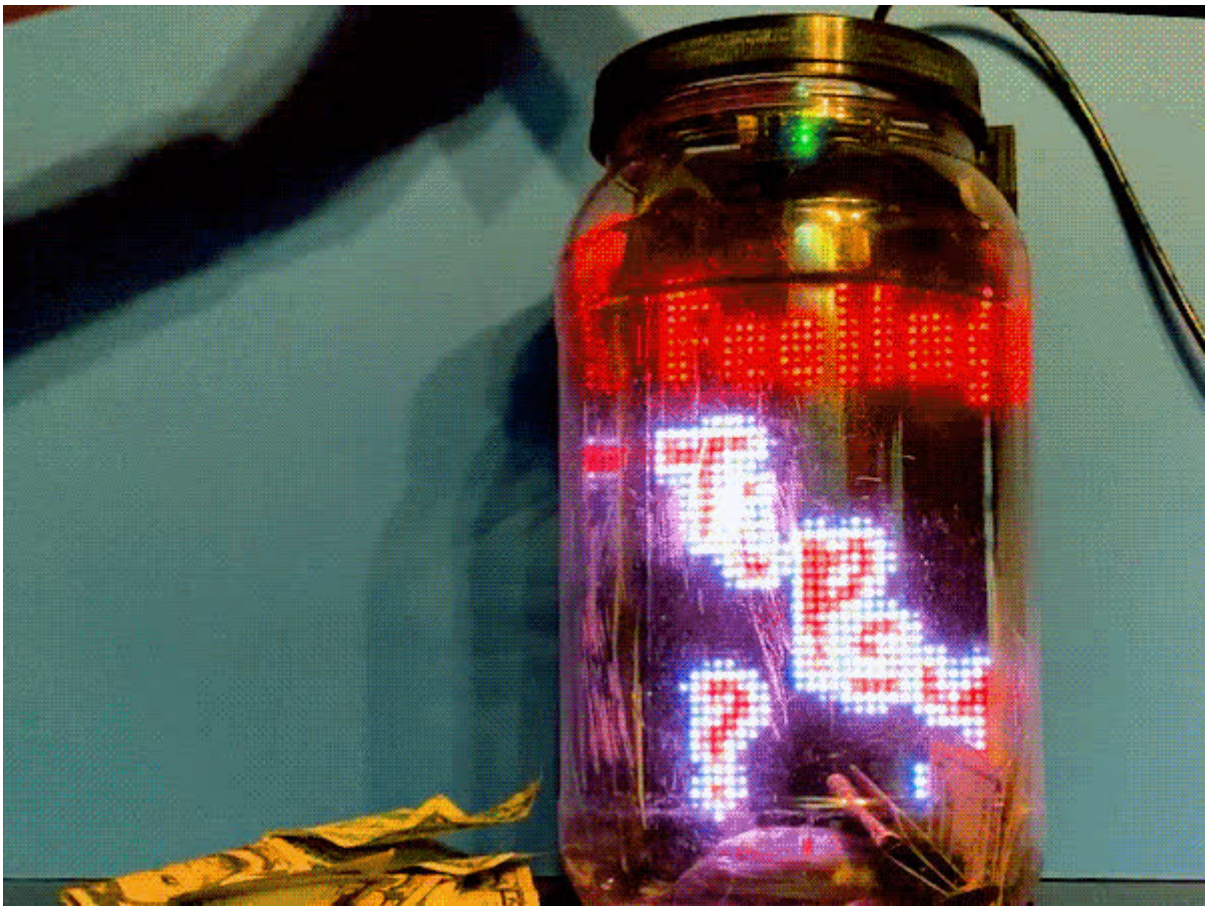
Matrix Portal Money-Sensing Tip Jar

Created by Erin St Blaine



https://learn.adafruit.com/matrix-portal-money-sensing-tip-jar

Last updated on 2025-02-24 06:27:02 PM EST

Table of Contents

Overview

Say "Thank You" in a fun and interactive way with this money-sensing tip jar. Whenever someone drops in a tip, the LED matrix at the back of the jar will reward them with a fun animation and some heartfelt gratitude.

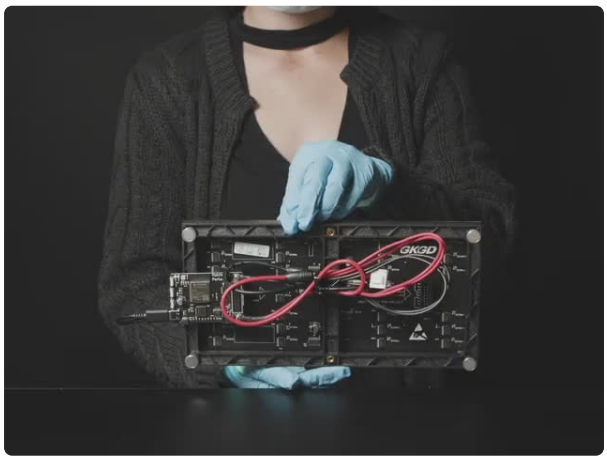
This is a fairly easy build, using an upcycled snack jar to collect your bountiful tips and powered by a Matrix Portal M4 board plugged into the RGB matrix of your choice. Use our premade images or upload your own animations to make a custom "thank you" message for your most generous customers.

Difficulty Level

This is a beginner project that doesn't require any soldering - you just need a screwdriver, some cutting tools, and some strong tape. I've also included an option that uses soldered headers for a slightly more secure build.

Materials Needed

A plastic jar with a lid, large enough to hold ALL the tips.

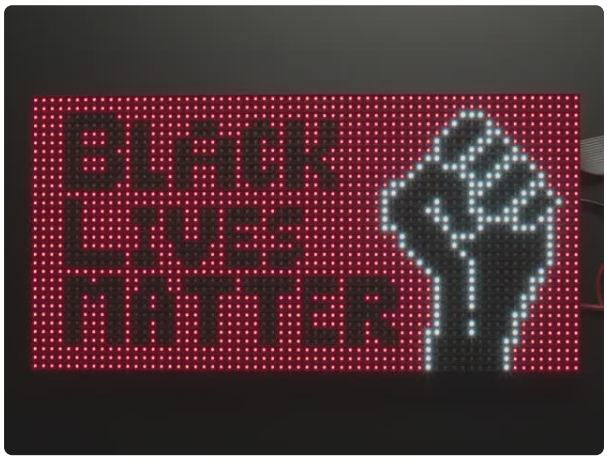


Adafruit Matrix Portal - CircuitPython Powered Internet Display
Folks love our wide selection of RGB matrices and accessories, for making custom colorful LED displays... and our RGB Matrix Shields...
https://www.adafruit.com/product/4745

1 x USB C Cable https://www.adafruit.com/product/4199
USB C to USB C Cable - USB 3.1 Gen 4 with E-Mark - 1 meter long

---

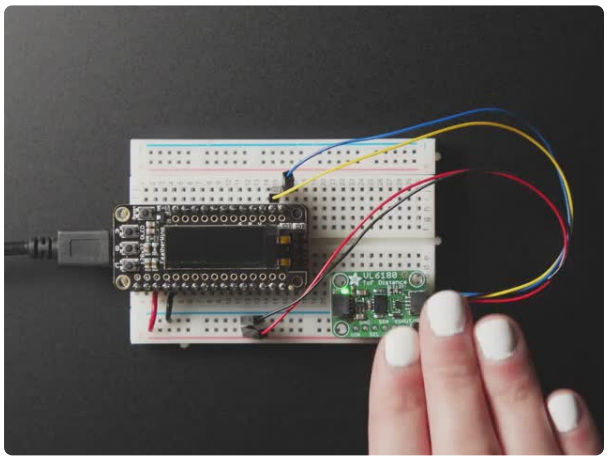
Choose the RGB Matrix size that best fits your jar. You can see all the different sizes here (https://adafru.it/NAX). I used the 4mm pitch 64x32 size.



64x32 RGB LED Matrix - 4mm pitch
Bring a little bit of Times Square into your home with this sweet 64 x 32 square RGB LED matrix panel. These panels are normally used to make video walls, here in New York we see them...
https://www.adafruit.com/product/2278

You'll also need a Time of Flight sensor. This is a very cool little sensor that uses a laser to detect proximity, and works within a really tiny range of a few millimeters.

Be sure to get one of the suggested cables to attach it to your Matrix Portal as well. Check out the "Add the Sensor" page for more details.



Adafruit VL6180X Time of Flight Distance Ranging Sensor (VL6180)
The VL6180X (sometimes called the VL6180) is a Time of Flight distance sensor like no other you've used! The sensor contains a very tiny laser...
https://www.adafruit.com/product/3316

1 x STEMMA Cable — https://www.adafruit.com/product/4210
STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long

Or:

1 x STEMMA to Male Header Cable — https://www.adafruit.com/product/4209
STEMMA QT / Qwiic JST SH 4-pin to Premium Male Headers Cable - 150mm Long

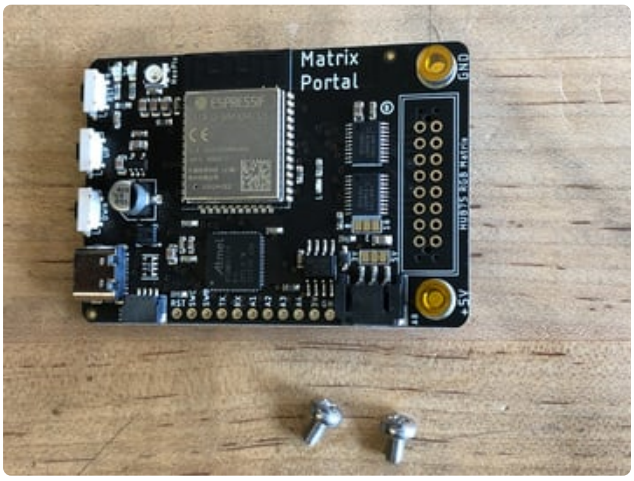
---

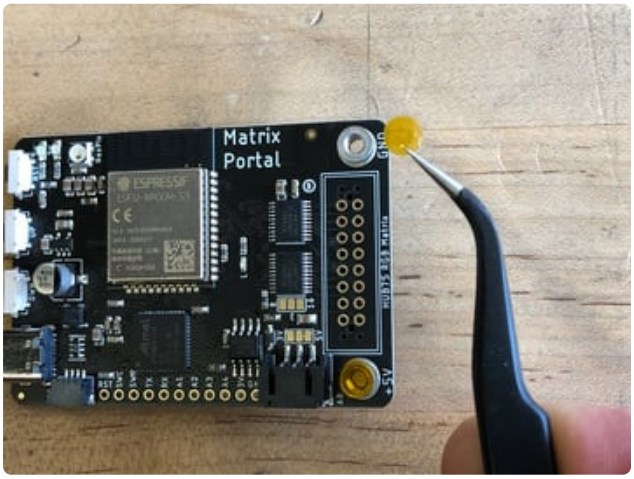
1 x Female Headers — https://www.adafruit.com/product/2886
Header Kit for Feather - 12-pin and 16-pin Female Header Set

Tools Needed

- Goo-gone or other solvent to remove any labels
- Box knife, rotary tool or saw to cut the plastic jar
- Rotary tool or heat tool to cut the lid
- Soldering iron if you're using the header-attachment method for the sensor (not needed if you're using the STEMMA plug-in cable)
- Gaffer's tape or other strong tape
- LED diffusion panel (optional) (http://adafru.it/4749)

Prep the MatrixPortal

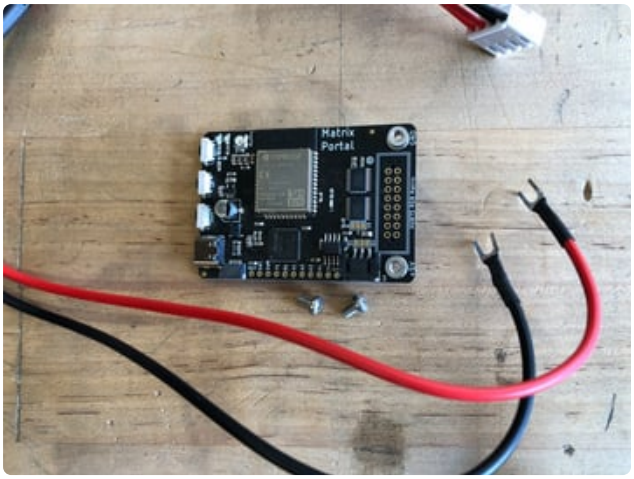


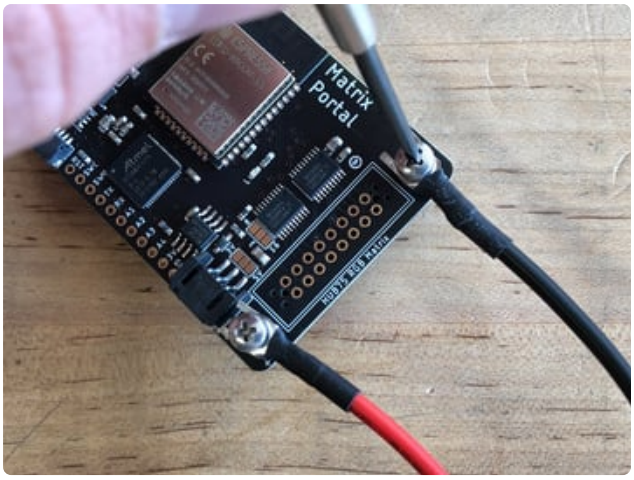


Power Prep

The MatrixPortal supplies power to the LED matrix display panel via two standoffs. These come with protective tape applied (part of the Adafruit manufacturing process) which MUST BE REMOVED!

Use some tweezers or a fingernail to remove the two amber circles.

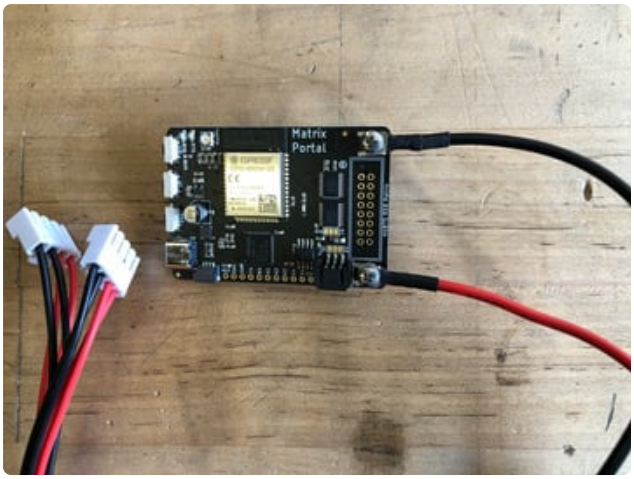


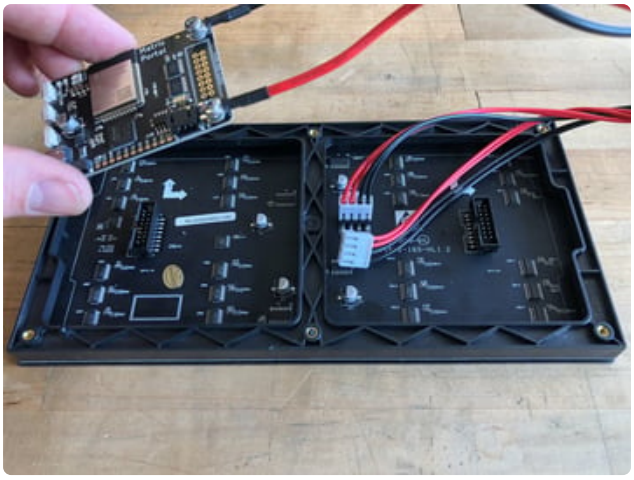


Power Terminals

Next, screw in the spade connectors to the corresponding standoff.

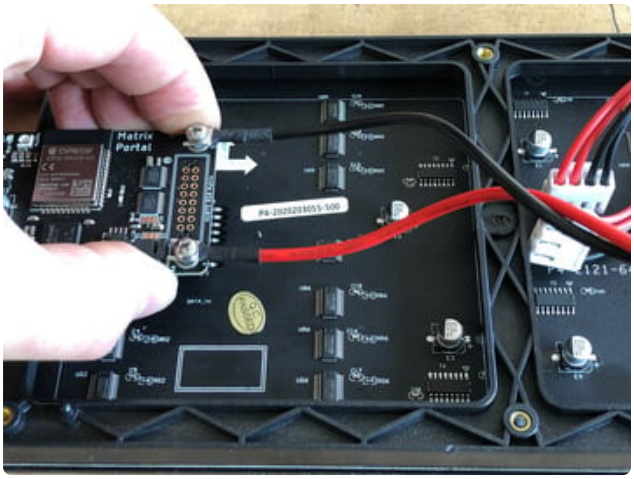
red wire goes to **+5V**
black wire goes to **GND**

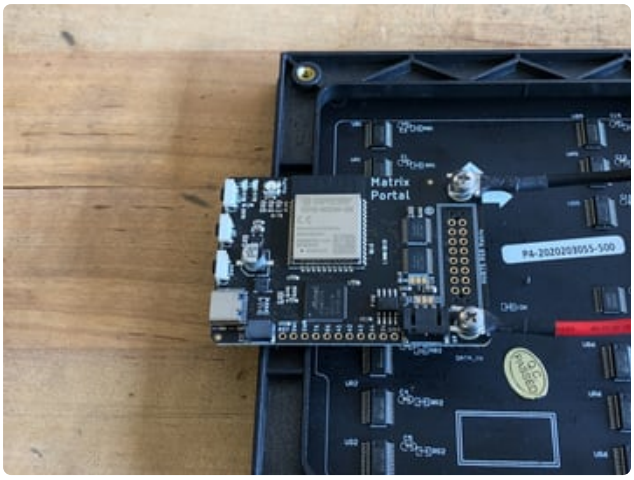




Panel Power

Plug either one of the four-conductor power plugs into the power connector pins on the panel. The plug can only go in one way, and that way is marked on the board silkscreen.

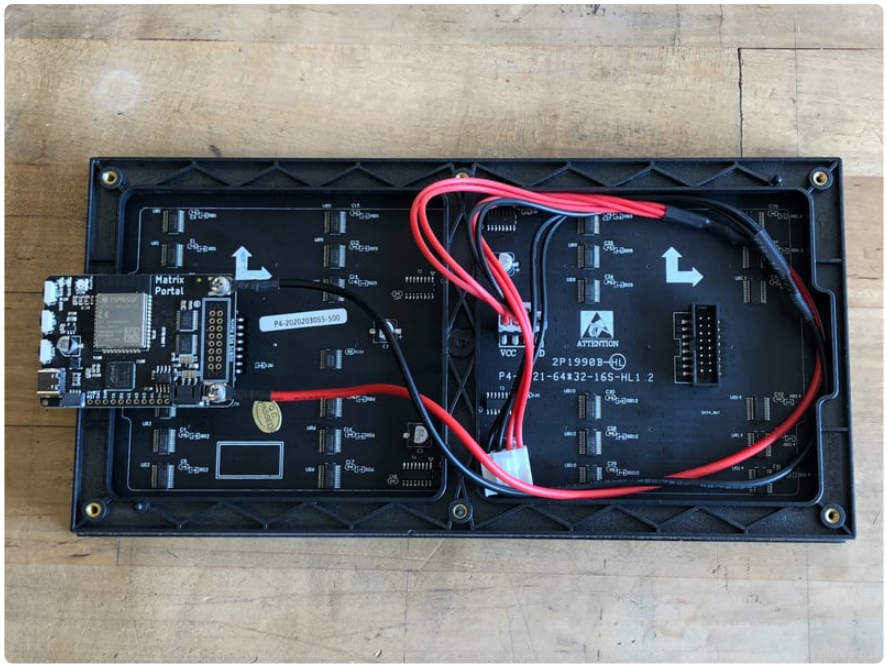




Board Connection

Now, plug the board into the left side shrouded 8x2 connector as shown. The orientation matters, so take a moment to confirm that the white indicator arrow on the matrix panel is oriented pointing up and right as seen here and the MatrixPortal overhangs the edge of the panel when connected. This allows you to use the edge buttons from the front side.

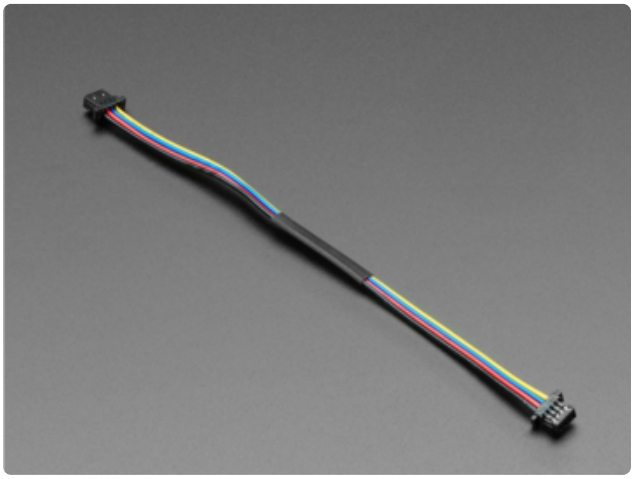
Check nothing is impeding the board from plugging in firmly. If there's a plastic nub on the matrix that's keeping the Portal from sitting flat, cut it off with diagonal cutters



For info on adding LED diffusion acrylic, see the page LED Matrix Diffuser.

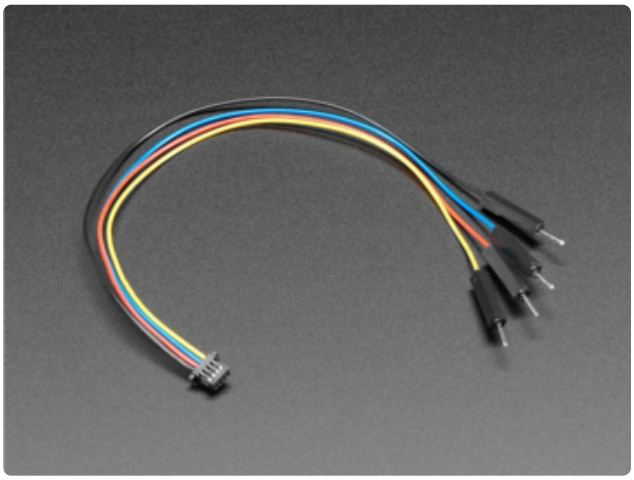
Add the Sensor

Easy Plug and Play with Stemma QT

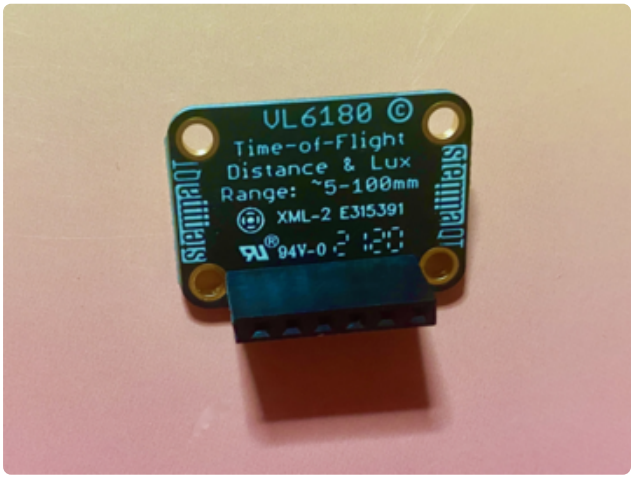


The simplest way to attach the sensor is to use this STEMMA cable. No soldering required! Just plug it right in to the port on either side.

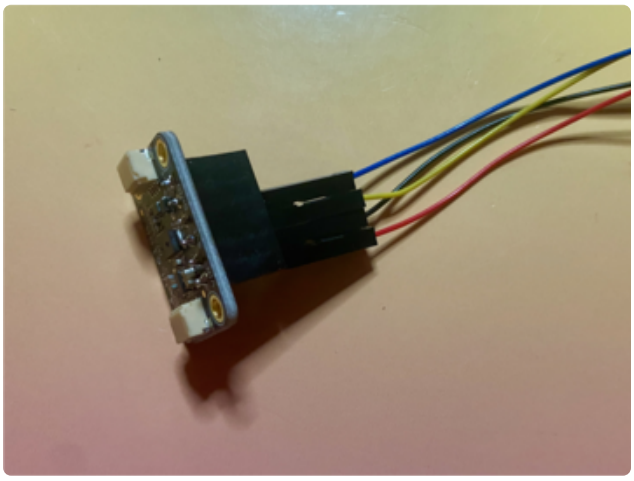
Soldering Non-QT Boards



I didn't have a double-ended STEMMA cable on hand during the build, so I used a slightly different method using this STEMMA-to-male-header pin cable. The benefit of this was that the female header on the sensor creates a really stable way to attach the sensor in a perpendicular orientation by the money slit.



Solder your female header onto the bottom side of the sensor. Plug in the four cables into the first four header ports as shown: red to VIN, black to G, yellow to SCL, and blue to SDA.



If you're new to headers, you can check out this guide for some tips. (https://adafru.it/OJb)

Plug the other end into your MatrixPortal and you're ready to go.

Install CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** drive to iterate.

Set up CircuitPython Quick Start!

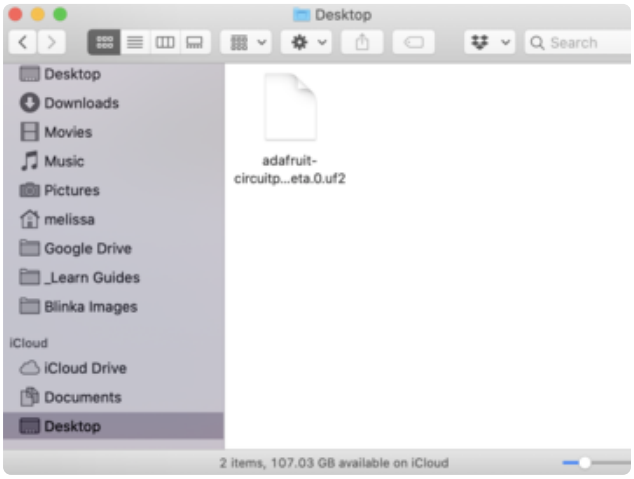
Follow this quick step-by-step for super-fast Python power :)

Download the latest version of CircuitPython for this board via circuitpython.org

https://adafru.it/Nte

Further Information

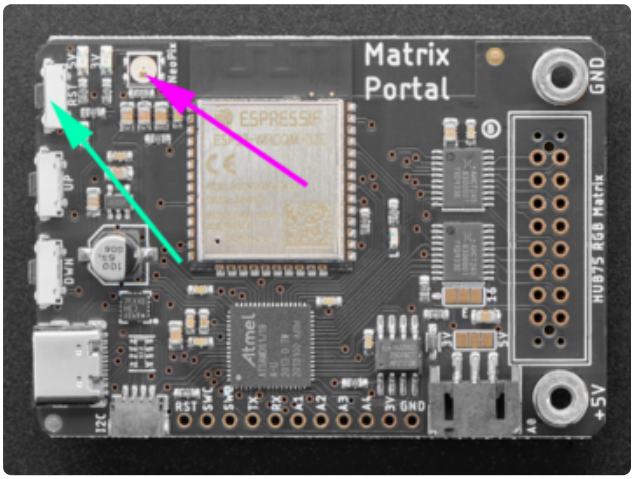
For more detailed info on installing CircuitPython, check out Installing CircuitPython (https://adafru.it/Amd).



Click the link above and download the latest UF2 file.

Download and save it to your desktop (or wherever is handy).

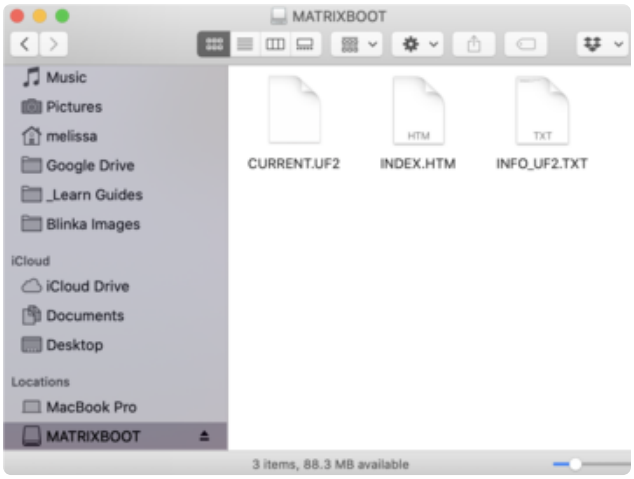
Plug your MatrixPortal M4 into your computer using a known-good USB cable.



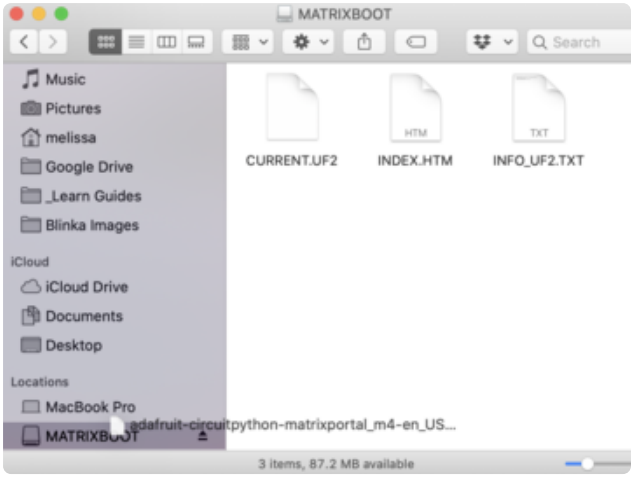
A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

Double-click the **Reset** button (indicated by the green arrow) on your board, and you will see the NeoPixel RGB LED (indicated by the magenta arrow) turn green. If it turns red, check the USB cable, try another USB port, etc.

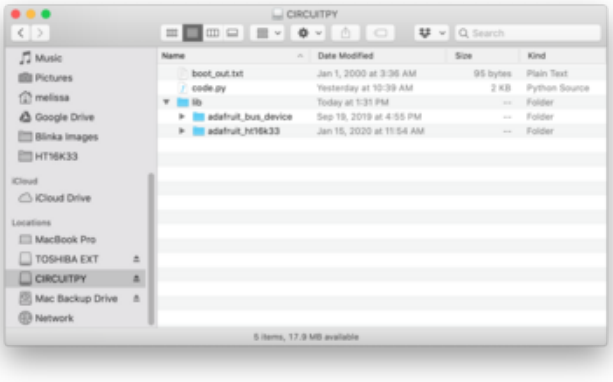
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!



You will see a new disk drive appear called **MATRIXBOOT**.



Drag the **adafruit_circuitpython_etc.uf2** file to **MATRIXBOOT.**



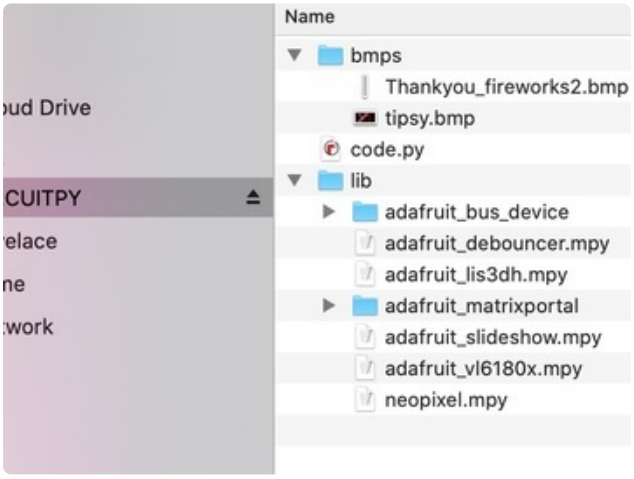
The LED will flash. Then, the **MATRIXBOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

Code with CircuitPython

This sample code will display .bmp image that encourages folks to leave you a tip. Whenever someone drops a coin or a bill through the slot, a "Thank You" animation with fireworks will play.

You can create and upload your own images, and also customize the number of times the animation plays before resetting.



Libraries

We'll need to make sure we have these libraries installed. (Check out this link (https://adafru.it/ABU) on installing libraries if needed.)

adafruit_bus_device
adafruit_debouncer
adafruit_lis3dh
adafruit_matrixportal
adafruit_slideshow
adafruit_vl6180x

Text Editor

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in this guide (https://adafru.it/ANO).

Alternatively, you can use any text editor that saves simple text files.

Code

Click the Download: Project Zip File link below in the code window to get a zip file with all the files needed for the project. Copy **code.py** from the zip file and place on the **CIRCUITPY** drive.

You'll also need to copy the following files to the **CIRCUITPY** drive. See the graphic at the top of the page as to filenames and where they go):

- **/bmps** directory, which contains the graphics .bmp files.

The button below contains our sample graphics: a "Feeling Tipsy?" .bmp file, and a "Thank You" fireworks animation sprite sheet.

bmps.zip

https://adafru.it/OJB

```
# SPDX-FileCopyrightText: 2020 Melissa LeBlanc-Williams for Adafruit Industries
# SPDX-FileCopyrightText: 2020 Erin St Blaine for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
Motion-sensing Animation Example using the Matrix Portal and 64 x 32 LED matrix
display
```

```
Written by Melissa LeBlanc-Williams and Erin St Blaine for Adafruit Industries
A VL6180X sensor causes a sprite sheet animation to play
"""

import time
import os
import board
import busio
import displayio
from digitalio import DigitalInOut, Pull
from adafruit_matrixportal.matrix import Matrix
from adafruit_debouncer import Debouncer
import adafruit_vl6180x
# pylint: disable=global-statement
# Create I2C bus.
i2c = busio.I2C(board.SCL, board.SDA)

# Create sensor instance.
sensor = adafruit_vl6180x.VL6180X(i2c)

SPRITESHEET_FOLDER = "/bmps"
DEFAULT_FRAME_DURATION = 0.1  # 100ms
ANIMATION_DURATION = 5
AUTO_ADVANCE_LOOPS = 1
THRESHOLD = 20

# --- Display setup ---
matrix = Matrix(bit_depth=4)
sprite_group = displayio.Group()
matrix.display.root_group = sprite_group

# --- Button setup ---
pin_down = DigitalInOut(board.BUTTON_DOWN)
pin_down.switch_to_input(pull=Pull.UP)
button_down = Debouncer(pin_down)
pin_up = DigitalInOut(board.BUTTON_UP)
pin_up.switch_to_input(pull=Pull.UP)
button_up = Debouncer(pin_up)

AUTO_ADVANCE = True

file_list = sorted(
    [
        f
        for f in os.listdir(SPRITESHEET_FOLDER)
        if (f.endswith(".bmp") and not f.startswith("."))
    ]
)

if len(file_list) == 0:
    raise RuntimeError("No images found")

CURRENT_IMAGE = None
CURRENT_FRAME = 0
CURRENT_LOOP = 0
FRAME_COUNT = 0
FRAME_DURATION = DEFAULT_FRAME_DURATION


def load_image():
    """
    Load an image as a sprite
    """
    # pylint: disable=global-statement
    global CURRENT_FRAME, CURRENT_LOOP, FRAME_COUNT, FRAME_DURATION
    while sprite_group:
        sprite_group.pop()

    filename = SPRITESHEET_FOLDER + "/" + file_list[CURRENT_IMAGE]
```

```
    # CircuitPython 6 & 7 compatible
    bitmap = displayio.OnDiskBitmap(open(filename, "rb"))
    sprite = displayio.TileGrid(
        bitmap,
        pixel_shader=getattr(bitmap, 'pixel_shader', displayio.ColorConverter()),
        tile_width=bitmap.width,
        tile_height=matrix.display.height,
    )

    # # CircuitPython 7+ compatible
    # bitmap = displayio.OnDiskBitmap(filename)
    # sprite = displayio.TileGrid(
    #     bitmap,
    #     pixel_shader=bitmap.pixel_shader,
    #     tile_width=bitmap.width,
    #     tile_height=matrix.display.height,
    # )

    sprite_group.append(sprite)

    FRAME_COUNT = int(bitmap.height / matrix.display.height)
    FRAME_DURATION = DEFAULT_FRAME_DURATION
    CURRENT_FRAME = 0
    CURRENT_LOOP = 0


def advance_image():
    """
    Advance to the next image in the list and loop back at the end
    """
    # pylint: disable=global-statement
    global CURRENT_IMAGE
    if CURRENT_IMAGE is not None:
        CURRENT_IMAGE += 1
    if CURRENT_IMAGE is None or CURRENT_IMAGE >= len(file_list):
        CURRENT_IMAGE = 0
    load_image()

def advance_frame():
    """Advance the frame"""
    # pylint: disable=global-statement
    global CURRENT_FRAME, CURRENT_LOOP
    CURRENT_FRAME = CURRENT_FRAME + 1
    if CURRENT_FRAME >= FRAME_COUNT:
        CURRENT_FRAME = 0
        CURRENT_LOOP = CURRENT_LOOP + 1
    sprite_group[0][0] = CURRENT_FRAME

def load_list_image(item):
    """Load the list item"""
    global CURRENT_IMAGE
    CURRENT_IMAGE = item
    load_image()

def load_tipsy():
    """Load the .bmp image"""
    load_list_image(1)

def play_thankyou():
    """load the thank you image"""
    load_list_image(0)
    while CURRENT_LOOP <= AUTO_ADVANCE_LOOPS:
        advance_frame()
        time.sleep(FRAME_DURATION)


advance_image()
```

```
while True:
    if sensor.range < THRESHOLD:
        play_thankyou()
    else:
        load_tipsy()
```

Pixel Art Specs

If you want to create your own artwork for display, these are the specifications to follow.

For the still .bmp image:

- Images should be a maximum of 64 pixels high
- Images can be up to 32 pixels wide
- Colors are 16-bit or 24-bit RGB
- Save files as .bmp format
- Since the orientation of the tip jar is vertical rather than horizontal, rotate your image 90% clockwise before uploading it.

We've found that crisp images (not too much anti-aliasing) work best.

We have a whole page on Pixel Art Fundamentals (https://adafru.it/EI8) here!

For the Sprite Sheet:

A Sprite Sheet is like a filmstrip: a vertical series of animation frames saved as a single .bmp. Our code will parse this into animation frames and play them back at the speed you specify.

Check out this guide for detailed instructions on building your own a sprite sheet with Aseprite (https://adafru.it/NTB).

How it Works

Libraries

Here's how the code works. First we import the libraries:

```
import time
import os
import board
import busio
import displayio
from digitalio import DigitalInOut, Pull
from adafruit_matrixportal.matrix import Matrix
from adafruit_debouncer import Debouncer
import adafruit_vl6180x
```

Next, we set up the sensor:

```
# Create I2C bus.
i2c = busio.I2C(board.SCL, board.SDA)

# Create sensor instance.
sensor = adafruit_vl6180x.VL6180X(i2c)
```

Variables

Then, we'll set the variables for `DEFAULT_FRAME_DURATION` (how quickly we move through the sprite sheet) and `ANIMATION_DURATION` (how long each animation stays on the screen). `AUTO_ADVANCE_LOOPS` determines how many times your sprite sheet plays through. And we'll set up the path to the image folders.

The `THRESHOLD` variable is the distance in mm between your sensor and the edge of the money slot. If your tip jar isn't reacting, try increasing this number a bit. Or, open your serial monitor in the Mu editor for a readout from the sensor to see exactly what readings you're getting.

```
SPRITESHEET_FOLDER = "/bmps"
DEFAULT_FRAME_DURATION = 0.1  # 100ms
ANIMATION_DURATION = 5
AUTO_ADVANCE_LOOPS = 1
THRESHOLD = 20
```

Display/Pin Setup

Next, we set up the display and the pins used for the buttons. We aren't using the buttons for anything in this code sample, but they are set up for you in case you want to add button functionality.

```
# --- Display setup ---
matrix = Matrix(bit_depth=4)
sprite_group = displayio.Group()
matrix.display.root_group = sprite_group

# --- Button setup ---
pin_down = DigitalInOut(board.BUTTON_DOWN)
pin_down.switch_to_input(pull=Pull.UP)
button_down = Debouncer(pin_down)
pin_up = DigitalInOut(board.BUTTON_UP)
pin_up.switch_to_input(pull=Pull.UP)
button_up = Debouncer(pin_up)
```

Functions

Next we'll set up our functions. These will govern the loading and playback of the images.

```
def load_image():
    """
    Load an image as a sprite
    """
    # pylint: disable=global-statement
```

```
    global CURRENT_FRAME, CURRENT_LOOP, FRAME_COUNT, FRAME_DURATION
    while sprite_group:
        sprite_group.pop()

    filename = SPRITESHEET_FOLDER + "/" + file_list[CURRENT_IMAGE]

    # CircuitPython 6 &amp; 7 compatible
    bitmap = displayio.OnDiskBitmap(open(filename, "rb"))
    sprite = displayio.TileGrid(
        bitmap,
        pixel_shader=getattr(bitmap, 'pixel_shader', displayio.ColorConverter()),
        tile_width=bitmap.width,
        tile_height=matrix.display.height,
    )

    # # CircuitPython 7+ compatible
    # bitmap = displayio.OnDiskBitmap(filename)
    # sprite = displayio.TileGrid(
    #     bitmap,
    #     pixel_shader=bitmap.pixel_shader,
    #     tile_width=bitmap.width,
    #     tile_height=matrix.display.height,
    # )

    sprite_group.append(sprite)

    FRAME_COUNT = int(bitmap.height / matrix.display.height)
    FRAME_DURATION = DEFAULT_FRAME_DURATION
    CURRENT_FRAME = 0
    CURRENT_LOOP = 0

def advance_image():
    """
    Advance to the next image in the list and loop back at the end
    """
    # pylint: disable=global-statement
    global CURRENT_IMAGE
    if CURRENT_IMAGE is not None:
        CURRENT_IMAGE += 1
    if CURRENT_IMAGE is None or CURRENT_IMAGE &gt;= len(file_list):
        CURRENT_IMAGE = 0
    load_image()

def advance_frame():
    """Advance the frame"""
    # pylint: disable=global-statement
    global CURRENT_FRAME, CURRENT_LOOP
    CURRENT_FRAME = CURRENT_FRAME + 1
    if CURRENT_FRAME &gt;= FRAME_COUNT:
        CURRENT_FRAME = 0
        CURRENT_LOOP = CURRENT_LOOP + 1
    sprite_group[0][0] = CURRENT_FRAME

def load_list_image(item):
    """Load the list item"""
    global CURRENT_IMAGE
    CURRENT_IMAGE = item
    load_image()

def load_tipsy():
    """Load the .bmp image"""
    load_list_image(1)

def play_thankyou():
    """load the thank you image"""
    load_list_image(0)
    while CURRENT_LOOP &lt;= AUTO_ADVANCE_LOOPS:
```

```
        advance_frame()
        time.sleep(FRAME_DURATION)
```

Main Loop

Since we did most of the heavy lifting in the functions, our main loop is really simple. Check the sensor and see if the `THRESHOLD` value has been triggered (if someone puts a dollar into the jar). If so, the Thank You animation will play `AUTO_ADVANCE_LOOPS` times through, then it will reset to playing the tipsy image.

```
while True:
    if sensor.range &lt; THRESHOLD:
        play_thankyou()
    else:
        load_tipsy()
```

Uploading & Testing

Save your code to the **CIRCUITPY** drive on your computer as **code.py**, at the root of the **CIRCUITPY** drive. The code will start to run automatically.

If you see a "blinka" error message, here are some things to try:

- Have you installed all the libraries?
- Are you running the latest version of CircuitPython?
- Are your .bmp images uploaded to a folder called /bmps?

If it's still not working, check out this guide for more troubleshooting tips (https://adafru.it/NDR).

Build the Tip Jar

Remove any labels or stickers from your jar. Goo-Gone works well for this. Put the jar on top of the RGB matrix and draw vertical cut lines with a permanent marker along the edges of the matrix.

Be extremely careful using any cutting tool and wear appropriate clothing and take necessary precautions.

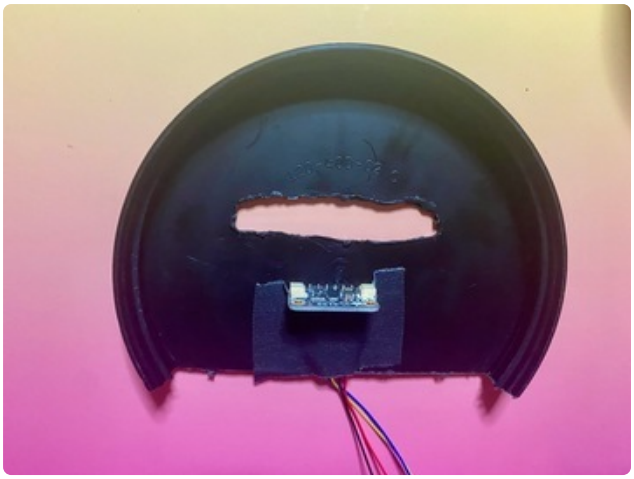
Use a box cutter or rotary tool to carefully cut along the cut lines, creating an open "flat" area to attach your RGB matrix.

Clean up the edges with flush cutters or a grinder, if needed. Try to get them as straight as possible. The jar should lay perfectly flat on the table without wiggling, if you put the cut side down.

Cover the raw edges with a piece of 1" gaffer's tape.

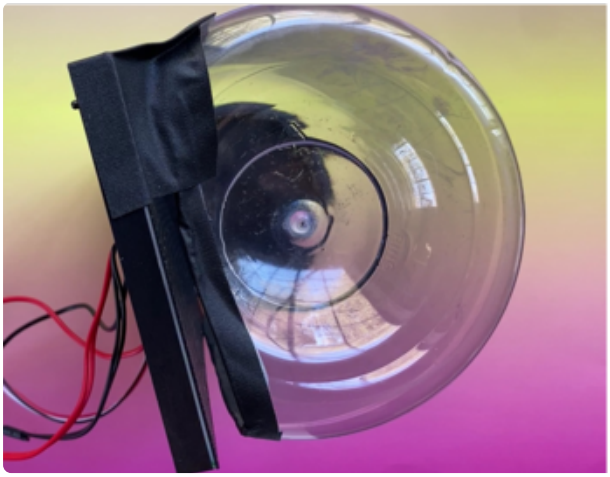
If your jar is the same height or shorter than your RGB matrix, you'll need to modify the lid as well. Put the lid on the jar and mark your cut line, then cut the lid to size with a saw or rotary tool.

Use a rotary tool or heat tool to make an opening to drop the money in.



Attach your sensor to the underside of the lid with the laser pointing toward the opening. Place it within 20mm of the opening if possible - that way you won't need to modify the sample code.

If your build requires a different alignment, you can modify the code without too much trouble.



Attach the RGB matrix to the back of the jar with the lights pointing inwards. Be sure to line it up evenly on the bottom so the jar doesn't tip over - it is a little bit back-heavy when it's not full of money. (So let's get on with it, and get those quarters rolling in!)

If you've already uploaded the code, test it out by dropping some coins or bills through the slot. Each time a tip is dropped in, you will see a Thank You animation on the pixels.

Powering

Power the project via the USB port on the Matrix Portal. You can plug directly into a power supply (like this one (http://adafru.it/4298)), or plug in a USB battery (http://adafru.it/1959) if you're busking outdoors and need a mobile tip jar.