RGB Matrix Slot Machine
Created by Ruiz Brothers

https://learn.adafruit.com/rgb-matrix-slot-machine

Last updated on 2023-11-13 11:47:57 AM EST
# Table of Contents

## Overview
- Fruit Machine
- Parts
- Black LED Acrylic

## Circuit Diagram
- Adafruit Library for Fritzing
- Wired Connections
- Powering

## 3D Printing
- Parts List
- Install Ball to Handle
- Spring–Loaded Handle
- Support Material for Frame

## CircuitPython on Feather M4 Express
- Set up CircuitPython Quick Start!
- Further Information

## CircuitPython Libraries
- Installing or upgrading CircuitPython
- Download the Adafruit CircuitPython Library Bundle
- Required Libraries
- Required Assets
- Upload Code
- Double Check

## CircuitPython Code Walkthrough
- Setup
- Main loop

## FeatherWing Setup
- Install Feather Headers
- RGB Matrix FeatherWing Headers
- Install 2x10 Socket Header
- Install Power Connectors
- Install Female Headers on RGB Matrix FeatherWing
- Solder Female Headers
- RGB Matrix FeatherWing Polarity Check
- Install RGB Matrix FeatherWing
- Install Feather M4 Express

## Wiring
- Power Cable
- Trim Power Cable
- Slice Power Cable
- Shortened Power Cable
- Wire PAM8302 Audio Amp
- Wire Speaker
- Wire Button Switch
• Solder Wires to RGB Matrix FeatherWing
• Wired RGB Matrix FeatherWing

Assembly

• Connect Power Cable
• Circuit Wiring Review
• Secure Amp to Mount
• Secure Speaker to Mount
• Install Frame to Matrix
• Frame Slot
• Install Acrylic to Frame
• Install Feet
• Secure Feet
• Installed Feet
• Install FeatherWing and Mount
• Secure Mount to RGB Matrix
• Connect Speaker to Amp
• Plug In Power Cable to RGB Matrix
• Wire Management
• Secure Handle to Frame
• Installed Handle
• Install Button to Handle
• Test Button Handle Actuation
Overview

Fruit Machine

This features emojis from Circuit Playground and works just like a fruit style slot machine. The handle is 3D printed and features a print-in-place spring loaded mechanism. It actuates a small push button and triggers animations that look a lot like a slot machine.

The display is powered by an Adafruit Feather and the RGB Matrix FeatherWing. It also has a speaker and amp for playing MP3s. The FeatherWing is designed to be modular and snap fits on the back of the RGB matrix.

Using CircuitPython you can quickly show text, bitmaps, and animations on a number of different RGB Matrices. It's designed to work with 64x32 RGB matrices.
Parts

List of parts used to build this project.

32x64 RGB Matrix (4mm pitch) ()
RGB Matrix FeatherWing ()
Feather M4 Express ()
PAM8302 Amp ()
3W Speaker ()
5V 4A Power Supply ()
Slim button switch ()
10-wire silicone ribbon cable ()

Black LED Acrylic
A sheet of black LED acrylic used to diffuse the RGB LED Matrix. From their website:

It looks like a matte black acrylic when it is not lit, but add a light behind it and watch it transform. This fantastic material has the ability to transmit light of any color.

Black LED acrylic is used to diffuse the LEDs that are otherwise harsh and over exposed. This material features a matte finish and has the ability to transmit any color of light. That makes it perfect for bringing out the vibrance in the colors of the LEDs.

The 3D printed frame is designed with a slot on the side so it’s easy to insert a sheet of Black LED acrylic.

Recommended Cut To Size Sheet: 257mm x 128mm (10.12in x 5.04in)

Black LED Acrylic from TAP Plastics
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

The 3D printed parts in this project are designed for this specific RGB LED Matrix (PID 2278)

Adafruit RGB Matrix Featherwing Kit
Ahoy! It's time to create a dazzling light up project with our new RGB Matrix FeatherWing. Now you can quickly and easily create...
https://www.adafruit.com/product/3036

Adafruit Feather M4 Express - Featuring ATSAMD51
It's what you've been waiting for, the Feather M4 Express featuring ATSAMD51. This Feather is fast like a swift, smart like an owl, strong like a ox-bird (it's half ox,....
https://www.adafruit.com/product/3857
Tactile Switch Buttons (6mm slim) x 20 pack
Slim clicky momentary switches are standard input "buttons" on electronic projects. These are half the width of classic 6mm tactile switches so they line up better on a...
https://www.adafruit.com/product/1489

Silicone Cover Stranded-Core Ribbon Cable - 10 Wire 1 Meter Long
For those who are fans of our silicone-covered wires, but are always looking to up their wiring game. We now have Silicone Cover Ribbon cables! These may look...
https://www.adafruit.com/product/3890

Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302
This super small mono amplifier is surprisingly powerful - able to deliver up to 2.5 Watts into 4-8 ohm impedance speakers. Inside the miniature chip is a class D controller, able to...
https://www.adafruit.com/product/2130

Speaker - 40mm Diameter - 4 Ohm 3 Watt
Hear the good news! This speaker is a great addition to any audio project where you need a 4 Ohm impedance and 3W or less of power. At 40mm diameter it...
https://www.adafruit.com/product/3968
5V 4A (4000mA) switching power supply - UL Listed
Need a lot of 5V power? This switching supply gives a clean regulated 5V output at up to 4 Amps (4000mA). 110 or 240 input, so it works in any country. The plugs are "US...
https://www.adafruit.com/product/1466

Fully Reversible Pink/Purple USB A to micro B Cable - 1m long
This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also fully reversible! That's right, you will save seconds a day by...
https://www.adafruit.com/product/4111

Circuit Diagram
The diagram below provides a visual reference for wiring of the components. This diagram was created using the software package Fritzing ().
Adafruit Library for Fritzing

Use Adafruit's Fritzing parts library to create circuit diagrams for your projects. Download the library or just grab individual parts. Get the library and parts from GitHub - Adafruit Fritzing Parts.

Wired Connections

- A+ from PAM8302 to A0 on RGB Matrix FeatherWing
- A- from PAM8302 to GND on PAM8302
- Vin from PAM8302 to 3V on RGB Matrix FeatherWing
- GND from PAM8302 to GND on RGB Matrix FeatherWing

Button Switch

- Switch to GND on RGB Matrix FeatherWing
- Switch to A1 on RGB Matrix FeatherWing

Speaker

- Red to (+)Positive on PAM8302
- Black to (–)Negative on PAM8302
Powering

The RGB Matrix will need to be powered by a [5V 4A wall adapter](https://www.adafruit.com/product/1466). 5V USB cable is not enough to fully power the display – Glitches and artifacts are displayed when RGB matrix is only powered by USB cable.

![5V 4A (4000mA) switching power supply - UL Listed](https://www.adafruit.com/images/pimoroni/adafruit_power_supply.jpg)

Need a lot of 5V power? This switching supply gives a clean regulated 5V output at up to 4 Amps (4000mA). 110 or 240 input, so it works in any country. The plugs are "US... https://www.adafruit.com/product/1466

3D Printing

Parts List

STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below.

matrix-handle.stl
matrix-frame.stl
speaker-holder.stl
2x handle-ball.stl
foot-left.stl
foot-right.stl

Download the 3D printable models from Thingiverse

The 3D printed parts in this project are designed for this specific RGB LED Matrix (PID 2278)
Install Ball to Handle
Two hemispheres (handle-ball.stl) are joined together to form the ball of the handle. They're are press fitted over the tip of the spring-loaded handle. Domes features recesses to accommodate for the handle. Optionally use glue to bond the domes to the handle.

Spring–Loaded Handle
The handle features print-in-place parts that require good bed leveling and sufficient active cooling. Overhanging geometry is supported by surrounding features. No supports are required. The handle and gears are printed with loose tolerances to avoid fusing.

Support Material for Frame
The frame requires support material. Enable supports in the slicing software to generate scaffolding to support overhanging geometry. Use the following recommended settings:

- Support Placement: Everywhere
- Support Overhang Angle: 80
- Support Pattern: ZigZag
- Support Density: 4%
- Enable Support Brim: No
- Support Z Distance: 0.22mm
- Enable Support Interface: No

Spring-Loaded handle design was inspired by Turbo Sunshine https://www.thingiverse.com/thing:4382544

The frame requires a minimum bed size of 265mm x 135mm.
CircuitPython on Feather M4 Express

CircuitPython is a derivative of MicroPython 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.

The following instructions will show you how to install CircuitPython. If you've already installed CircuitPython but are looking to update it or reinstall it, the same steps work for that as well!

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

Click the link above and download the latest UF2 file.

Download and save it to your desktop (or wherever is handy).
Plug your Feather 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 next to the USB connector on your board, and you will see the NeoPixel RGB LED turn green. If it turns red, check the USB cable, try another USB port, etc. Note: The little red LED next to the USB connector will pulse red. That's ok!

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 FEATHERBOOT.

Drag the adafruit_circuitpython_etc.uf2 file to FEATHERBOOT.
The LED will flash. Then, the FEATHERBOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

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

Further Information

For more detailed info on installing CircuitPython, check out Installing CircuitPython.

CircuitPython Libraries

Installing or upgrading CircuitPython

You should ensure you have CircuitPython 5.3 or greater on your board. Plug your board in with a known good data + power cable (not the cheesy USB cable that comes with USB power packs, they are power only). You should see a new flash drive pop up.

If the drive is CIRCUITPY, then open the boot_out.txt file to ensure the version number is 5.3 or greater.

Adafruit CircuitPython 5.3.0 on 2020-04-29; Adafruit Feather M4 Express with samd51j19

If the version is less than 5 -or- you only get a drive named FEATHERBOOT then follow the Feather M4 guide on installing CircuitPython.

Download the Adafruit CircuitPython Library Bundle

In order to run the code, we'll need to download a few libraries. Libraries contain code to help interface with hardware a lot easier for us.

Use the Feather M4 page on Installing Libraries to get the library that matches the major version of CircuitPython you are using noted above.
The green button below links to a file containing all the libraries available for CircuitPython. To run the code for this project, we need the large number of libraries in the Required Libraries list below. Unzip the library bundle and search for the libraries. Drag and drop the files into a folder named lib on the CIRCUITPY drive (create the folder if it is not already on the Feather M4).

### Required Libraries
- adafruit_display_text
- adafruit_bitmap_font
- adafruit_imageload

### Required Assets
- emoji.bmp
- slots.mp3

Once we have all the files we need, a directory listing will look similar to below as far as files and directories.

## Upload Code

Click on the Download: Project Zip link below to grab the project files in a zip file directly from GitHub. Place the code.py file and bitmap graphics files onto the CIRCUITPY main (root) directory. The code will run properly when all of the files have been uploaded including libraries.

Use any text editor or favorite IDE to modify the code. We suggest using Mu as noted above.

```python
# SPDX-FileCopyrightText: 2020 Jeff Epler for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import random
import time
import adafruit_imageload.bmp
import audiioio
import audiomp3
import board
import displayio
import digitalio
import framebufferio
import rgbmatrix

displayio.release_displays()

matrix = rgbmatrix.RGBMatrix...
```
# Each wheel can be in one of three states:
STOPPED, RUNNING, BRAKING = range(3)

# Return a duplicate of the input list in a random (shuffled) order.
def shuffled(seq):
    return sorted(seq, key=lambda _ : random.random())

# The Wheel class manages the state of one wheel. "pos" is a position in
# floating point coordinates, with one 1 pixel being 1 position.
# The wheel also has a velocity (in positions
# per tick) and a state (one of the above constants)
class Wheel(displayio.TileGrid):
    def __init__(self, bitmap, palette):
        # Portions of up to 3 tiles are visible.
        super().__init__(bitmap=bitmap, pixel_shader=palette,
                         width=1, height=3, tile_width=20, tile_height=24)
        self.order = shuffled(range(20))
        self.state = STOPPED
        self.pos = 0
        self.vel = 0
        self.termvel = 2
        self.y = 0
        self.x = 0
        self.stop_time = time.monotonic_ns()
        self.step()

    def step(self):
        # Update each wheel for one time step
        if self.state == RUNNING:
            # Slowly lose speed when running, but go at least terminal velocity
            self.vel = max(self.vel * .99, self.termvel)
            if time.monotonic_ns() > self.stop_time:
                self.state = BRAKING
        elif self.state == BRAKING:
            # More quickly lose speed when baking, down to speed 0.4
            self.vel = max(self.vel * .85, 0.4)
        # Advance the wheel according to the velocity, and wrap it around
        # after 24*20 positions
        self.pos = (self.pos + self.vel) % (20*24)

        # Compute the rounded Y coordinate
        yy = round(self.pos)
        # Compute the offset of the tile (tiles are 24 pixels tall)
        yyy = yy % 24
        # Find out which tile is the top tile
        off = yy // 24

        # If we're braking and a tile is close to midscreen,
        # then stop and make sure that tile is exactly centered
        if self.state == BRAKING and self.vel <= 2 and yyy < 8:
            self.vel = 0
            self.y = 0
            self.state = STOPPED

        # Move the displayed tiles to the correct height and make sure the
        # correct tiles are displayed.
        for i in range(3):
            self[i] = self.order[(19 - i + off) % 20]

        # Set the wheel running again, using a slight bit of randomness.
# The 'i' value makes sure the first wheel brakes first, the second
# brakes second, and the third brakes third.
def kick(self, i):
    self.state = RUNNING
    self.vel = random.uniform(8, 10)
    self.termvel = random.uniform(1.8, 4.2)
    self.stop_time = time.monotonic_ns() + 3000000000 + i * 350000000

# This bitmap contains the emoji we're going to use. It is assumed
# to contain 20 icons, each 20x24 pixels. This fits nicely on the 64x32
# RGB matrix display.
the_bitmap, the_palette = adafruit_imageload.load(
    "~/emoji.bmp",
    bitmap=displayio.Bitmap,
    palette=displayio.Palette)

# Our fruit machine has 3 wheels, let's create them with a correct horizontal
# (x) offset and arbitrary vertical (y) offset.
g = displayio.Group()
wheels = []
for idx in range(3):
    wheel = Wheel(the_bitmap, the_palette)
    wheel.x = idx * 22
    wheel.y = -20
    g.append(wheel)
wheels.append(wheel)
display.root_group = g

# We want a digital input to trigger the fruit machine
button = digitalio.DigitalInOut(board.A1)
button.switch_to_input(pull=digitalio.Pull.UP)

# Enable the speaker
enable = digitalio.DigitalInOut(board.D4)
enable.switch_to_output(True)

mp3file = open("~/triangles-loop.mp3", "rb")
sample = audiomp3.MP3Decoder(mp3file)

# Play the sample (just loop it for now)
speaker = audioio.AudioOut(board.A0)
speaker.play(sample, loop=True)

# Here's the main loop
while True:
    # Refresh the display (doing this manually ensures the wheels move
    # together, not at different times)
    display.refresh(minimum_frames_per_second=0, target_frames_per_second=60)
    all_stopped = all(si.state == STOPPED for si in wheels)
    if all_stopped:
        # Once everything comes to a stop, wait until the lever is pulled and
        # start everything over again. Maybe you want to check if the
        # combination is a "winner" and add a light show or something.
        while button.value:
            pass
        for idx, si in enumerate(wheels):
            si.kick(idx)

    # Otherwise, let the wheels keep spinning...
    for idx, si in enumerate(wheels):
        si.step()
Double Check

See the directory listing above and double check that you have all the files listed to make this project function. If any are missing or in an incorrect directory, move them so they’re in the right places.

CircuitPython Code Walkthrough

Setup

CircuitPython Libraries

The CircuitPython code begins by importing the libraries that are used.

```python
import random
import time
import adafruit_imageload.bmp
import audiocore
import audiomp3
import board
import displayio
import digitalio
import framebufferio
import rgbmatrix
```

Display setup

First, any display left over from a previous code.py is released. Then, the LED matrix is initialized according to its pinout. Because we want to ensure that all 3 wheels on screen are updated at the same time, we specify `auto_refresh=False`. To learn more about how to write code using the rgbmatrix module, see the dedicated guide.

```python
displayio.release_displays()
matrix = rgbmatrix.RGBMatrix(
    width=64, height=32, bit_depth=3,
    addr_pins=[board.A5, board.A4, board.A3, board.A2],
    clock_pin=board.D13, latch_pin=board.D0, output_enable_pin=board.D1)
display = framebufferio.FramebufferDisplay(matrix, auto_refresh=False)
```

Additional definitions

The three states let us track whether each wheel is stopped, running, or braking.
shuffled lets us take an input sequence and return a new sequence with the order of the elements randomized. This will be used to put the symbols in a different order on each wheel.

Wheel is a class that controls the behavior of each of the 3 fruit wheels. Check out the full source code if you want to see how it works!

```
# Each wheel can be in one of three states:
STOPPED, RUNNING, BRAKING = range(3)

# Return a duplicate of the input list in a random (shuffled) order.
def shuffled(seq):
    return sorted(seq, key=lambda _: random.random())

# The Wheel class manages the state of one wheel. "pos" is a position in
# floating point coordinates, with one 1 pixel being 1 position.
# The wheel also has a velocity (in positions
# per tick) and a state (one of the above constants)
class Wheel(displayio.TileGrid):
    def __init__(self, bitmap, pixel_shader):
        ...  
    def step(self):
        """Update each wheel for one time step""

    def kick(self, i):
        """Set the wheel running again, using a slight bit of randomness.
        The 'i' value makes sure the first wheel brakes first, the second
        brakes second, and the third brakes third.""

Setting up the wheels

First, load the emoji bitmap into RAM memory. It is small (just a few hundred bytes), and doing this lets the display update much more fluidly than when the bitmap is in FLASH memory.

Then, create 3 wheels and put them all in a group, setting their X and Y positions so that they span the display from left to right.

Want to learn more about setting up these kinds of images, which are also known as Sprite Sheets? There's a guide for that!

```
# This bitmap contains the emoji we're going to use. It is assumed
# to contain 20 icons, each 20x24 pixels. This fits nicely on the 64x32
# RGB matrix display.
bitmap, palette = adafruit_imageload.load(
    "~/emoji.bmp",
    bitmap=displayio.Bitmap,
    palette=displayio.Palette)

# Our fruit machine has 3 wheels, let's create them with a correct horizontal
# (x) offset and arbitrary vertical (y) offset.
g = displayio.Group()
wheels = []
for idx in range(3):
    ```
wheel = Wheel(bitmap, palette)
wheel.x = idx * 22
wheel.y = -20
g.append(wheel)
wheels.append(wheel)
display.root_group = g

Setting up the activation switch

The switch is connected to A1, and closing it connects it to GND. By enabling a pull up, the `button.value` will be `False` when it is pressed and `True` otherwise.

```python
# We want a digital input to trigger the fruit machine
button = digitalio.DigitalInOut(board.A1)
button.switch_to_input(pull=digitalio.Pull.UP)
```

Setting up the background sounds

The speaker is connected to A0. Play casino-style noises in MP3 format. If you want to change the MP3, make sure you use a file with the following properties:

- 16000 or 22050 sample rate
- 1 channel (mono)
- 64kbit/s or lower MP3 CBR encoding

You might also like to have multiple different sounds, for instance to play when the lever is pulled and when the wheels stop. You can change to a different sound file by opening a new file and assigning it to `sample.file`. Make sure your files all match in their sample rate, channels, and encoding.

```python
mp3file = open("/slots.mp3", "rb")
sample = audiomp3.MP3Decoder(mp3file)
speaker = audioio.AudioOut(board.A0)
speaker.play(sample, loop=True)
```

Main loop

Here’s how the main loop runs:

- Refresh the screen each time through the loop
- If all the wheels are stopped, then wait for the lever to be pulled (check `button.value`) and when it is, set all the wheels in motion by `kick()` ing them
- Otherwise, let each wheel rotate a little bit by calling `step()`
while True:
    # Refresh the display (doing this manually ensures the wheels move
    # together, not at different times)
    display.refresh(minimum_frames_per_second=0, target_frames_per_second=60)

    all_stopped = all(si.state == STOPPED for si in wheels)
    if all_stopped:
        # Once everything comes to a stop, wait until the lever is pulled and
        # start everything over again. Maybe you want to check if the
        # combination is a "winner" and add a light show or something.
        while button.value:
            pass
        for idx, si in enumerate(wheels):
            si.kick(idx)

    # Otherwise, let the wheels keep spinning...
    for idx, si in enumerate(wheels):
        si.step()
Install 2x10 Socket Header
Place the 2x10 socket header onto the bottom of the RGB Matrix FeatherWing. Solder the pins in place to secure the socket header to the PCB.

Install Power Connectors
Place the screw block terminal and 2.1 barrel jack on top of the RGB FeatherWing. Solder the pins in place to secure the two power connectors.

Install Female Headers on RGB Matrix FeatherWing
Install the 12-pin and 16-pin onto the top of the RGB Matrix FeatherWing. Use the row of pins that are close to the edge of the PCB.
Solder Female Headers
Use the Feather M4 Express to keep the headers on the RGB Matrix FeatherWing steady while soldering pins.

RGB Matrix FeatherWing Polarity Check
Match the outline of the header on the silkscreen with the outline on the RGB matrix. The notch in the center should be facing the same way. Reference the image for the correct polarity.

Install RGB Matrix FeatherWing
The RGB Matrix FeatherWing is fitted on top of the IDC shrouded header on the display. Use the IDC shrouded header that features an arrow pointing to the right.
Install Feather M4 Express
The Feather M4 Express is placed on top of the RGB Matrix FeatherWing with the USB port facing the 2.1mm barrel jack.

Wiring

Power Cable
The RGB Matrix includes a plug in power cable. It features two female connectors that are linked together with two male terminals. The male terminals are connected to the screw block terminals on the RGB Matrix FeatherWing. The female connect is connected to the power header on the back of the RGB Matrix display.

Trim Power Cable
The cable is cut and shortened to make the wiring more organized. The wires measure to 90mm in length.
Slice Power Cable
Use wire strippers to remove a bit of insulation from each wire. Tin the exposed wire using a bit of solder. Add pieces of heat shrink tubing before slicing the wires together.

Shortened Power Cable
The power cable is shortened and has heat shrink tubing to insulate the exposed wires. The whole cable measures 18cm in length.

Wire PAM8302 Audio Amp
Use a piece of silicone ribbon cable to wire the audio amplifier to the RGB Matrix FeatherWing. The cable measures 70mm in length. A 4-wire silicone ribbon cable is soldered to the A+, A-, GND and VIN pins. The speaker is soldered to the positive and negative pins on the PAM8302.
Wire Speaker
The speaker can be wired directly to the PAM8302 amp. In this project, we used a 2-pin JST extension cable so that it can be easily disconnected. The cable measures 10.6cm in length.

Wire Button Switch
The tactile button is wired to a lengthy piece of silicone ribbon cable. The cable measures 15.2cm in length.

Solder Wires to RGB Matrix FeatherWing
Make the following connections to the RGB Matrix FeatherWing.

A+ from Amp to A0 on FeatherWing
A- from Amp to GND on FeatherWing
VIN from Amp to 3V on FeatherWing
GND from Amp to GND on FeatherWing
Button to A1 on FeatherWing
Button to GND on FeatherWing
Wired RGB Matrix FeatherWing
Double check the wiring is correct.

Assembly

Connect Power Cable
Insert the negative and positive leads from the power cable to the screw block terminals. Double check the polarities are correct. Use a small flat head screw driver to tighten the terminals to secure them in place.

Circuit Wiring Review
Double check the wiring to ensure components are properly connected.
Secure Amp to Mount
The PAM8302 amp snap fits into the built-in holder on the mount. Orient the mounting holes with the standoffs. Insert PCB at an angle and firmly press down to install.

Secure Speaker to Mount
The magnet from the speaker is press fitted into the holder on the mount.

Install Frame to Matrix
Insert the RGB LED Matrix into the frame. Reference the labels on the silkscreen to determine the "upright" orientation.
Frame Slot
Check the side of the frame with the slot is facing the right side of the RGB LED Matrix.

Install Acrylic to Frame
The sheet of black LED acrylic is inserted and pushed through the slot on the right side of the frame. Note: The photo is showing the matrix upside down, hence on the left side of the frame.

Install Feet
Place one of the feet over the two mounting holes on the back of the RGB LED Matrix. Line up the tabs on the foot with the mounting holes.
Secure Feet
Use the included M3 thumb screws to secure the foot to the RGB Matrix.

Installed Feet
Proceed to secure the second foot to the RGB LED Matrix using the remaining M3 thumb screws. Reference the photo for the correct placement and orientation.

Install FeatherWing and Mount
Insert the RGB Matrix FeatherWing onto the 2x8 IDC shrouded header on the RGB Matrix. Place the mount over the second set of mounting holes to the right of the FeatherWing. Reference the photo for correct orientation and placement.
Secure Mount to RGB Matrix
Use 2x M3 x 8mm long machine screws to secure the mount to the back of the RGB matrix.

Connect Speaker to Amp
Plug in the JST connectors from the PAM8302 amp to the speaker.

Plug In Power Cable to RGB Matrix
Insert the connector from the power cable to the power port on the back of the RGB Matrix. Check to ensure the tab is fully seated over the connector.
Wire Management
Optionally adjust the cables so they're fitted underneath the speaker and amp mount. The mount can be used to hold the excess wiring in place.

Secure Handle to Frame
Place the handle mechanism over the right side of the frame. Line up the mounting holes. Insert and fasten 2x M3 x 10mm long machine screws through the two mounting holes on the handle. Fasten tight.

Installed Handle
Check the handle is in the correct position and can be properly pulled down and springs back up automatically.
Install Button to Handle
Place the button into the built-in holder on the side of the handle.

Test Button Handle Actuation
Pull the handle down and test out the button actuation. The lever should actuate the button and trigger the slot machine animations.