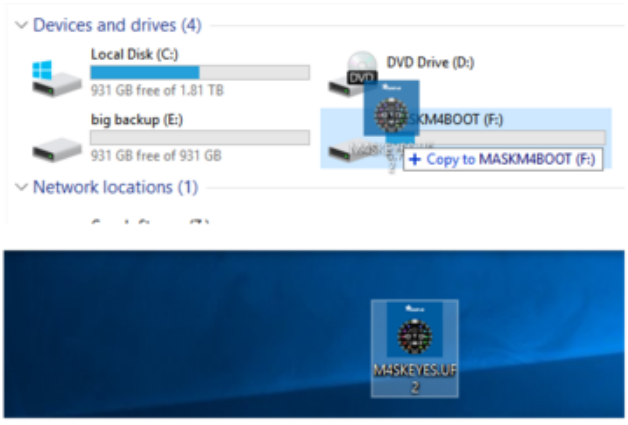
M4SKEYES.UF2

https://adafru.it/TPe

For a **HALLOWING M4** board, use this .UF2 instead:

HALLOM4EYE.UF2

https://adafru.it/19c9



The firmware update sequence is:

1. Connect a USB cable and put the power switch in the “on” position
2. Double-tap the reset button
3. **Wait for the MASKM4BOOT drive to appear!**
4. Drag the M4SKEYES.UF2 file to the MASKM4BOOT drive and wait for it to copy over
5. The board will automatically reboot

After installing the firmware (and a brief pause while the software initializes) you should get some **animated eyeballs**.

If you **don't** see any eyes, make sure you dragged the **M4SKEYES.UF2** file to the **M4SKBOOT** bootloader drive not the **CIRCUITPY** drive

If you get simple **flat-colored** eyes like shown here...that just means **no graphics files** are installed yet. We cover that on the next page. But at least we know the code's installed!

If you get **textured** eyes that **blink**...code and graphics are all in good shape! The next page shows how to install different looks...and later we get into total customization.

Ready-Made Graphics

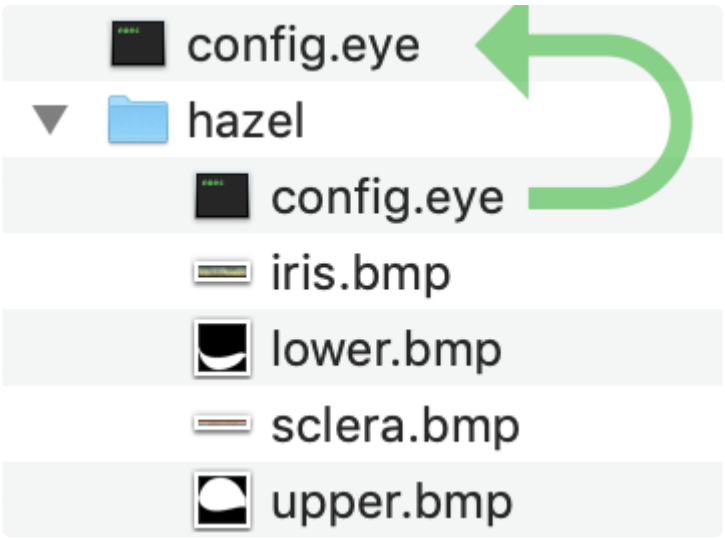
To install different "looks," download and unzip this collection of eyeball graphics:

Download Eye Graphics

https://adafru.it/FAH

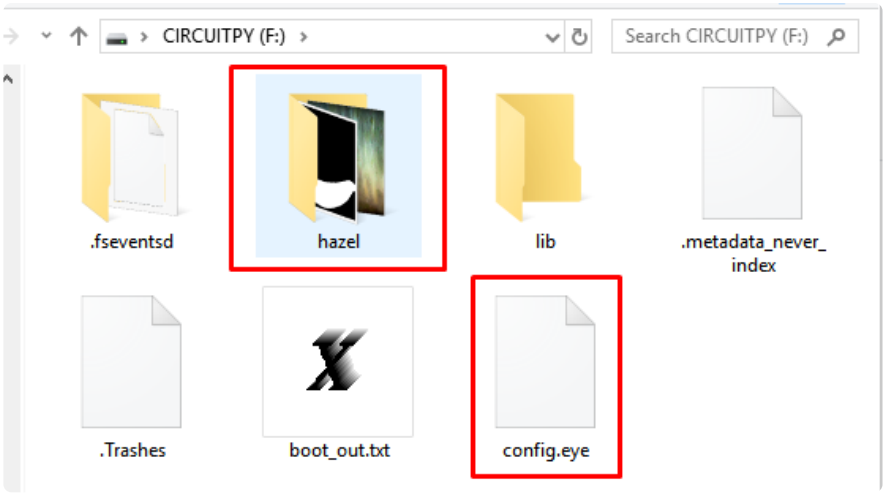
Inside the "eyes" folder are several sub-folders, one for each of our ready-made eye designs. "hazel" is our standard human eye from prior projects...then we're adding more as Halloween approaches.

Each folder contains a set of .bmp images for that eye, plus a file called **config.eye**. Copy one of these folders to the **CIRCUITPY** drive — not the individual files, but the whole folder. Then copy or move the file called **config.eye** out of the folder and into the drive's root directory.



Copy these files to the **CIRCUITPY** drive.

Remember that **config.eye** goes in the root directory, along with the **hazel** (or other eye name) folder.



Press the board's **reset** button...

There will be a **delay** of **several seconds** as the eye code initializes. Then...animation!

If you get a flat-colored white eyeball with blue irises and no eyelids, something's wrong! Verify that you've copied over a whole folder of graphics (not the individual files) and moved or copied that folder's **config.eye** to the root directory.

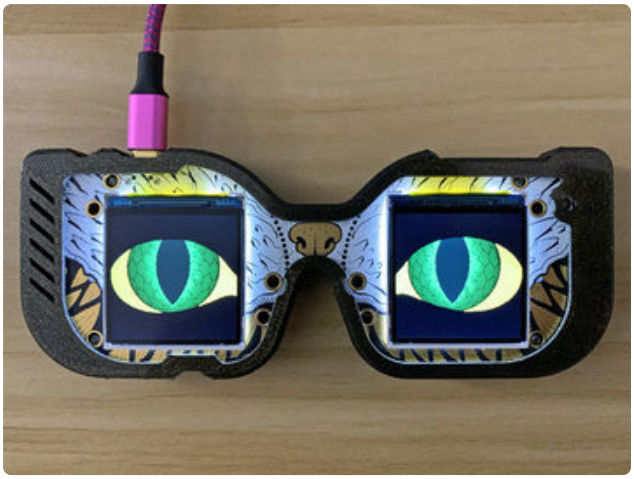
Aside from the stock **hazel** eyes, some of the alternate designs include:

big_blue is a pair of large and friendly blue-gray eyes. The sclera doesn't have all the veins of the hazel eyes, making this less creepy.

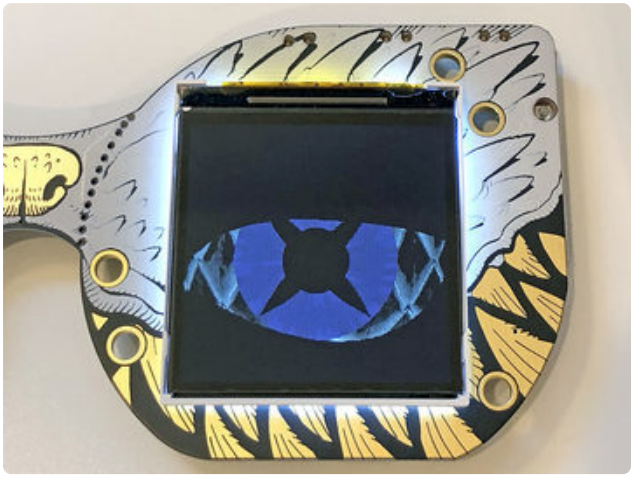
fish_eyes is the same unblinking eyes used in our Fish Head MONSTER M4SK project (https://adafru.it/FQj).

hypno_eyes have that cartoon mesmerizing look. "You are in my power!"

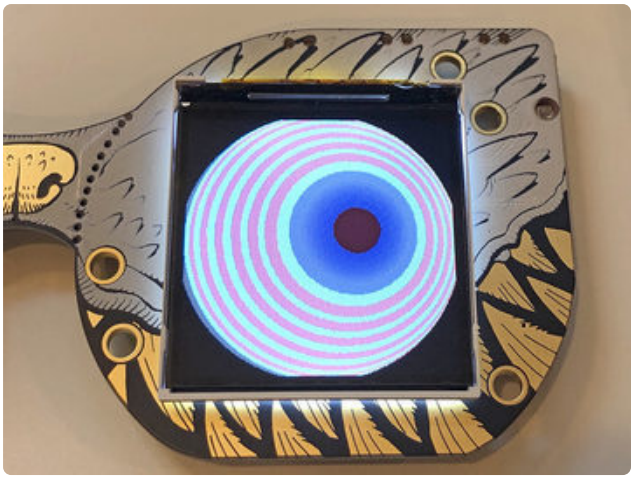
reflection looks like a pair of shiny spheres. No eyelids or pupils, just spheres looking extremely reflective. *ting!*



snake_green is perfect for dragons and other reptilian characters. Rar! This shows off the **slit pupil** option...also useful for cats and the like.

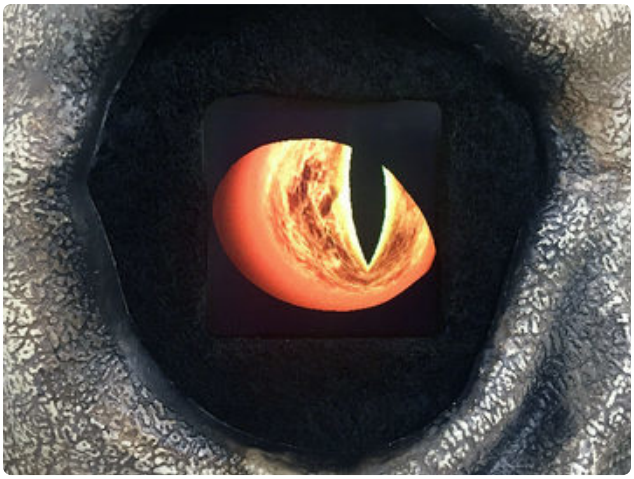


spikes is adapted from the guide CustomEyesation: DIY Monster M4SK Graphics (https://adafru.it/FFU) and demonstrates a bit how the distortion of texture mapping works.



toonstripe is a different “mesmerizing eyes” look...candy colors, no eyelids.

doom-red and **doom-spiral** were designed for the MONSTER M4SK Toon Hat (https://adafru.it/FQk) guide but might have other uses or tricks to learn from.



The **demon** eyes have swirling fire ... a bit of an “Eye of Sauron” resemblance.

anime does its best approximation of The Quaking Moist Anime Eyes Effect™.

fizzgig resembles the fuzzy Dark Crystal character. These react to ambient light and were designed with the lenses in mind.

SECRET FEATURE: You can select among four different eye configurations on startup. Hold down one of the three buttons on top when powering on (or pressing reset) to load files **config1.eye** (inner button), **config2.eye** (middle button) or **config3.eye** (outer button) instead of the default **config.eye**. All go in the root directory for now (but graphics & such can go in folders).

If one of the ready-made designs does what you need...fantastic, you're all done! If you want to make **changes**, or create your own **custom eyes**, read on...

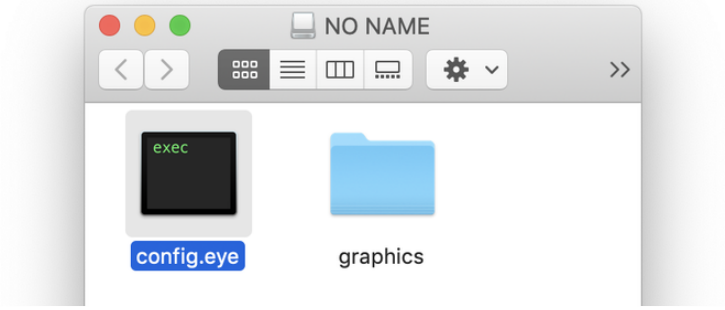
Customization Basics

If our ready-made eye designs don't meet your needs, you can personalize quite a bit by editing a text file and providing some graphics.

On startup, our eyeball code loads a file called **config.eye** from the device's root directory. This is a plain-text file that any simple editor can handle...Notepad, TextEdit, vim and so forth.

You wont be able to double-click this file, so open it from WITHIN your text editor

If customizing your eyes, maintain a **backup** of the board's contents on your computer. There is a small but nonzero chance of a glitch that could require a full reinstall. See the Troubleshooting (https://adafru.it/TPc) page for help.



The syntax of this file is a format called **JSON**. There's good and bad news...

The good: JSON has a standardized syntax and we can leverage existing code to read it...we're not starting from scratch with a new file format.

The bad: JSON files are really meant to be read and written by machines, as a way to preserve program state...not edited by humans. It's phenomenally picky about getting syntax just right and does not fall back gracefully.

The saving grace: if you're just changing colors and textures, you might not need to edit this file at all! Quite often you can just substitute different graphics files with the same names.

Troubleshooting JSON

If even a single character is out of place in **config.eye**, the whole thing will fail to load and there's no helpful indication of where the problem lies.

If this happens, the eyes will run in a **default state**: blue eyes with no eyelids, and everything is "flat" colors, no textures. This isn't helpful in isolating the problem but at least tells you there is a problem.

One way to troubleshoot this is to start with a known valid eye configuration, then change one setting at a time and restart the code. If the eyes revert to the default state, the last change contains the syntax error. Quite often it's just a missing double quote or extra comma.

Another option is to use a **JSON validator** such as this one at hjson.org (https://adafru.it/FzV). Copy-and-paste your eyeball configuration into the left pane...if there's a problem, this will be reported on the right.

Here's an simple **config.eye** file to start with:

```
{
  "pupilColor"    : [ 0, 0, 0 ],
  "backColor"     : [ 140, 40, 20 ],
  "irisTexture"   : "graphics/iris.bmp",
  "scleraTexture" : "graphics/sclera.bmp",
  "upperEyelid"   : "graphics/upper.bmp",
  "lowerEyelid"   : "graphics/lower.bmp"
}
```

Some things to notice:

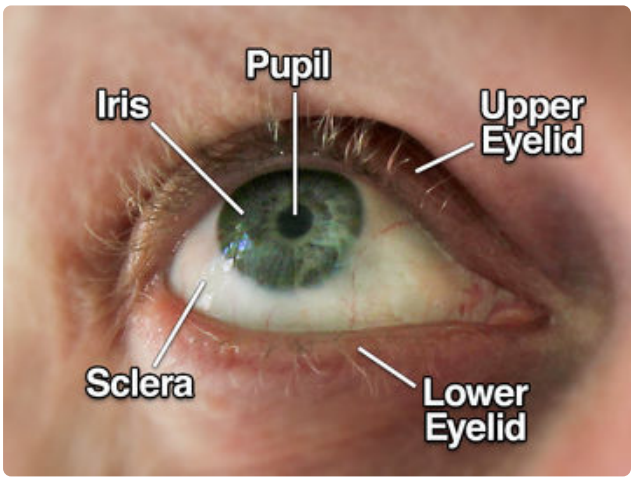
- The whole thing is contained within curly braces — { and }

- Each item has a **name** (always in quotes), a colon **separator** (**:**) and a **value**. Values might be numbers, strings (in quotes) or arrays (in square brackets) depending on what's being configured.
- This is a list and each line ends with a **comma**...except for the last item in the list. **Watch out for missing or surplus commas!**
- JSON is **case-sensitive**. "Foo", "foo" and "FOO" are all different things.
- Using extra spaces to line up columns really isn't necessary, just something I do to assist with legibility. Some folks just find it annoying.
- Comments (starting with //) are not standard JSON syntax, but the library we use allows them and I find them very helpful.

Each option will be explained in detail on the "Configurable Settings" page. But first let's talk about **graphics**...

Preparing Graphics

So we're singing from the same page, let's lay out some eye terminology...

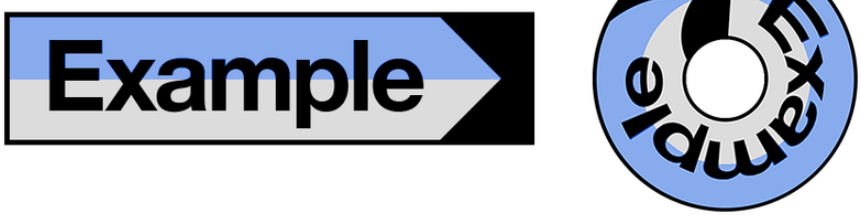


The technical term for the "white" of the eye is the sclera.

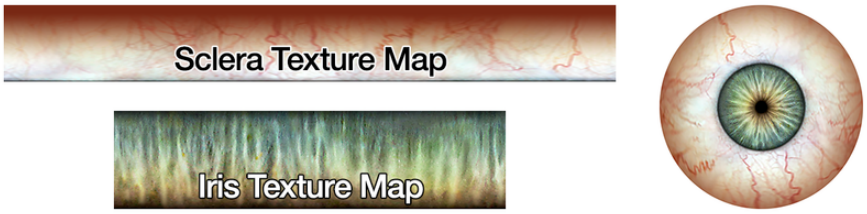
The iris is the muscle that contracts to adjust the size of the pupil in response to light.

The upper and lower eyelids are involved in blinking.

This program's eye graphics are stored flat and unrolled, like a map projection. The horizontal (X) axis works like the longitude, or angle around the eye, while the vertical (Y) axis is the latitude. The images are wrapped around the pupil in a **clockwise** direction.



There are **two** images (or texture maps) associated with the eyes...one for the iris, another for the sclera.



The **iris** is what we think of as the "color" of the eye and is most often what you'll want to edit. Sometimes you just need to edit the hue & saturation in a program like Photoshop, or you can make something totally custom if you're after a particular look.

The **sclera** is the "white" of the eye...which really isn't that white at all. There's veins and blotches and gross stuff!

Image Storage

The texture maps are stored as **24-bit BMP** images...nobody's favorite, but easy for microcontrollers to handle. (This is also sometimes called "Windows Bitmap" format, though plenty of non-Windows software can read and write these images...and not to be confused with "X BitMap" or "Portable Bitmap Format," different animals.)

Textures are loaded from drive into RAM, downsampled to 16-bit color (what the displays natively use) and then written to the chip's internal flash memory (different from the flash filesystem). Although the code places no strict limit on image dimensions, **RAM and flash are both finite resources**, and this limits to how large these textures can be...both individually and in total.

No single image can exceed available RAM, or about **160 kilobytes**. The total of all images must fit within flash, or about **360 kilobytes**. These figures might change a bit in the future, so try to leave yourself some overhead.

Use the following formula to determine the space needed for an image:

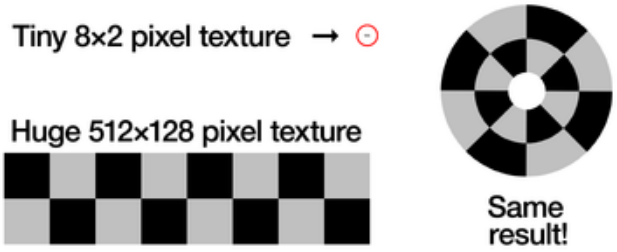
width in pixels × height in pixels × 2 bytes

For example, a 500 × 150 pixel texture would consume 500 × 150 × 2 = 150,000 bytes. This fits in RAM just fine, and takes up a bit less than half of the flash space.

You don't have to texture-map both the iris and sclera if you don't want to...on the Configurable Settings page we explain how to use **solid colors** for either or both. Also

shown there...it's possible to assign independent textures to the left and right eyes. When both eyes are sharing the same texture, the code will place only one instance in flash, saving space.

To make the re-use of textures less obvious, the "seam" where a texture map wraps around is normally at the 12 o'clock position for the right eye and 6 o'clock for the left eye. The Configurable Settings page shows how to change this.



Eye textures can be any size (RAM permitting), on either axis...even down to a single pixel. When wrapping small images around the whole eye, **nearest-neighbor** sampling (no interpolation) is used. This can be exploited to create stylized blocky or grid designs...no need to waste space on a big image when just a few pixels will do.

Too-large images may exceed available space, while too-small images may exhibit visible jaggies (unless you're aiming for that effect as described above). The **ideal size** can be calculated based on the circumference of the iris or the whole eye...

The iris and overall eye size are configurable (shown on following page)...but for example, let's assume you've got an iris with a 60 pixel radius (120 pixel diameter) and the eyeball has a 125 pixel radius (250 pixel diameter)...both of these are the defaults.

Multiply the iris and/or eye **diameters** by **Pi (3.14)** to get the **ideal width in pixels** for the iris and sclera images.

For example: iris with 120 pixel diameter. 120 px × 3.14 = 377 pixels wide. You can use that, or round up or down a smidge to a round number if you like (e.g. 360 or 380 pixels).

Sclera image width for 250 pixel diameter eye: 250 × 3.14 = 785 pixels wide...but again, OK to round up or down a little...use 800 pixels wide if you like, unless really pressed for space.

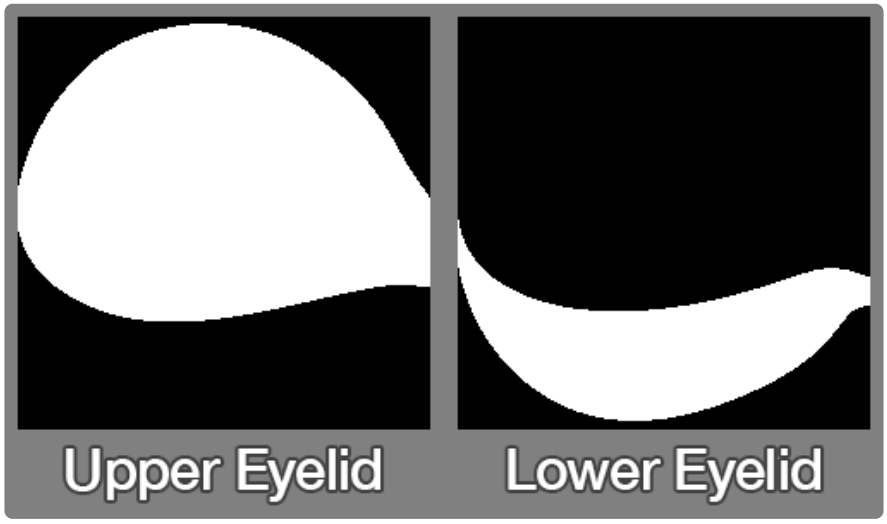
The ideal image heights are a bit different. First, although the code can load any size image, it won't actually benefit above 128 pixels on the vertical axis, it's just wasted space. For the iris, use its radius (60 pixels in the case described above) or even a little less, since the pupil is always open a bit. For the sclera...try 200 minus the iris

radius, keeping in mind the 128 pixel recommended maximum (e.g. with a 60 pixel iris, 140 is our target, then cap it at 128). But...with an 800 pixel wide sclera as described above...800 × 128 × 2 = 204 kilobytes...quite a bit over the 160K RAM limit! Whittle down one or both axes, whatever you think can best handle less resolution, until you find a size that fits. Once in motion, and at a reasonable viewing distance, minor "jaggies" aren't that noticeable.

Photoshop users: the eye code maps textures differently than Photoshop's "Polar Coordinates" filter. Ours wraps clockwise, with the top of the image becoming the outer circumference, while Photoshop wraps counterclockwise, with the bottom becoming the outer circumference. If using Photoshop to create or preview textures, simply rotate the rectangular image 180° before applying a rectangular-to-polar filter, or after a polar-to-rectangular filter.

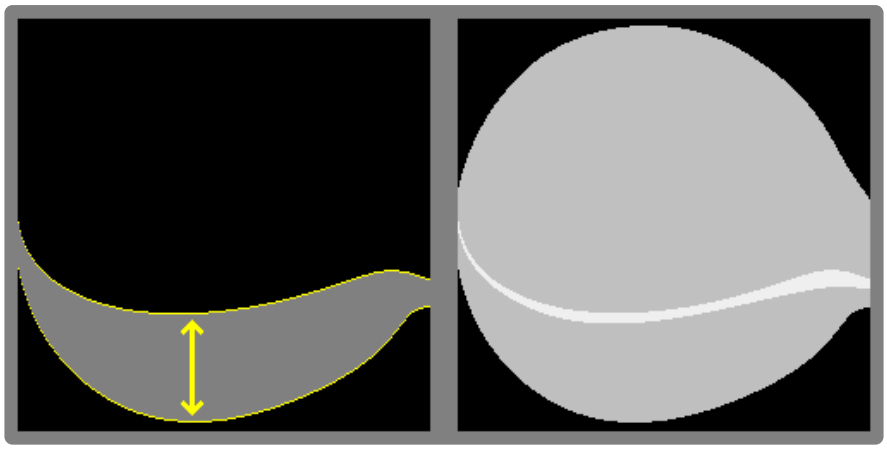
Eyelids

The eyelids have stricter requirements. There are always **two files** (one each for the **upper** and **lower** eyelids) both **240 × 240 pixels** exactly, both **1-bit BMP images** (NOT 24-bit like the texture maps!).



The white area — which should span the **entire 240 pixel width** — represents the extent of vertical motion, the "open-most" and "closed-most" range. The eyes won't hold the open-most position all the time when idle...the upper eyelid "tracks" the moving pupil, because that's how eyes work.

A good set of eyelid images will **overlap by a few pixels**, so the eye makes a good solid blink.



This particular set of eyelids is slightly **asymmetrical** to approximate the eye's caruncle* — that triangular bit by the tear duct. The shape is mirrored between the two eyes. But it is super 100% okay to make **symmetrical** eyelids if you prefer...a simple "football shape"...that usually looks better on a single-eye board like the HalloWing M4. For some of the eye designs we'll offer both.

* Which itself is a vestigial remnant of the nictitating membrane, the "third eyelid" that reptiles have. How cool is that!?

The eyelid images as shown above are for the right eye. That is...**the monster's right eye**, meaning the eye on the **left** when looking at the M4SK.

Configurable Settings

Let's look at the configuration for our stock "hazel" human eye. It doesn't reference every configurable setting, but shows the general format of these files...

```
{
  "eyeRadius"     : 125,
  "eyelidIndex"   : "0x00", // From table: learn.adafruit.com/assets/61921
  "pupilColor"    : [ 0, 0, 0 ],
  "backColor"     : [ 140, 40, 20 ],
  "irisTexture"   : "hazel/iris.bmp",
  "scleraTexture" : "hazel/sclera.bmp",
  "upperEyelid"   : "hazel/upper.bmp",
  "lowerEyelid"   : "hazel/lower.bmp",
  "left" : {
  },
  "right" : {
  }
}
```

This is a plain text file that any simple editor should be able to handle...Notepad or TextEdit or whatever comes bundled on your computer.

It's worth reiterating these points from the "Customization Basics" page:

- The whole thing is contained within curly braces — { and }
- Each item has a **name** (always in quotes), a colon **separator** (**:**) and a **value**. Values might be numbers, strings (in quotes) or arrays (in square brackets) depending on what's being configured.
- This is a list and each line ends with a **comma**...except for the last item in the list. **Watch out for missing or surplus commas!**
- JSON is **case-sensitive**. "Foo", "foo" and "FOO" are all different things.
- Using extra spaces to line up columns really isn't necessary, just something I do to assist with legibility. Some folks just find it annoying.
- Comments (starting with //) are not standard JSON syntax, but the library we use allows them and I find them very helpful.

The "Customization Basics" page also has some **JSON troubleshooting tips**. If you encounter trouble, review that page!

Sizes and Shapes

Two settings define the **basic geometry** of the eye...

`eyeRadius` establishes the size of the overall eyeball, in **pixels**. This is a **radius** — center to edge — so the overall eye size is twice this across. For example `eyeRadius:125` configures the eye to be 250 pixels wide. This is the default if left unspecified.

The screens are only **240 pixels** wide. Reason the eye is made a little bigger is because the code uses tricks to fake a rotating sphere...and that faking is more apparent as the pupil approaches an edge. So we push the edge out a few extra pixels, then cover it up with eyelids.

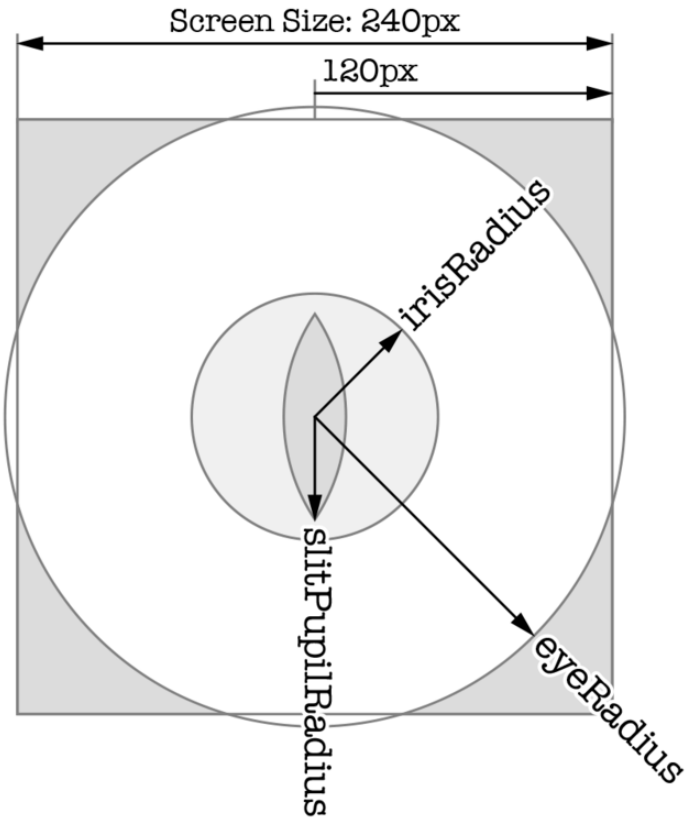
If designing an eye with **no eyelids**, you might want `eyeRadius:120` instead, which provides a nice **perfect circle** on the screen.

Remember that JSON is **case-sensitive**. This **must** be spelled `eyeRadius`. Different capitalization will cause it to be ignored!

`irisRadius` establishes the size of the **iris**...again a radius, in pixels.

`irisRadius:60` will make the iris 120 pixels across, or half the width of the screen. If you plan to use **lenses** over the displays, consider scaling down this number a bit to compensate.

Some creatures...cats and so forth...have very large irises and almost no visible sclera. In that case you can set `irisRadius` much larger, up to (but not exceeding) `eyeRadius`.



A third setting, `slitPupilRadius`, lets you make cat or dragon type eyes with a vertical slit pupil (only a vertical slit is available, no goat pupils, sorry). If set to **0** (the default), a normal **round** pupil is used. Larger numbers (up to `irisRadius`) make a taller/thinner pupil. This number sets the **height**. You'll probably want an in-between value...maybe `irisRadius:80` (160 pixels round) and `slitPupilRadius:60` (120 pixels tall) to start.

Note that using `slitPupilRadius` makes the program **a bit slower to initialize**... you'll just see blank screens for several seconds while it works. This is normal and just an unfortunate math thing.

Colors and Textures

The **texture map images** for **iris** and **sclera** are specified with `irisTexture` and `scleraTexture`:

```
  "irisTexture"   : "hazel/iris.bmp",
  "scleraTexture" : "hazel/sclera.bmp",
```

If you prefer a **solid color** to an image, omit those lines and instead use `irisColor` and/or `scleraColor`, with color specified as three values [red, green,blue] in brackets:

```
  "scleraColor"   : [ 255, 255, 255 ],
```

Colors can either be **three integers** in the range **0 to 255** (like most folks are used to) or three **floating-point** values from **0.0 to 1.0** (same idea, just different scale).

Solid colors save a TON of space compared to texture maps, if your design can get away with it.

A few more items have configurable colors:

```
  "pupilColor"    : [ 0, 0, 0 ],
  "backColor"     : [ 140, 40, 20 ],
```

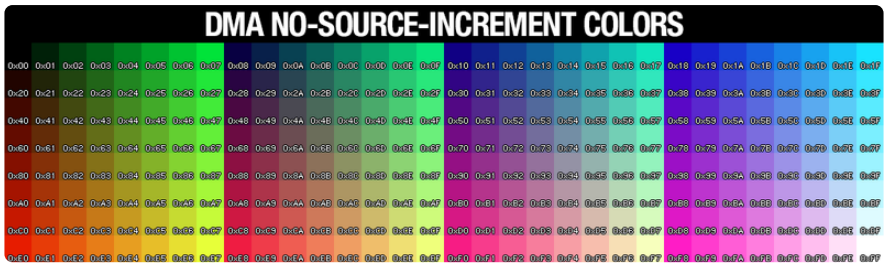
`pupilColor` is self-explanatory...like if you want glowing red or white pupils or something.

`backColor` covers the outermost/backmost part of the eye where the sclera texture map (or color) doesn't reach. With a little planning, this and your sclera texture map can be designed to blend together...otherwise you'll see a conspicuous crescent of `backColor` when the eye is looking off to a side.

The eyelid (and background) color is also configurable, but it's **not a normal RGB color**. Instead, use `eyelidIndex` and an 8-bit value:

```
  "eyelidIndex"   : "0x00",
```

Notice the hexadecimal value must be **quoted**. The value corresponds to one of the colors in this palette:



`eyelidIndex` doesn't use a normal RGB value because the code is using an optimization trick here, which limits the available colors in this one area. The default `eyelidIndex` if unspecified is "**0x00**" — black.

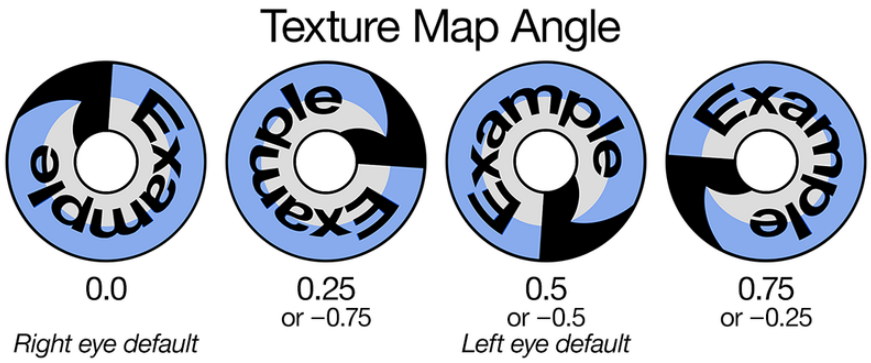
Distinct Left/Right Eyes

Colors/textures can be independent on the left and right eyes. Notice in the config file there's these two empty sections:

```
  "left" : {
  },
  "right" : {
  }
```

Between each item's curly braces, independent `irisTexture`, `scleraTexture`, `irisColor`, `scleraColor`, `pupilColor` and `backColor` can be specified, which will override anything set outside of these.

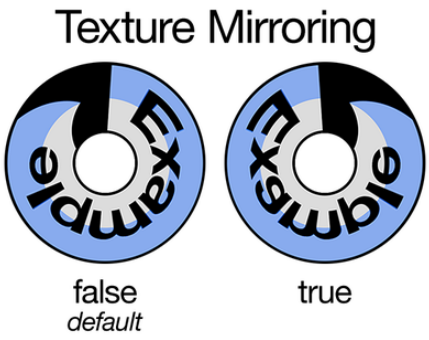
It was mentioned earlier that the left eye's textures are rotated 180° so it's less obvious that the images are being re-used. If you need to override this, so both eyes have aligned textures, the `irisAngle` and `scleraAngle` settings can be used (either globally or in the `left` / `right` sections).



`irisAngle` and `scleraAngle` are a rotation offset in the clockwise direction, using either normalized floating point where 0.0=seam at top, 0.25=seam at right, 0.5=bottom, 0.75=left, and 1.0=back to top. You can also use integer units from 0 to 1024 (0=top, 256=right and so forth). Negative values are accepted for both the floating-point and integer modes, in which case the angle will be counter-clockwise.

`irisSpin` and `scleraSpin` can be used to make the texture maps revolve around the pupil, which might be handy for certain fun effects (spiral hypno-eyes, cyborg stuff, etc.). These are specified in RPM, with positive values being clockwise. So for example, setting `irisSpin` to -60 will make the iris rotate counter-clockwise once per second. These can be specified globally or on a left/right basis.

There's also `irisMirror` and `scleraMirror` settings (specify as true or false, or 1 or 0 — default setting is false), which flip the way the texture is wrapped (clockwise default vs counterclockwise override). If mirroring on just one eye, you might want to set the corresponding `irisAngle` or `scleraAngle` value to 0 depending on what you're after.



Screen orientation is specified with the `rotate` keyword, with a value from 0 to 3 (default is 3). This can be helpful if you've **split** your MONSTER M4SK in two. Sometimes the mask-halves will fit into a project better if they're each turned 90 degrees, and this setting compensates so the eyes are right-side-up again. You can put separate `rotate` values in the `left` and `right` sections...try 0 and 2 (or 2 and 0, depending which way the eyes are turned).

None of this `left`/`right` stuff applies on the HalloWing M4, which only has one eye. Having these sections won't cause the config load to fail, it just ignores anything eye-specific.

The following are NOT left/right configurable: `eyeRadius`, `irisRadius` and `slitPupilRadius` (these affect how certain tables are calculated and there isn't enough RAM to have independent left and right tables).

Eyelids

The eyelid graphics format is explained on the "Preparing Graphics" page. These are 1-bit BMPs, 240 pixels square. Use the `upperEyelid` and `lowerEyelid` settings to specify the filenames of these images.

Eyelids are one of those global (not configurable per-eye) things. The design of the eyelid graphics might be symmetrical or asymmetrical...some eye designs might provide two sets of eyelid images, one for a single-eye device (like HalloWing M4) and another for the Monster M4SK, where the left and right eyelid shapes are mirrored.

If you want no eyelids at all, just leave out any `upperEyelid` or `lowerEyelid` filenames. You'll then have a circular unblinking eye...looks good for skulls!

With eyelids enabled, normally the upper lid "tracks" the movement of the pupil (when the eye looks down, the eyelid follows with it). This is something that eyes do in real life...but some folks think it looks sleepy, or just want a particular caffeinated look. Use the `tracking` keyword with a value of `false` to disable the eyelid tracking and maintain wide-open eyes.

Light Sensor

The pupils normally do some dilation movement on their own...but you can have them **respond to light** if you like! Use the `lightSensor` keyword along with a pin number where the light sensor is connected. If it's on a Seesaw interface chip (e.g. on MONSTER M4SK), add 100 to the pin number. For example, on **MONSTER M4SK** the built-in light sensor is:

```
"lightSensor" : 102
```

And for **HalloWing M4**, use:

```
"lightSensor" : 21
```

(Add a trailing comma if this appears mid-file.)

Voice Changer

This used to be a separate program...it now works together with the eyes! This requires the following:

- **MONSTER M4SK** board (http://adafru.it/4343)
- **PDM microphone** (http://adafru.it/4346) and JST SH cable **(http://adafru.it/4210)**
- You can test with **headphones**...but for portable or costume use, you'll want a **battery-operated amplified speaker** and a male/male 3.5mm audio cable (http://adafru.it/2698). Some Bluetooth boom box speakers include an aux input jack,

or I've used this belt-worn speaker from Monoprice (https://adafru.it/FNT) with some success (see notes below).

Usage

The **PDM microphone** connects using a tiny 4-pin cable to the "PDM MIC" port on MONSTER M4SK — it's near the reset button. You can optionally fashion a pop filter over the mic using a little fabric or foam, it'll probably sound better.

Connect an audio cable from MONSTER M4SK **headphone jack** to the **aux input** on the powered speaker.

The voice changer is **off by default**! It saps a fair bit of compute cycles (anywhere from about 25 to 50 percent...with a corresponding drop in eye animation frame rates) so you'll have to turn this on only if you really want it. To do so, you'll add a line to the `config.eye` JSON file on the root level of your MONSTER M4SK. Use:

```
"voice" : true
```

to enable the voice changer. See the link below (https://adafru.it/FSh) for an example config file that's been set up with voice changer parameters. Add a trailing comma if it's not the last line.

There are **three buttons** along the top edge of the monster's left eye. Tapping the **inner button** (the one closest to the nose) **raises** the pitch by 5%. Tapping the **outer button** (near the corner) **lowers** the pitch by 5%. Tapping the **middle button** resets the pitch to its **default**.

The **default pitch** is set with the `pitch` keyword. This is a floating-point value, where 1.0 is normal (voice is passed straight through, no change), 2.0 will double the

frequency (raising the voice by one octave), 0.5 will halve the frequency (lowering by one octave).

`pitch` can be from 0.4 to 4.0...but the actual usable range where you can still understand things is a bit narrower, perhaps 0.6 to 2.0...you'll want to experiment a bit to find a setting that achieves the desired effect with your own voice.

Microphone gain (sensitivity) is set with the `gain` keyword. If installed in a mask and you need to adjust the microphone to compensate for its placement relative to your mouth, use this with a floating-point value where 1.0 is "normal" sensitivity, 0.5 is quieter by half, 2.0 is double the loudness and so forth. There are limits to what can be done here, you may want to experiment a bit with this setting and the volume of an external amplified speaker.

Don't shout! **Speak in a normal to soft voice**, let the speaker take care of amplification. This helps the "weird" voice be heard over your own.

Similarly...**speak at your normal voice pitch** and let the voice changer do its thing. You don't need to make a funny voice.

Need a **Dalek** voice effect? With the voice changer enabled as described above, also add `"waveform" : "sine"` to enable this effect, which applies a **30 Hz sine wave modulation** to the pitch-adjusted voice — same as used for the original Dr Who Daleks. You can try other waveforms (`"square"`, `"sine"`, `"tri"` and `"saw"` are all supported) and other modulation frequencies (`"modulate" : 100` for a 100 Hz modulation wave)...but, to be perfectly honest...this all turned out a bit disappointing, the feature is only left in there because the 30 Hz Dalek modulation was spot-on. With some experimentation with different `pitch` and `modulation` settings you might also get a passable "Chicken, fight like a robot!" voice from Berzerk, if anyone even remembers that one.

Example Config.eye File

```
{
  // Doom-spiral eyes with voice changer
  "voice"          : true, //Turns on voice changer
  "waveform"       : "sine" , //Modulates voice with sine wave
  "modulate"       : 55 , //Modulation wave freq. in Hz
  "eyeRadius"      : 125,
  "eyelidIndex"    : "0x00", // From table: learn.adafruit.com/assets/61921
  "irisRadius"     : 125,    // Iris = whole eye!
  "pupilMin"       : 0,      // Pupil is always 0 size
  "pupilMax"       : 0,
  "pupilColor"     : [ 255, 255, 169 ], // Shouldn't show, but just in case
  "scleraColor"    : [ 255, 0, 0 ],
  "backColor"      : [ 255, 0, 0 ],
  "irisTexture"    : "doom-spiral/spiral.bmp",
  // The doom-red and doom-spiral eyelid bitmaps don't fully close.
```

```
  // This is to give the IMPRESSION of a blink without actually blinking,
  // so human eye behind is hidden better when doing Pepper's ghost trick.
  "upperEyelid"    : "doom-spiral/upper.bmp",
  "lowerEyelid"    : "doom-spiral/lower.bmp",
  "left" : {
    "irisSpin"     : 80    // Rotate iris @ 80 RPM
  },
  "right" : {
    "irisMirror"   : true, // Flip spiral image
    "irisSpin"     : 70    // Slightly different speed for weirdness
  }
}
```

Tips for using the Monoprice 5-Watt Guitar Amplifier

I have a love/hate thing with this speaker. On the plus side: it's pretty inexpensive, is rechargeable, and is slim(ish) and clips to one's belt or a lanyard, making it handy for costume use.

It's really designed for guitar use and MP3 playback (from microSD card) and there's some hoops necessary to get it to pass through audio undistorted...

- Connect MONSTER M4SK to the **AUX** phono jack (center of three), not the MIC input.
- After powering on, wait a moment and then press the "**M**" button to pass through audio.

You can see in the photo that I've labeled mine and highlighted the correct jack and button...I use it infrequently and forget this ritual (also helps when others are borrowing it).

This is not an Adafruit product and we do not provide support. Please check with Monoprice if you encounter trouble.

Building Eyes from Source Code

The next couple of pages explain how to set up the Arduino software to work with the SAMD microcontroller boards like the HalloWing M4 or MONSTER M4SK. **If you've previously worked with Adafruit SAMD boards with Arduino, that part's already set up and you can skip ahead to the "Libraries and Settings" page.**

In summary: use the latest Arduino IDE, install the latest Adafruit SAMD boards package, install Windows drivers if necessary, and install all the library dependencies for this project...

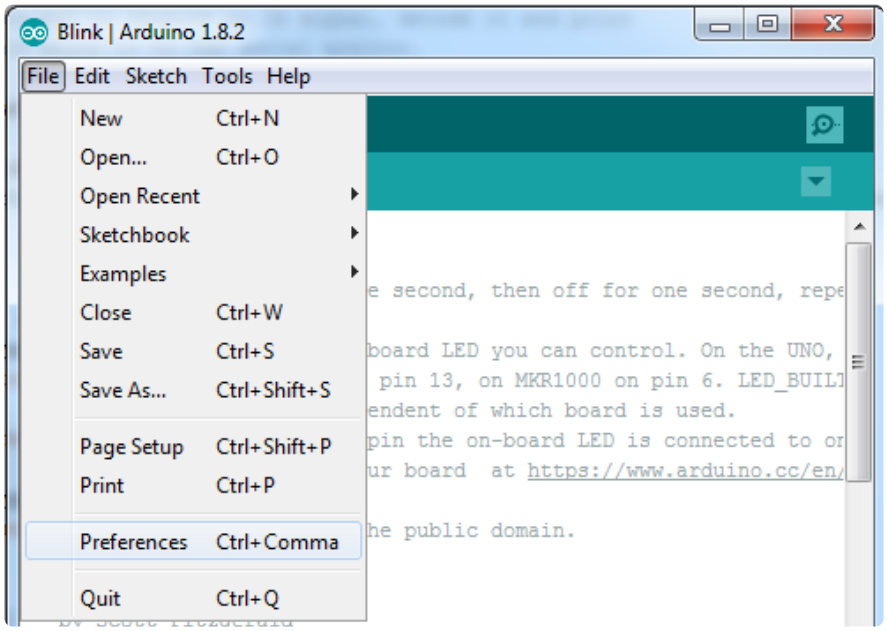
Arduino IDE Setup

The first thing you will need to do is to download the latest release of the Arduino IDE. You will need to be using **version 1.8** or higher for this guide

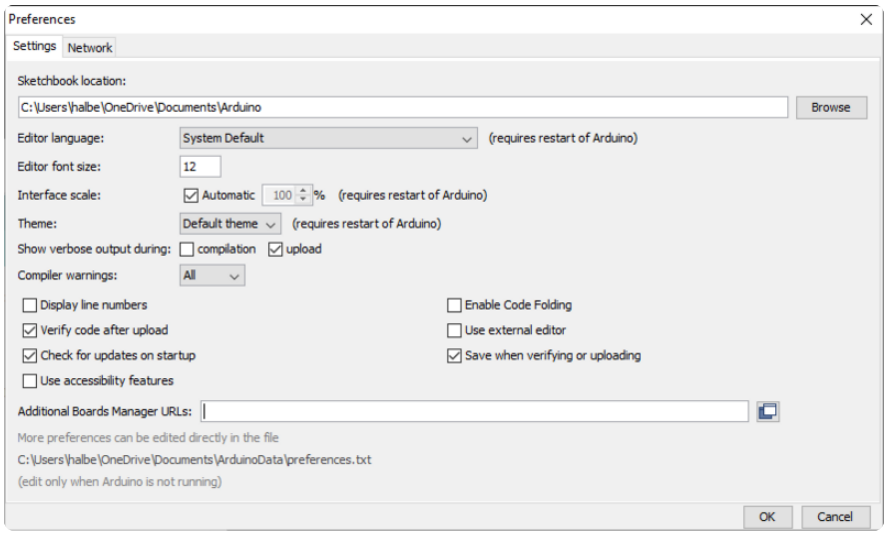
Arduino IDE Download

https://adafru.it/f1P

After you have downloaded and installed **the latest version of Arduino IDE**, you will need to start the IDE and navigate to the **Preferences** menu. You can access it from the **File** menu in Windows or Linux, or the **Arduino** menu on OS X.



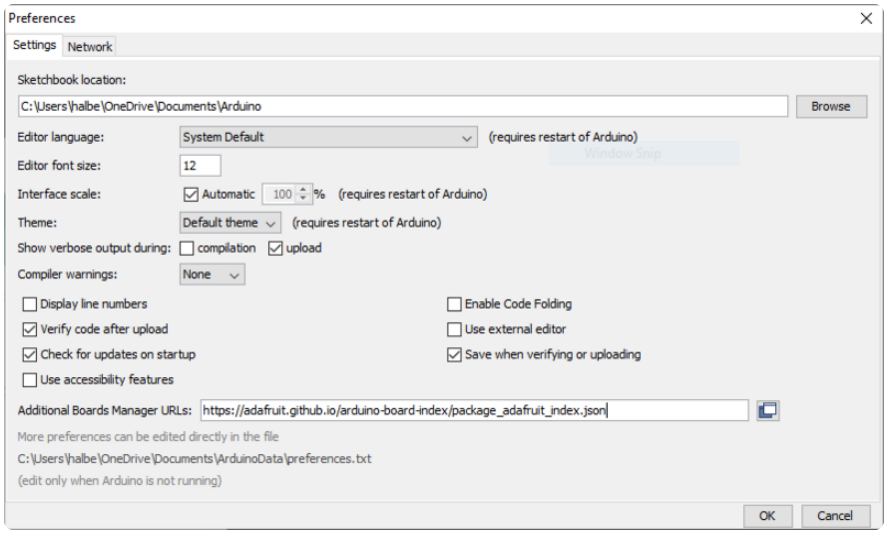
A dialog will pop up just like the one shown below.



We will be adding a URL to the new **Additional Boards Manager URLs** option. The list of URLs is comma separated, and you will only have to add each URL once. New Adafruit boards and updates to existing boards will automatically be picked up by the Board Manager each time it is opened. The URLs point to index files that the Board Manager uses to build the list of available & installed boards.

To find the most up to date list of URLs you can add, you can visit the list of third party board URLs on the Arduino IDE wiki (https://adafru.it/f7U). We will only need to add one URL to the IDE in this example, but **you can add multiple URLS by separating them with commas**. Copy and paste the link below into the **Additional Boards Manager URLs** option in the Arduino IDE preferences.

```
https://adafruit.github.io/arduino-board-index/package_adafruit_index.json
```



Here's a short description of each of the Adafruit supplied packages that will be available in the Board Manager when you add the URL:

- **Adafruit AVR Boards** - Includes support for Flora, Gemma, Feather 32u4, ItsyBitsy 32u4, Trinket, & Trinket Pro.
- **Adafruit SAMD Boards** - Includes support for Feather M0 and M4, Metro M0 and M4, ItsyBitsy M0 and M4, Circuit Playground Express, Gemma M0 and Trinket M0
- **Arduino Leonardo & Micro MIDI-USB** - This adds MIDI over USB support for the Flora, Feather 32u4, Micro and Leonardo using the arcore project (https://adafru.it/eSI).

If you have multiple boards you want to support, say ESP8266 and Adafruit, have both URLs in the text box separated by a comma (,)

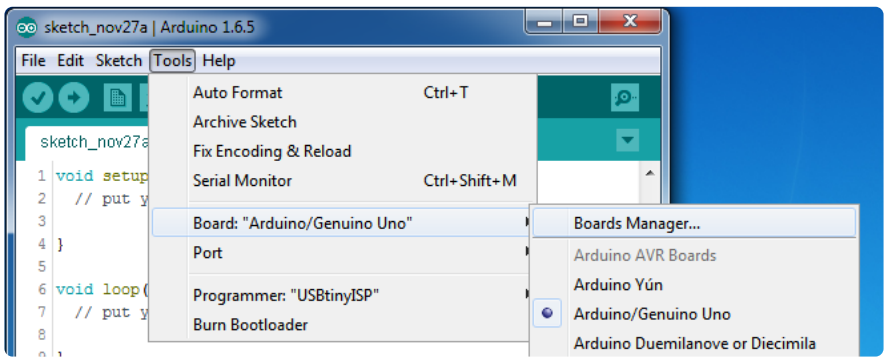
Once done click **OK** to save the new preference settings. Next we will look at installing boards with the Board Manager.

Now continue to the next step to actually install the board support package!

Using with Arduino IDE

Adafruit boards that use ATSAMD21 ("M0") or ATSAMD51 ("M4") chips are easy to get working with the Arduino IDE. Most libraries (including the popular ones like NeoPixels and display) will work with those boards, especially devices & sensors that use I2C or SPI.

Now that you have added the appropriate URLs to the Arduino IDE preferences in the previous page, you can open the **Boards Manager** by navigating to the **Tools->Board** menu.



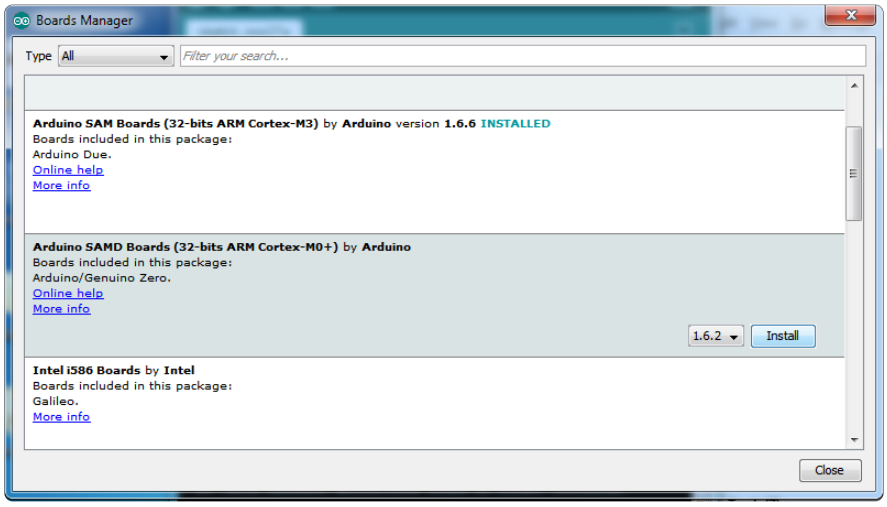
Once the Board Manager opens, click on the category drop down menu on the top left hand side of the window and select **All**. You will then be able to select and install the boards supplied by the URLs added to the preferences.

Remember you need SETUP the Arduino IDE to support our board packages - see the previous page on how to add adafruit's URL to the preferences

Install SAMD Support

First up, install the latest **Arduino SAMD Boards (**version **1.6.11** or later)

You can type **Arduino SAMD** in the top search bar, then when you see the entry, click **Install**

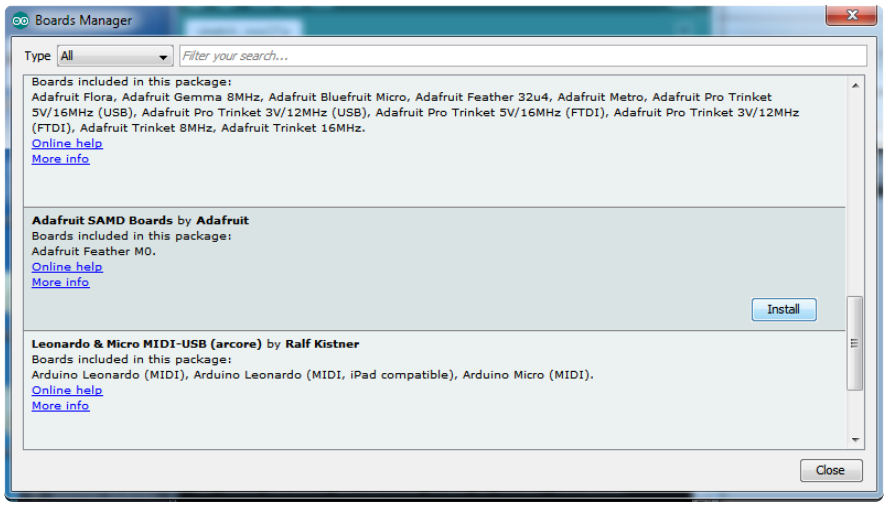


Install Adafruit SAMD

Next you can install the Adafruit SAMD package to add the board file definitions

Make sure you have **Type All** selected to the left of the Filter your search... box

You can type **Adafruit SAMD** in the top search bar, then when you see the entry, click **Install**

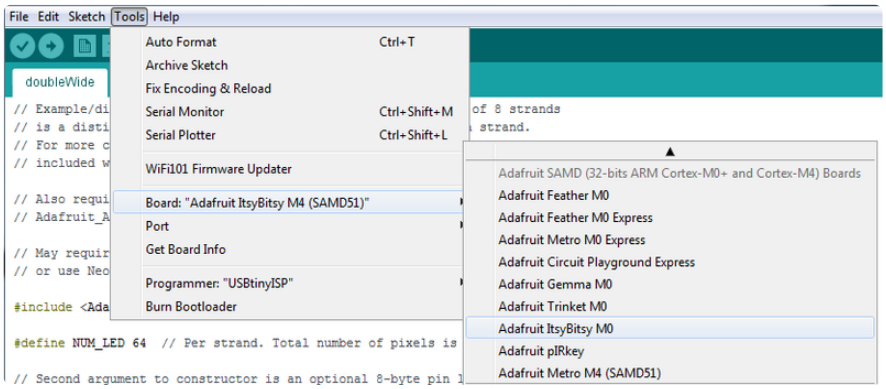


Quit and reopen the Arduino IDE to ensure that all of the boards are properly installed. You should now be able to select and upload to the new boards listed in the **Tools->Board** menu.

Select the matching board, the current options are:

- **Feather M0** (for use with any Feather M0 other than the Express)
- **Feather M0 Express**
- **Metro M0 Express**
- **Circuit Playground Express**
- **Gemma M0**
- **Trinket M0**
- **QT Py M0**
- **ItsyBitsy M0**
- **Hallowing M0**
- **Crickit M0** (this is for direct programming of the Crickit, which is probably not what you want! For advanced hacking only)
- **Metro M4 Express**
- **Grand Central M4 Express**
- **ItsyBitsy M4 Express**
- **Feather M4 Express**
- **Trellis M4 Express**
- **PyPortal M4**
- **PyPortal M4 Titano**
- **PyBadge M4 Express**
- **Metro M4 Airlift Lite**
- **PyGamer M4 Express**
- **MONSTER M4SK**
- **Hallowing M4**
- **MatrixPortal M4**

- **BLM Badge**



Windows 7 and 8.1

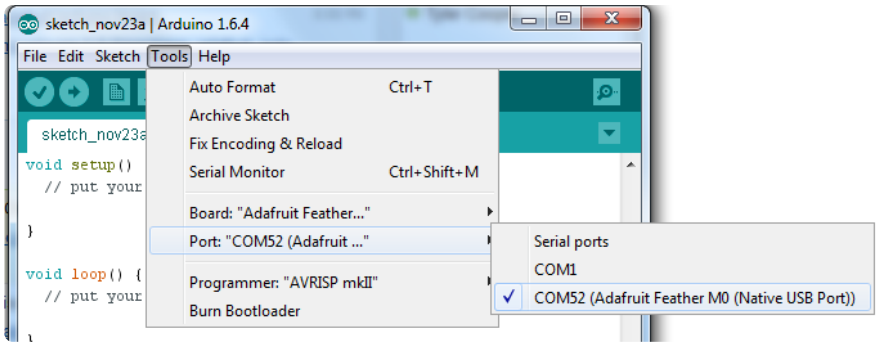
Windows 7 and Windows 8.1 have reached end-of-life and are no longer supported. They required driver installation. A limited set of drivers is available for older boards, but drivers for most newer boards are not available.

Blink

Now you can upload your first blink sketch!

Plug in the SAMD21 M0 or SAMD51 M4 board, and wait for it to be recognized by the OS (just takes a few seconds). It will create a serial/COM port, you can now select it from the drop-down, it'll even be 'indicated' as Trinket/Gemma/Metro/Feather/ItsyBitsy/QT Py/Trellis or whatever the board is named!

A few boards, such as the QT Py SAMD21, Trellis M4 Express, and certain Trinkey boards, do not have an onboard pin 13 LED. You can follow this section to practice uploading but you won't see an LED blink!



Now load up the Blink example

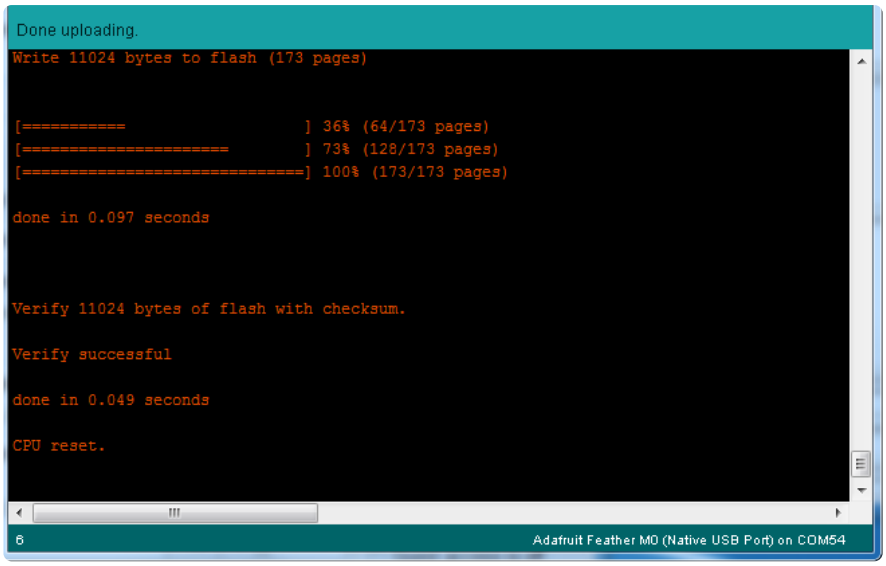
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

And click upload! That's it, you will be able to see the LED blink rate change as you adapt the **delay()** calls.

If you are having issues, make sure you selected the matching Board in the menu that matches the hardware you have in your hand.

Successful Upload

If you have a successful upload, you'll get a bunch of red text that tells you that the device was found and it was programmed, verified & reset



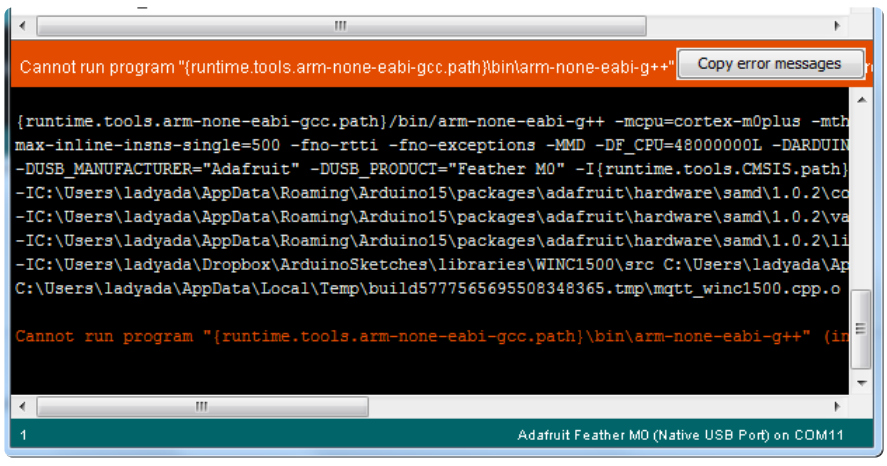
After uploading, you may see a message saying "Disk Not Ejected Properly" about the ...BOOT drive. You can ignore that message: it's an artifact of how the bootloader and uploading work.

Compilation Issues

If you get an alert that looks like

Cannot run program "{runtime.tools.arm-none-eabi-gcc.path}\bin\arm-non-eabi-g++"

Make sure you have installed the **Arduino SAMD** boards package, you need both Arduino & Adafruit SAMD board packages

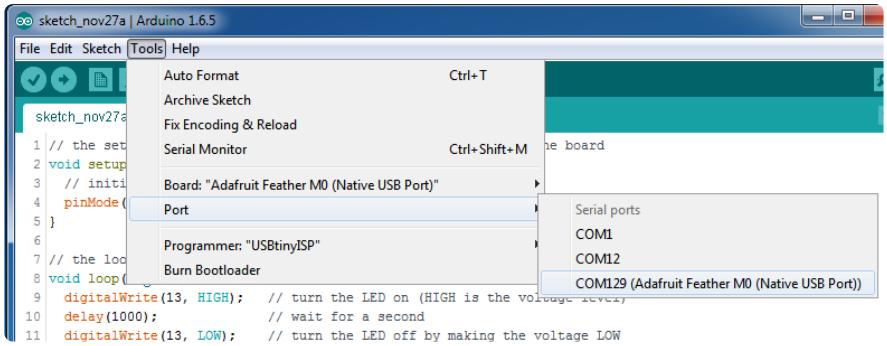


Manually bootloading

If you ever get in a 'weird' spot with the bootloader, or you have uploaded code that crashes and doesn't auto-reboot into the bootloader, click the **RST** button **twice** (like a double-click)to get back into the bootloader.

The red LED will pulse and/or RGB LED will be green, so you know that its in bootloader mode.

Once it is in bootloader mode, you can select the newly created COM/Serial port and re-try uploading.



You may need to go back and reselect the 'normal' USB serial port next time you want to use the normal upload.

Ubuntu & Linux Issue Fix

Follow the steps for installing Adafruit's udev rules on this page. (https://adafru.it/iOE)

Source, Libraries and Settings

At this point you should be able to upload code (such as the “Blink” sketch) to the board, the basics are confirmed working. If not, work though the prior two pages.

Source Code

Source code for this project is found in the Adafruit_Learning_System_Guides (https://adafru.it/Clx) repository on Github, specifically in the M4_Eyes (https://adafru.it/11uF) subdirectory. Here’s a direct download link:

Download M4_Eyes project from GitHub

https://adafru.it/FAD

Libraries

The following libraries are used by the eye code (or are referenced indirectly by libraries used). These can be installed through the Arduino Library Manager (**Sketch→Include Library→Manage Libraries**...)

- Adafruit_Arcada
- Adafruit_GFX
- Adafruit_BusIO
- Adafruit_ImageReader
- Adafruit_LIS3DH
- Adafruit_MSA301
- Adafruit_NeoPixel
- Adafruit_Seesaw
- Adafruit_Sensor (Adafruit Unified Sensor in Library Manager)
- Adafruit_SPIFlash
- Adafruit_ST7789
- Adafruit_TinyUSB
- Adafruit_TouchScreen
- Adafruit_ZeroDMA (Adafruit Zero DMA in Library Manager)
- Adafruit_ZeroPDM (Adafruit Zero PDM in Library Manager)
- Adafruit_ZeroTimer
- ArduinoJson (not Arduino_JSON)
- SdFat - Adafruit fork (not the standard SdFat fork)

Recent versions of the Arduino IDE install dependent automatically, so installing Ada fruit_Arcada should cover most if not all of this.

Additionally, depending on which of the “user” code tabs is active, you might need:

- Adafruit_AMG88xx (used by user_watch.cpp)

Project Settings

- Tools→CPU Speed→180 MHz (overclock) (200 MHz is a bit too much for some boards and may lock up, but you can give it a try. We use 180 MHz for our prepackaged .UF2 files since it's likely to work on more boards in the wild.)
- Tools→Optimize→Faster (-O3) (**do not** use higher settings for this code, it may corrupt the flash filesystem requiring a complete wipe)
- Tools→USB Stack→TinyUSB (the code will not compile without this selected!)

Troubleshooting

Board does not show up as drive when connected to computer

- The most common cause is using a **charge-only** USB cable rather than **charge-and-sync.** Try **swapping out for a different cable.**
- If you're not getting the **CIRCUITPY** filesystem: have you installed CircuitPython on the board to initialize the drive the first time? See CircuitPython instructions below.
- If you can't access the bootloader (MASKM4BOOT or HALLOWM4BOOT drive), it might be the timing of the reset double-tap. Try a little slower or a little faster.

Board Unresponsive; Flashing Red LED

There is an extremely small but nonzero chance of a glitch that wipes the board's flash filesystem. No eyes, just a flashing red LED. We've yet to track down the source of this bug. Fortunately it's pretty rare, and usually just when loading a new design.

If this happens with your board, you'll need to completely wipe and reinstate the flash filesystem. Any files on your board are lost … it's strongly recommended to **maintain a backup** of any customized files.

To recover from this situation:

1. Visit the **CircuitPython download page** (https://adafru.it/Em8) and search for your board — either **MONSTER M4SK** or **HalloWing M4** (not M0).
2. Download two files there: a CircuitPython "UF2" installer (current stable release), and bootloader updater UF2 (further down on the same page).

3. Connect board to computer with USB cable, set switch to "**on**" position, and **double-tap** the board's reset button. After a few seconds, you should see a small flash drive called **MASKM4BOOT** or **HALLOM4BOOT**.
4. Drag the bootloader UF2 file to the boot drive. That drive will go away for about 10 seconds while the software works, and then reappear once it's finished.
5. Drag the CircuitPython UF2 file to the boot drive. That drive will go away and is replaced in a few seconds with one called **CIRCUITPY**. You should see some text on the board's display now.
6. Reload eye graphics files to the CIRCUITPY drive, either one of the ready-made designs (https://adafru.it/FVx), or from a backup of your customized files.
7. Download and install the M4 EYES firmware as explained on this page (https://adafru.it/G8d). This is another double-tap-drag-UF2 sequence, like we did above.

Everything on your board should be factory fresh now.

Ol' Blue Eyes

If the eyes used to work but after editing have reverted to flat-colored blue irises, that's almost always a JSON syntax error...usually a missing or extra comma. The "Customization Basics" page has more tips for ensuring proper syntax.

Crossed Eyes

Occasionally the eyes might go **crossed** and **stay** that way. The software thinks the "nose booper" is being held and this is its attempt at being funny.

This false booping can happen when the **battery is running low** and its voltage starts to drop.

Occasionally though...it just gets a false reading, maybe due to gradual changes in temperature, humidity or other unknown factors. **Tap the reset button** (or toggle power off and on) and this should clear up...the booper calibrates it's un-booped state on start-up.

Technicolor Snow

If you see one or more of your texture maps show up as multicolored snow/noise, it may be an indication that you've run out of texture memory and will need to reduce the file size of some of your maps.

If the entire screen is snow, it usually means you've run out of RAM, and the most likely culprit there is the `eyeRadius` setting. Larger eyes require geometrically more RAM. Dial it back until everything fits.

Erase the Filesystem

If your board has a filesystem problem that's not fixed by the troubleshooting steps up top, you can try a more thorough QSPI flash initialization. Download the UF2 file linked below, then plug in your M4SK and double-click the reset button. You'll see a M4SKM4BOOT drive show up on your computer. Drag the M4SK_QSPI_Eraser.UF2 onto that drive. After a few moments, the board's drive will disconnect. Double-click reset to put the board back into bootloader mode. Then follow the CircuitPython, graphics and eye firmware steps above.

M4SK_QSPI_Eraser.UF2

https://adafru.it/FUE

Diagnostics

Want to see some stats on your M4SK? Double-click the board's reset button to get to bootloader mode, then drag the M4SK_Diagnostics.UF2 onto the M4SKM4BOOT drive!

M4SK_Diagnostic.UF2

https://adafru.it/FUF

Screen(s) Coming Unstuck

You might find over time that the foam tape holding one or both screens softens up... or the tape is fine, but the thin plastic light diffuser behind the LCDs is giving way. Use a toothpick to add dots of a pliable glue like E-6000 or Duco Cement between the PCB and side edges of the screen. Will usually need to dry overnight...follow directions and safety guidance on the glue packaging.

Lens Holders

Lens Holders

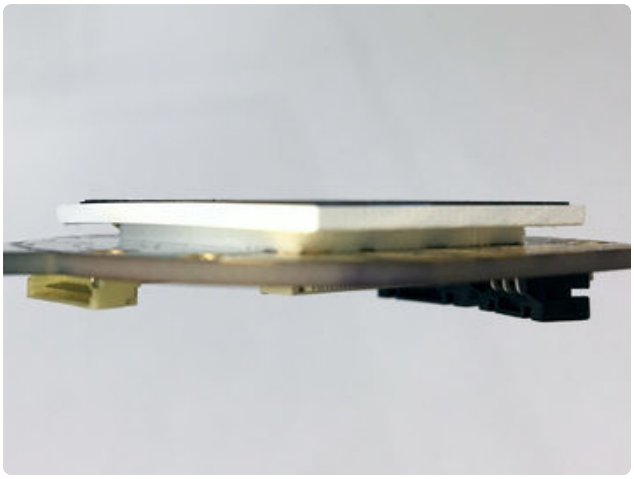
Here’s an Adobe Illustrator file for laser-cut 1/8" thick (3mm) acrylic that provides frames for two plastic (http://adafru.it/3917) or glass (http://adafru.it/3853) lenses:

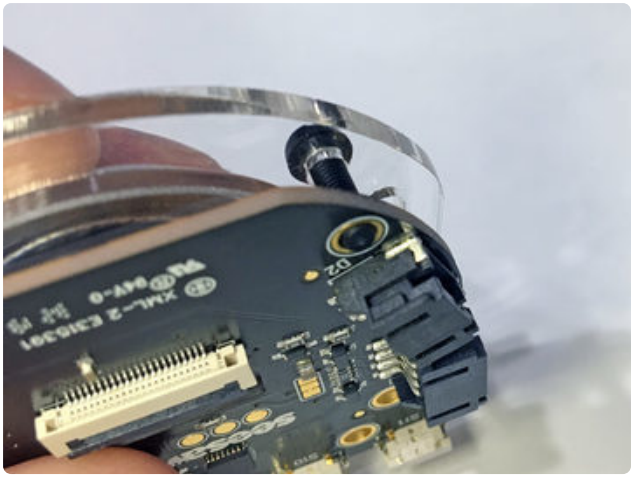
Monster_Lens_Holders.ai

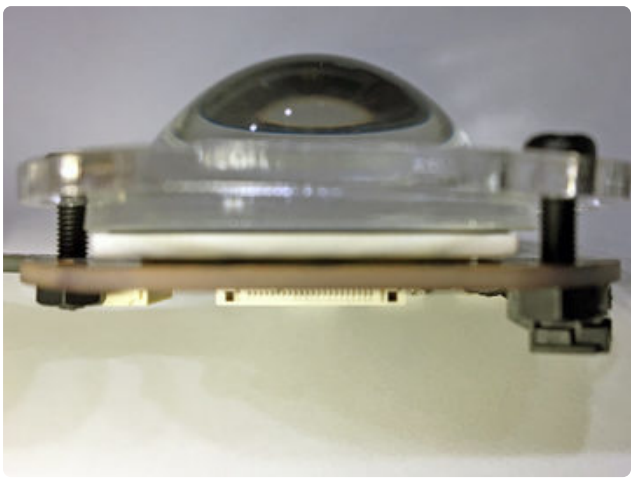
https://adafru.it/FB5

Each frame is held in place with **three M3 x ~12-14mm screws**. Slots are provided for a 3/4" (20mm) **elastic strap**. This can be worn around a hat or the forehead of a mask. **Not worn over your own eyes please**...it blocks your vision and might scratch your eyes!

The screws that come with the lens holder kit might be a little under 12mm, but they'll still fit - gently press on the lens to squish the foam as you install each hex nut at a time and dont go beyond finger-tightening.







A last-minute change added **foam tape** to secure the screens. The additional thickness meant that the “12mm-ish” screws won’t always make a solid connection. **Here’s what to do...**

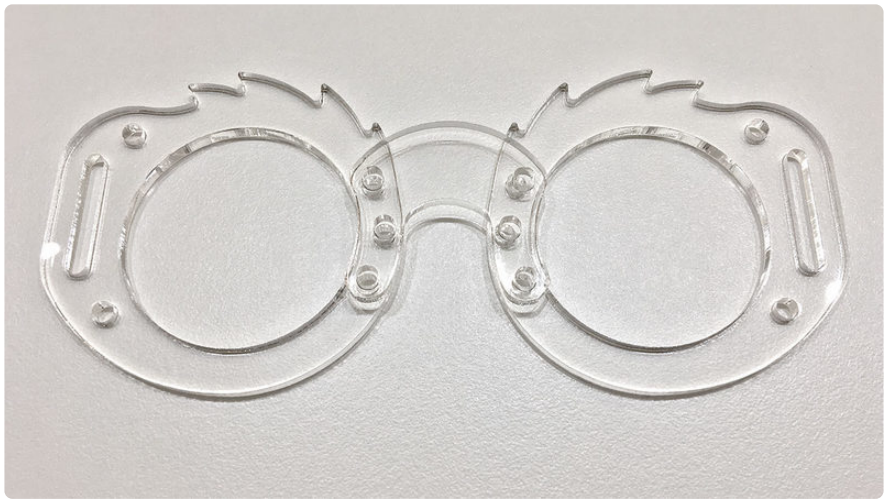
Remove the plastic screen protector if you haven’t already. Make sure the screen and back of the lens are **clean**. Align the lens holder over this.

Insert a screw, then **apply firm and even pressure** on the **lens** against the screen. With pressure **distributed across the whole screen** like this (using the lens), it won’t break.

It should be possible to push through just enough threads for the **nut** to **catch**.

Repeat with other screws. Once they’re all caught, move between the three screws giving each 1/4 to 1/2 turn at a time. The foam turns out to be quite compressible and it should be possible to get the tips of the screws **flush** with the face of the nuts.

You can skip the lenses and just use the frames for the strap if that's all you need. Regardless, **don't make the strap too snug**...the board is designed to be split in two, and we don't want that happening unintentionally.



There is an optional extra **nose bridge** piece. If you have **split** your MONSTER M4SK apart and want to use it in its original shape again, this piece holds the frames at their original spacing. An optional **9-pin JST cable** must link the two boards, you'll need some **different length** M3 screws, and you **will not get the boopable nose** feature back.

3D Printed Options

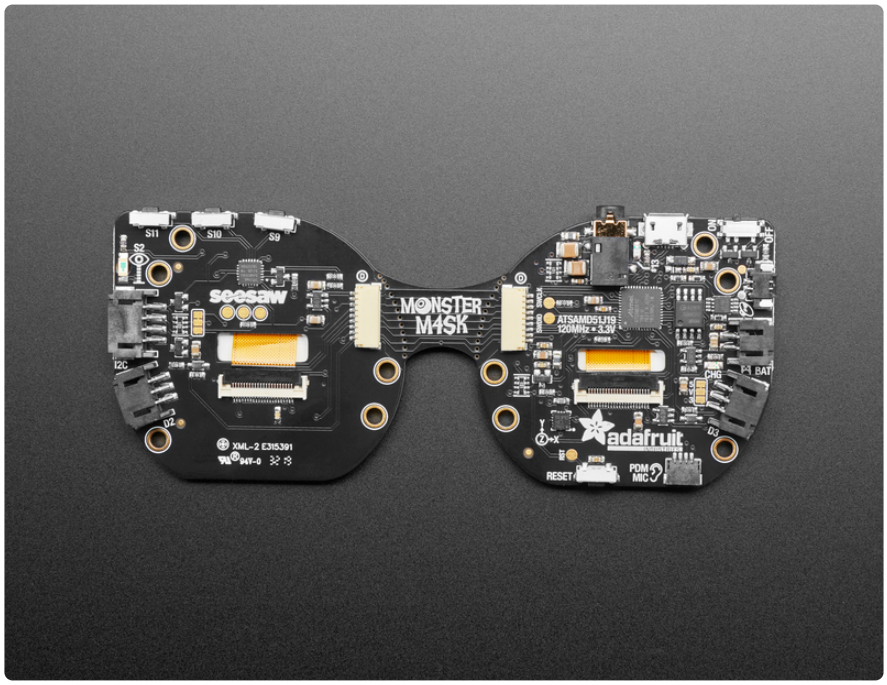
We also have a nice 3D-printable MONSTER M4SK case (https://adafru.it/FNy) — or use just the **lens holder rings** if that's the only part you need.

Voice Changer

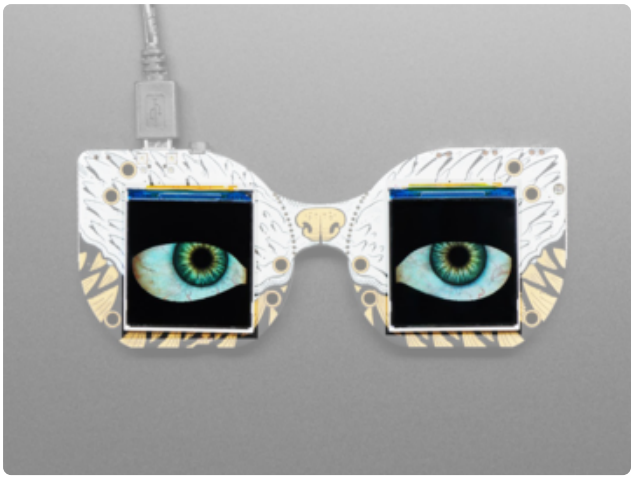
The voice changer is no longer a standalone program...it's been merged in with the eye code and **runs at the same time** now!

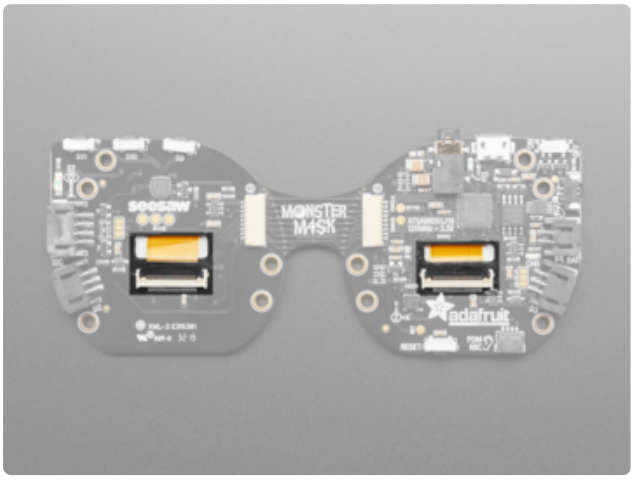
Enabling and customizing the voice changer are now documented on the Configurable Settings — Voice Changer (https://adafru.it/FSh) page.

Pinouts



Displays and Display Connectors



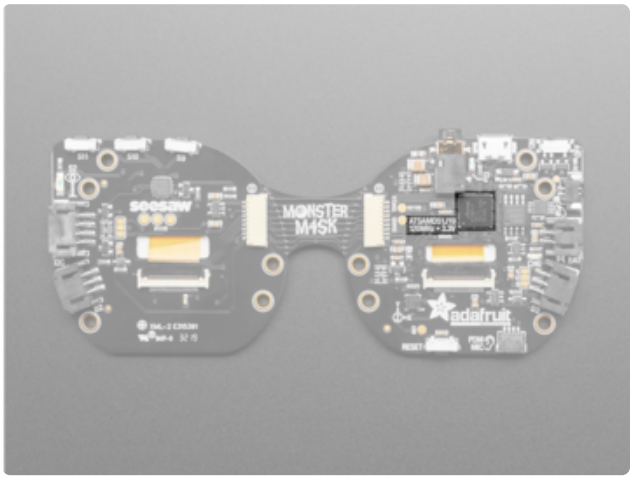


On the front are **two 240x240 IPS TFTs**.

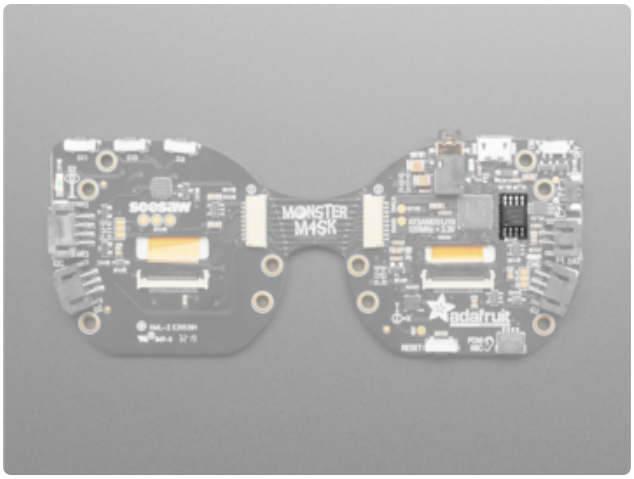
They are ST7789V chipset displays, with 240x240 pixels and backlight control. Each has an individual unshared SPI bus so we can DMA-blast pixels out as fast as possible.

The TFT displays are connected to the M4SK using **display connectors** located on the back of the board. The ribbon cables for the displays fit through a slot in the center of each side of the M4SK.

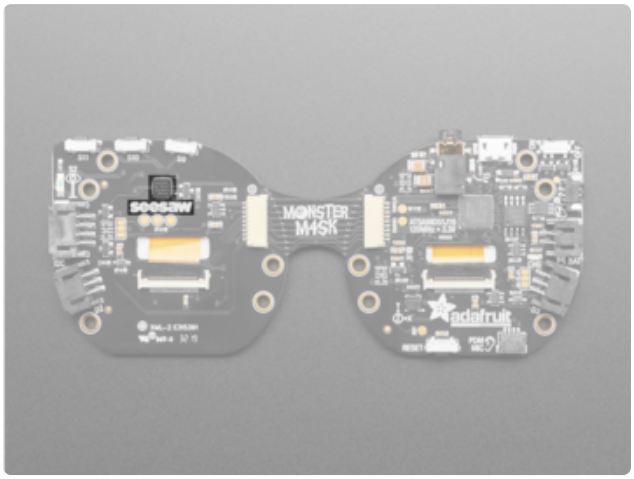
Microcontrollers and Flash



The main processor chip is the **ATSAMD51G19** Cortex M4 microcontroller running at 120MHz with 512KB Flash, and 192KB RAM.

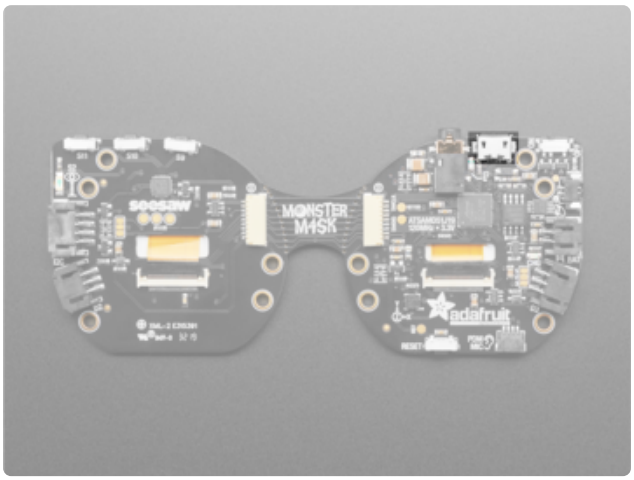


Towards the center left on the right side, next to the microcontroller, is **8 MB QSPI flash** for storing graphics and sound effects.



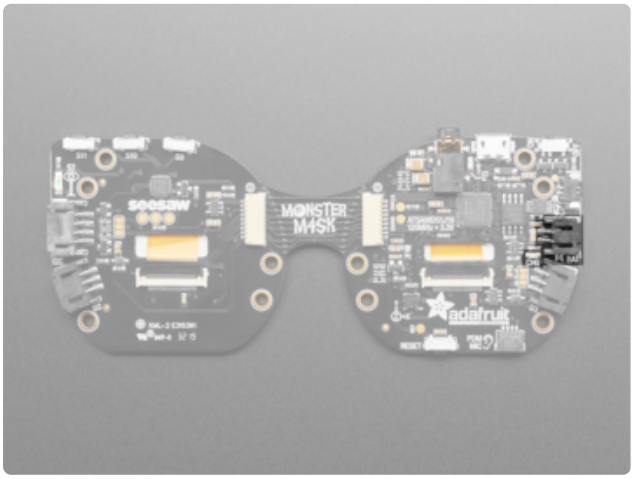
Towards the top center of the left side is the **seesaw chip**. This chip handles the tactile buttons and the light sensor. You can read the data from them using the I2C seesaw library and the pins specified. This is what lets us have only 9 wires to bridge the two halves of the PCB

Power Switch and Ports



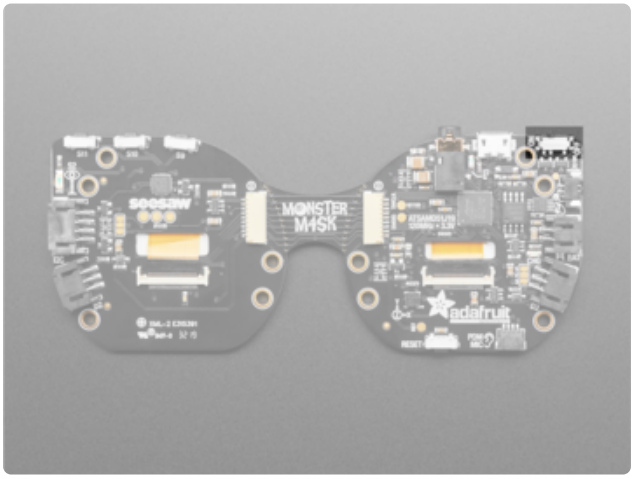
There is one USB port on the board.

On the top of the right side is a **USB Micro** port, which is used for powering and programming the board.



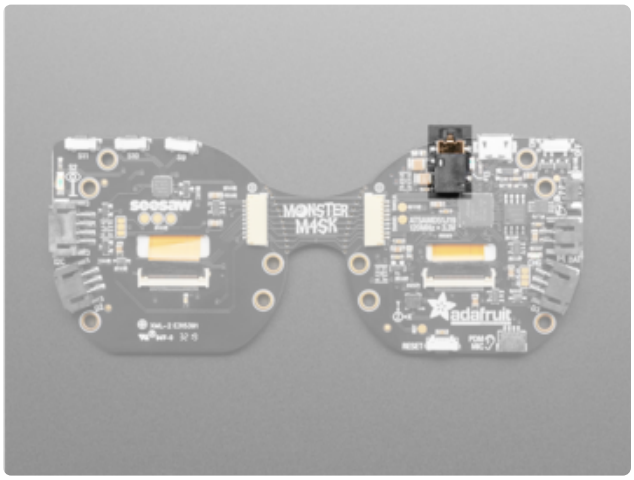
You can also power your MONSTER M4SK by plugging in a 3.7/4.2V Lipoly battery into the **JST 2-PH battery port** found in the center of the right side. You can then recharge the battery over the Micro USB jack. When charging, the **CHG LED** will light up.

It's normal for the yellow **CHG LED** to flicker when no battery is in place, that's the charge circuitry trying to detect whether a battery is there or not. If you are powering only over USB, you can cover it with tape

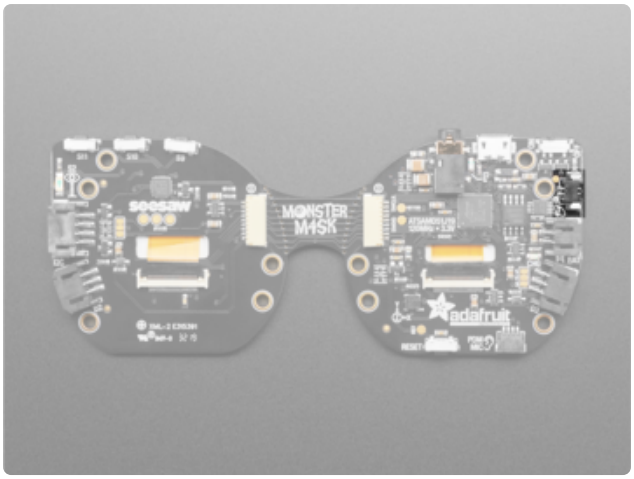


In the upper right corner on the right side is the **power switch**. You can power off the board when you're not using it to conserve power.

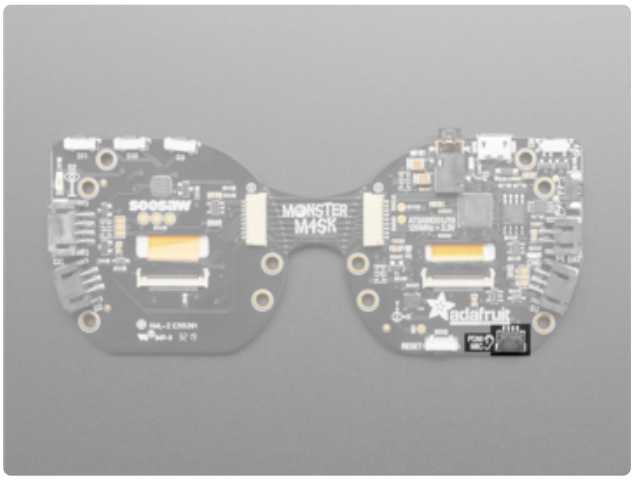
Audio



There is a **stereo headphone jack** on the top of the right side. This is connected to the two analog DAC pins on the SAMD51 - **A0** and **A1**



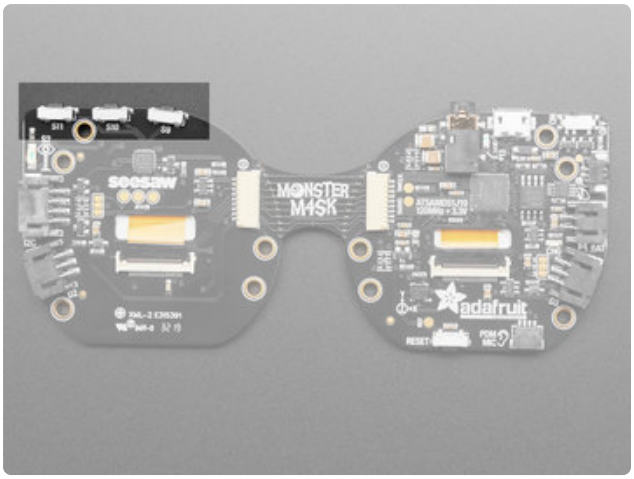
There is a mono speaker driver for smaller 8 ohm 1W speakers. Add a speaker using the Molex PicoBlade (https://adafru.it/C8p) **speaker connector** located on the right side of the board towards the top. The speaker is connected to the **A0** analog DAC, and can be enabled/disabled using the **SPEAKER ENABLE** pin (pin 20)



At the bottom on the right side is a **4 pin JST SH port** for connecting an optional **PDM microphone**. Note that the PDM microphone is connected to two SERCOM pins, and we use **SPI2** to read PDM data (if you think about it, PDM is just fixed-frequency SPI!) You could in theory turn this into an I2C or UART port. The SAMD51G19 chip used on this board does not support I2S, so we can't use I2S to read the PDM data.

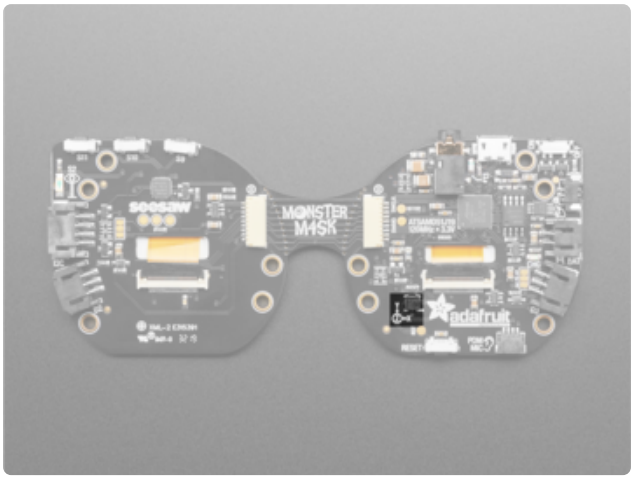
Capacitive Touch Pad and Buttons

Front and center, is a **capacitive touch pad nose** this is connected to **D2**. It's not a great capacitive touch input, because of the zener diode on D2's protection circuitry, but it can detect presses with some startup-calibration. Use as an input.

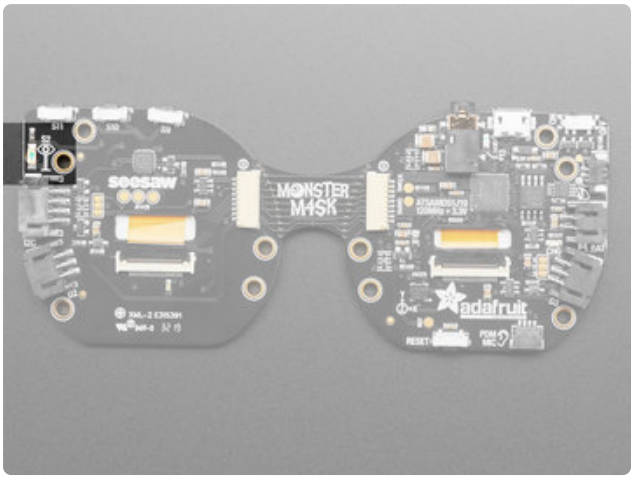


On the left side are **three tactile buttons** for user interfacing. They are connected to the **seesaw chip** on **pins 9, 10, 11**. You need to use the seesaw library to enable the pull-ups on all three and read the buttons (they will be connected to ground when pressed)

Sensors

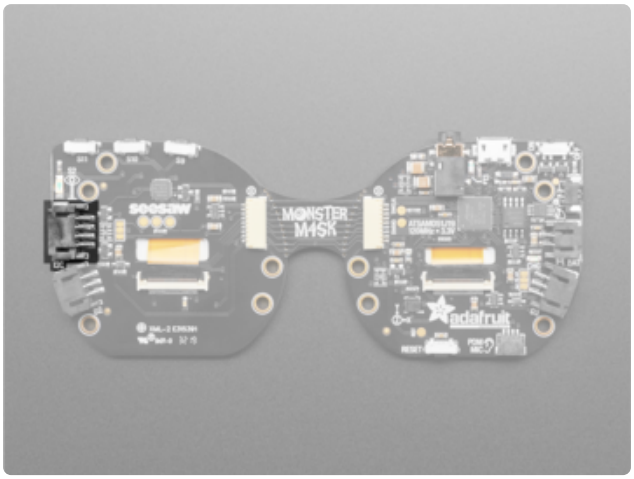


There is an **LIS3DH 3-axis accelerometer**, on the right side near the bottom towards the middle, connected to the I2C pins for detection motion, tilt or taps.



On the left side is a **light sensor**. It's connected to the **seesaw chip** on **pin 1**. Read the analog value using the seesaw library. Lower values like 50 mean its dark. Higher values like 1000 means its bright. It's reverse mount so you can read light levels from the front on the right side of the M4SK.

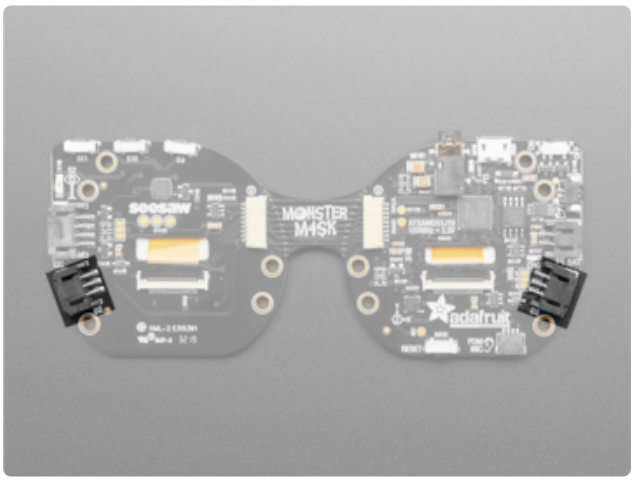
I2C Connector



There is a 4-pin JST **I2C connector** in the center on the left, that is STEMMA and Grove compatible. In CircuitPython, you can use the STEMMA connector with `board.SCL` and `board.SDA`, or `board.STEMMA_I2C()`. The I2C connection is shared with the seesaw chip and the accelerometer.

The I2C connector defaults to 5V. There is a jumper located immediately to the right of the connector that you can cut or solder to change it between 5V and 3V.

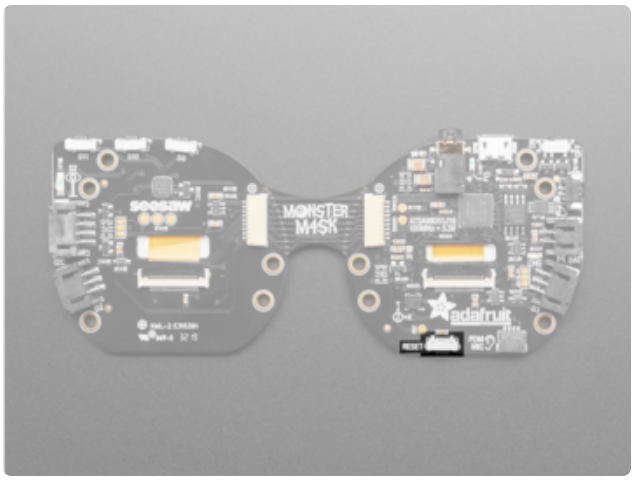
Digital/Analog Connectors



On the right and left sides, towards the bottom, are two connectors labeled **D2** and **D3**. These are **3-pin JST digital or analog connectors** for sensors or NeoPixels. These pins can be analog inputs or digital I/O.

They have protection 1K resistors + 3.6V zener diodes so you can drive an LED directly from the output. Connect to them via `board.D2` and `board.D3` or Arduino `2` and `3`. For analog reading in Arduino use **A2** for **D2** and **A3** for **D3.**

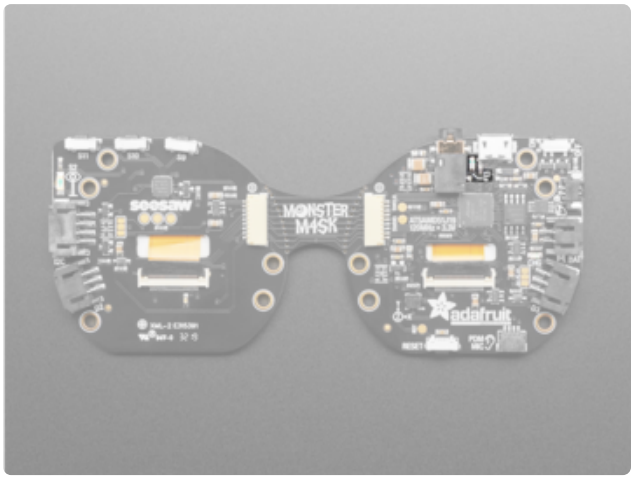
Reset Button



The **reset button** is located on the bottom of the right side.

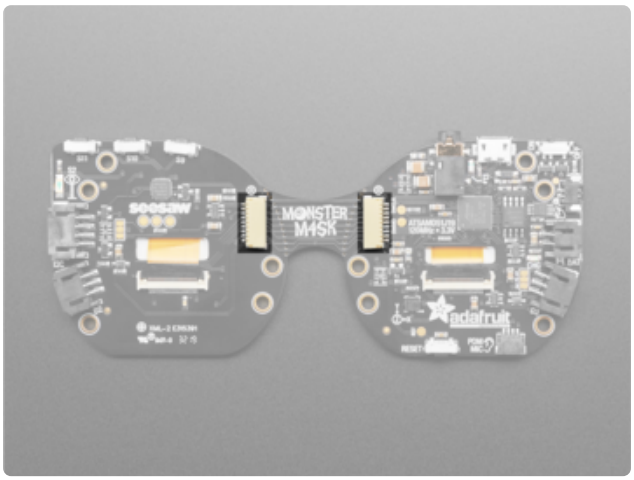
Click it once to re-start your firmware.
Click twice to enter bootloader mode.

LED



On the top of the right side, next to the USB connector, is a **little red LED** tied to pin **D13**. You can control it in your code. It pulses when the board is in bootloader mode.

SECTION TITLE HERE



Located on the inside edge of each side is a **9-pin JST SH connector**. This connector works with a 9-pin cable and allows you to break the board apart along the perforation, and mount the two halves of the board separately from each other!

Downloads

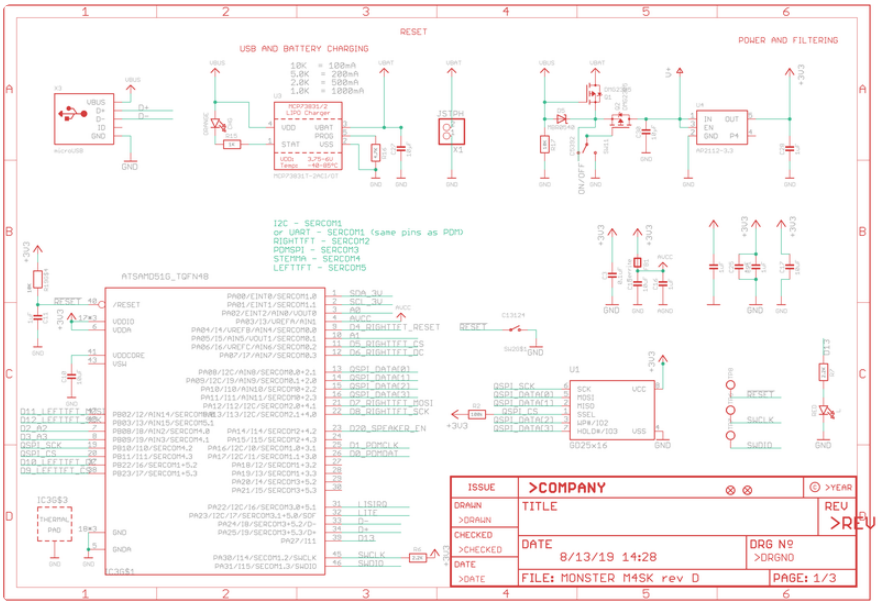
Files:

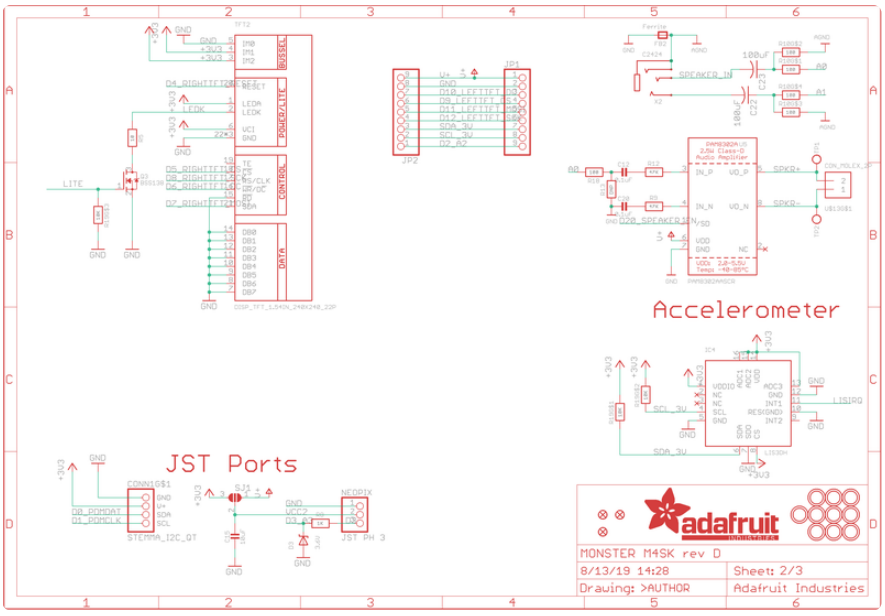
- Product page with datasheet for ATSAMD51J19 (https://adafru.it/Bf8)
- EagleCAD files on GitHub (https://adafru.it/FTq)
- 3D Models on GitHub (https://adafru.it/FN2)
- Fritzing object in the Adafruit Fritzing library (https://adafru.it/FTr)

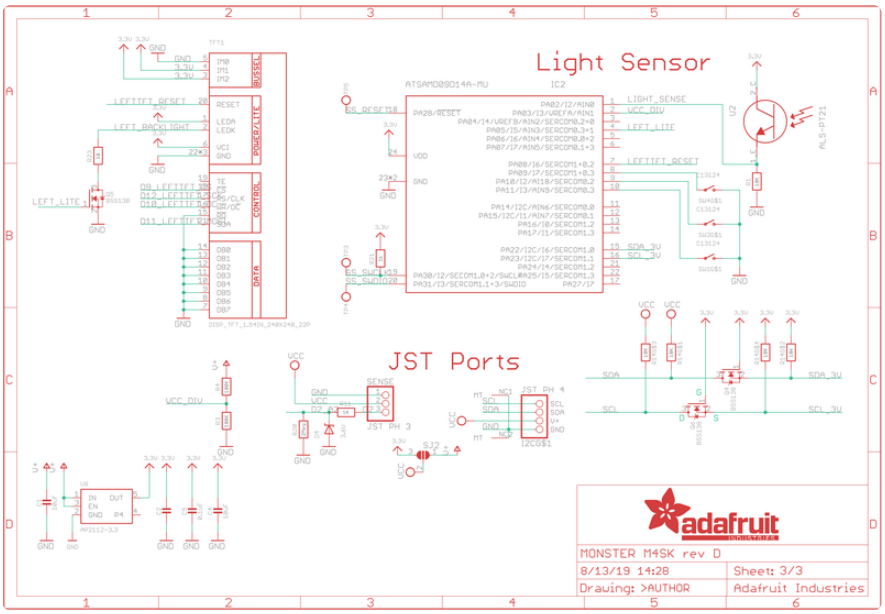
seesaw_firmware.zip

https://adafru.it/19CA

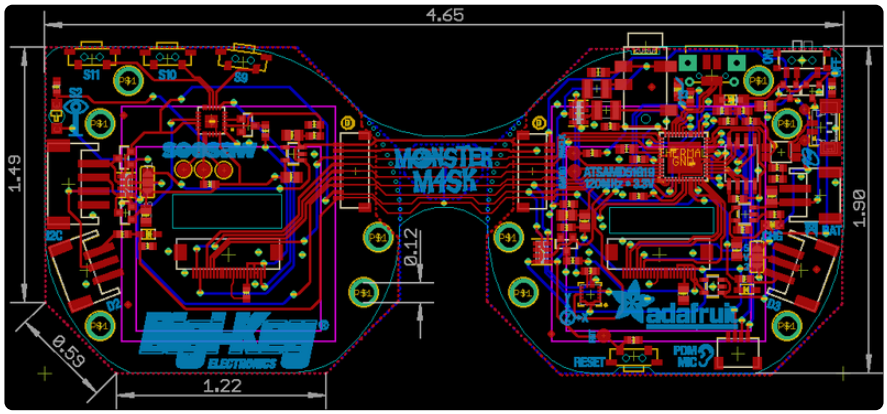
Schematic


RESET
USB AND BATTERY CHARGING
POWER AND FILTERING
I2C - SERCOM1
or UART - SERCOM1 (same pins as PDM)
RIGHTTFT - SERCOM2
PDMSPI - SERCOM3
STEMMA - SERCOM4
LEFTTFT - SERCOM5
ATSAMD51G_TQFN48
ISSUE
>COMPANY
TITLE
REV
>REV
DRAWN
>DRAWN
CHECKED
>CHECKED
DATE
>DATE
8/13/19 14:28
DRG Nº
>DRGNO
FILE: MONSTER M4SK rev D
PAGE: 1/3


JST Ports
Accelerometer
MONSTER M4SK rev D
8/13/19 14:28
Sheet: 2/3
Drawing: >AUTHOR
Adafruit Industries


Light Sensor
JST Ports
adafruit
MONSTER M4SK rev D
8/13/19 14:28
Sheet: 3/3
Drawing: >AUTHOR
Adafruit Industries

Fab Print


4.65
1.49
1.90
0.12
0.59
1.22
MONSTER M4SK
adafruit