Arcade Synth Controller

Created by John Park

https://learn.adafruit.com/arcade-synth-controller

Last updated on 2022-02-04 11:12:55 AM EST
# Table of Contents

## Overview
- Parts 4
- Optional 6
- Enclosure Materials 7

## Code the Arcade Synth
- Easy Code Upload 7

## Build the Arcade Synth Circuit
- The Circuit 15
- The Proto Board 15
- The Proto Board 17
- The Feather Tripler 18
- LED Arcade Button STEMMA QT Boards 21
- Arcade Button Wiring 22
- Reset and Enable Button Wiring 23

## Build the Arcade Enclosure
- The Enclosure 24
- Enclosure Parts 25
- Affix the Arcade Boards 26
- Affix the Tripler 28
- Rear Panel Buttons 30
- Side Panels 32
- Front Panel Buttons 33
- Top Layers Buttons 35
- Joystick Attach 37
- Plug in the Buttons 38
- Close the Case 40
- Joystick Washer and Ball 41
- Feet 41

## Play the Arcade Synth Controller

©Adafruit Industries
Overview

Build your own arpeggiator synthesizer/MIDI controller in the form of a fight stick/piano keyboard mashup! It's perfect for chiptunes. You'll use LED Arcade 1x4 boards, MIDI FeatherWing, Feather M4 Express, and lots of shiny buttons to build this retro-style synth.

Play root notes with the light-up arcade buttons, adjust tempo and octave with the joystick, and use the front-panel function buttons to start/stop the arps, adjust arp range, and to select different patterns.

Output square wave synth sounds over stereo headphone jack, and/or send MIDI over USB or classic DIN-5 MIDI cable to your software and hardware synthesizers.

This project is inspired directly by the amazing Pianocade (https://adafruit.it/Ybl).

On-board synthesis using Audio Library:

Arcade Synth Controller controlling Game Boy via MIDI:
Parts

**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

**Adafruit LED Arcade Button 1x4 - STEMMA QT I2C Breakout**
The only thing better than a glowy arcade button is, perhaps, FOUR glowing arcade buttons - and that's what the Adafruit LED Arcade Button 1x4 QT I2C...
https://www.adafruit.com/product/5296

4 x **Adafruit LED Arcade Button 1x4 STEMMA QT I2C Breakout**
https://www.adafruit.com/product/5296

**Adafruit MIDI FeatherWing Kit**
Turn your Feather into a song-bird with this musically-enabled FeatherWing that adds MIDI input and output jacks to just about any Feather. You get both input and output DIN-5 MIDI...
https://www.adafruit.com/product/4740
Small Arcade Joystick
This snappy 8-way joystick beckons you to play with it! It is rugged, and not too large, reminiscent of a Pac Man arcade cabinet. Unlike the potentiometer-based 2-axis and mini...
https://www.adafruit.com/product/480

13 x Arcade Button with LED
30mm Translucent Red
https://www.adafruit.com/product/3489

3 x 30mm Translucent Clear
Arcade Button with LED
https://www.adafruit.com/product/3491

1 x FeatherWing Tripler Mini Kit - Prototyping Add-on For Feathers
This is the FeatherWing Tripler - a prototyping add-on and more for all Feather boards. This is similar to our

1 x FeatherWing Proto - Prototyping Add-on For All Feather Boards
https://www.adafruit.com/product/2884

1 x Breadboard-Friendly 3.5mm Stereo Headphone Jack
TRS
https://www.adafruit.com/product/1699

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

1 x STEMMA QT / Qwiic JST SH 4-Pin Cable
200mm Long
https://www.adafruit.com/product/4401
1 x SparkFun STEMMA QT / Qwiic Breadboard Breakout Adapter
I2C - JST-SH

16mm Illuminated Pushbutton - Green
Latching On/Off Switch
A switch is a switch, and an LED is an LED, but this LED illuminated button is a lovely combination of both! It’s a medium sized button, large enough to press easily but not too...
https://www.adafruit.com/product/1443

16mm Illuminated Pushbutton - Yellow
Momentary
A button is a button, and an LED is a LED, but this LED illuminated button is a lovely combination of both! It’s a medium sized button, large enough to press easily but not too big...
https://www.adafruit.com/product/1441

4 x Arcade Button Quick-Connect Wire Pairs
0.11" (10 pack)
https://www.adafruit.com/product/1152

1 x 10μF 50V Electrolytic Capacitors
Pack of 10
https://www.adafruit.com/product/2195

1 x Through-Hole Resistors 1.0K Ω
5% 1/4W - Pack of 25
https://www.adafruit.com/product/4294

2 x Little Rubber Bumper Feet
Pack of 4
https://www.adafruit.com/product/550

Optional

You can use two or three sets of aluminum bumper feet for the base if you're feeling highly extra!
Gray Anodized Aluminum Bumper Feet - Pack of 2
Keep your electronics from going barefoot, and give them sleek cyberpunk metal feet! These aluminum bumpers are originally designed for keyboard enclosures that are made of anodized...
https://www.adafruit.com/product/5049

Enclosure Materials

To build the enclosure you'll need 3mm thick ("1/8") acrylic stock in gray, black, and white:

- 2x sheets of matte gray (https://adafruit.it/Ybm) 1/8" x 12"x24" (https://adafruit.it/Ybm)
- 1x sheet of frosted black (https://adafruit.it/Ybo) 1/8" x 12"x24" (https://adafruit.it/Ybo)
- 1x sheet of translucent white 1/8" x 12"x24" (https://adafruit.it/Ybn)

You'll use M4 screws and nuts to secure the case, as well as the joystick:

- M4 x 0.7mm nuts (https://adafruit.it/Ybp)
- M4 x 80mm socket head screws (https://adafruit.it/Ybq)
- M4 x 14mm socket head screws (https://adafruit.it/Ybr)

Code the Arcade Synth

Easy Code Upload

You can get the code onto your Feather M4 Express as easy as drag-and-drop! Simply plug in the Feather to your computer with a known good USB data cable (not power only!) and then double-click the reset button.

The board will show up on your computer as a USB drive named FEATHERBOOT. Download the Arcade_Synth_Controller.UF2 file linked below and then drag it onto the FEATHERBOOT drive.
The board will automatically reset and run the code.

Arcade_Synth_Controller.UF2

https://adafruit.it/YaV

Source Code Hacking

If you want to dig in deeper, you can download the source code here (https://adafruit.it/YaW). This will require some knowledge of the Arduino IDE and how to upload code to your board.

```cpp
// SPDX-FileCopyrightText: 2022 John Park and Tod Kurt for Adafruit Industries
// SPDX-License-Identifier: MIT

// Arcade Synth Controller II: Son of Pianocade -- The Enriffening
// written by John Park and Tod Kurt
// Synthesizer/MIDI arpeggiator for with multiple LED Arcade boards & joystick input

// Arpy library: https://github.com/todbot/mozzi_experiments/blob/main/eighties_arp/Arpy.h
// midi_to_freq and ADT patch: https://github.com/todbot/tal_experiments/tree/main/arpy_test

// - to do: when arp is off it acts as a normal keyboard.
#include <Arduino.h>
#include <Adafruit_TinyUSB.h>
#include <MIDI.h>
#include <Audio.h>
#include <Bounce2.h>
#include "Adafruit_seesaw.h"

#include "ADT.h"
#include "midi_to_freq.h"
#include "Arpy.h"

// ----- LED Arcade 1x4 STEMMA QT board pins-----
// pin definitions on each LED Arcade 1x4
#define SWITCH1 18  // PA01
#define SWITCH2 19  // PA02
#define SWITCH3 20  // PA03
#define SWITCH4 2   // PA04
#define PWM1 12  // PC00
#define PWM2 13  // PC01
#define PWM3 0  // PA04
#define PWM4 1  // PA05
#define I2C_BASE_ADDR 0x3A // boards are in order, 0x3A, 0x3B, 0x3C, 0x3D
#define NUM_BOARDS 4

Adafruit_seesaw ledArcades[ NUM_BOARDS ];

//----- board variables
int boardNum = 0;   //used to read each board
int switchNum = 0;  //used to read each switch
int boardSwitchNum = 0; //unique button ID accross all boards/buttons
int led_low = 10;   //min pwm brightness
int led_med = 60;
int led_high = 220; // max brightness
```
```c
bool lastButtonState[16];
bool currentButtonState[16];

//---- joystick pins-----
const int joyDownPin = 11;  // down
const int joyUpPin = 12;     // up
const int joyLeftPin = 9;    // left
const int joyRightPin = 10;  // right
const int joyGroundPin = 6;  // "fake" ground pin

//---- joystick debouncer
Bounce joyDown = Bounce();
Bounce joyUp = Bounce();
Bounce joyLeft = Bounce();
Bounce joyRight = Bounce();

//---- MIDI instances-----
Adafruit_USBD_MIDI usb_midi;
MIDI_CREATE_INSTANCE(Adafruit_USBD_MIDI, usb_midi, MIDIusb);  // USB MIDI
MIDI_CREATE_INSTANCE(HardwareSerial, Serial1, MIDIclassic);  // classic midi over RX/TX

//----- Audio Library Synth parameters
#define NUM_VOICES 4
AudioSynthWaveform *waves[] = {
    &wave0, &wave1, &wave2, &wave3
};

int filterf_max = 6000;
int filterf = filterf_max;

uint32_t lastControlMillis = 0;

uint8_t arp_octaves = 1;
uint8_t root_note = 0;

//---- create arpy arpeggiator
Arpy arp = Arpy();

int bpm = 160;
int octave_offset = 3;  // initially starts on MIDI note 36 with the offset of 3 octaves from zero
bool arp_on_off_state;

void setup() {
    Wire.setClock(400000);
    //----- MIDI and Serial setup-----
    //
    MIDIusb.begin(MIDI_CHANNEL_OMNI);
    MIDIclassic.begin(MIDI_CHANNEL_OMNI);
    Serial.begin(115200);
    MIDIusb.turnThruOff();
    delay(2000);  // it's hard getting started in the morning
    Serial.println("[.::.:::.] Welcome to Arcade Synth Controller II: Son of Pianocade -- The Enriffening [.::.:::]");
    Serial.println("MIDI USB/Classic and Serial have begun");
    //----- end MIDI and Serial setup-----

    //----- joystick pins setup-----
    //
    pinMode(joyDownPin, INPUT);
    pinMode(joyUpPin, INPUT);
    pinMode(joyLeftPin, INPUT);
    pinMode(joyRightPin, INPUT);
    pinMode(joyGroundPin, OUTPUT);
```
joyDown.attach( joyDownPin, INPUT_PULLUP);
joyUp.attach( joyUpPin, INPUT_PULLUP);
joyLeft.attach( joyLeftPin, INPUT_PULLUP);
joyRight.attach( joyRightPin, INPUT_PULLUP);
digitalWrite(joyGroundPin, LOW);

//----- end joystick pins setup-----

//----- LED Arcade 1x4 setup-----
//
for ( int i = 0; i < NUM_BOARDS; i++ ) {
    if ( !ledArcades[i].begin( I2C_BASE_ADDR + i ) ) {
        Serial.println(F("LED Arcade not found!
        while (1) delay(10);
    }

for ( int i = 0; i < NUM_BOARDS; i++ ) {
    ledArcades[i].pinMode(SWITCH1, INPUT_PULLUP);
    ledArcades[i].pinMode(SWITCH2, INPUT_PULLUP);
    ledArcades[i].pinMode(SWITCH3, INPUT_PULLUP);
    ledArcades[i].pinMode(SWITCH4, INPUT_PULLUP);
    ledArcades[i].analogWrite(PWM1, led_low);
    ledArcades[i].analogWrite(PWM2, led_low);
    ledArcades[i].analogWrite(PWM3, led_low);
    ledArcades[i].analogWrite(PWM4, led_low);
}

// brighten default root note
ledArcades[0].analogWrite(PWM1, led_high);

// turn down brightness of the function buttons
ledArcades[3].analogWrite(PWM2, 0);
ledArcades[3].analogWrite(PWM3, led_low);
ledArcades[3].analogWrite(PWM4, led_low);

//----- end LED Arcade 1x4 setup-----

//----- Arpy setup-----
//
arpp.setNoteOnHandler(noteOn);
arpp.setNoteOffHandler(noteOff);
arpp.setRootNote( root_note );
arpp.setOctaveOffset(octave_offset);
arpp.setBPM( bpm );
arpp.setGateTime( 0.75 ); // percentage of bpm
arpp.off();

//----- Audio Library Synth setup-----
// (patch is saved in ADT.h file)
AudioMemory(120);

filter0.frequency(filterf_max);
filter0.resonance(0.5);

env0.attack(10);
env0.hold(2);
env0.decay(100);
env0.sustain(0.5);
env0.release(100);

// Initialize processor and memory measurements
AudioProcessorUsageMaxReset();
AudioMemoryUsageMaxReset();

Serial.println("Arpy setup done");
}

// end setup()

int waveform = WAVEFORM_SQUARE;
void noteOn(uint8_t note) {
  waves[0]->begin(0.9, tune_frequencies2_PGM[note], waveform);  
  waves[1]->begin(0.9, tune_frequencies2_PGM[note] * 1.01, waveform); // detune  
  waves[2]->begin(0.9, tune_frequencies2_PGM[note] * 1.005, waveform); // detune  
  waves[3]->begin(0.9, tune_frequencies2_PGM[note] * 1.025, waveform); // detune  
  filterf = filterf_max;  
  filter0.frequency(filterf);  
  env0.noteOn();  
  MIDIusb.sendNoteOn(note, 127, 1);  
  MIDIclassic.sendNoteOn(note, 127, 1);  
}

void noteOff(uint8_t note) {
  env0.noteOff();  
  MIDIusb.sendNoteOn(note, 0, 1);  
  MIDIclassic.sendNoteOn(note, 0, 1);  
}

void midiPanic() {
  for (uint8_t m = 0; m < 128; m++) {
    MIDIusb.sendNoteOn(m, 0, 1);  
    MIDIclassic.sendNoteOn(m, 0, 1);  
    yield(); // keep usb midi from flooding  
  }
}

void lightLED(uint8_t buttonLED) {
  uint8_t pwms[4] = {PWM1, PWM2, PWM3, PWM4};  
  boardNum = map(buttonLED, 0, 12, 0, 3);  
  // first dim all buttons on first three boards  
  for (int b = 0; b < 3; b++) {
    for (int p = 0; p < 4; p++) {
      ledArcades[b].analogWrite(pwms[p], led_low);
    }
  }
  // dim first button on fourth board (the other two are function buttons)  
  ledArcades[3].analogWrite(PWM1, led_low);  
  // then brighten the selected one  
  ledArcades[boardNum].analogWrite(pwms[buttonLED % 4], led_high);
}

#define SWITCHMASK ((1 << SWITCH1) | (1 << SWITCH2) | (1 << SWITCH3) | (1 << SWITCH4))

void arcadeButtonCheck() {
  for (boardNum = 0; boardNum < NUM_BOARDS; boardNum++) {
    // check all boards, all switches  
    int pos = boardNum*4;  
    uint32_t switches = ledArcades[boardNum].digitalReadBulk(SWITCHMASK);  
    currentButtonState[pos+0] = !(switches & (1<<SWITCH1));  
    currentButtonState[pos+1] = !(switches & (1<<SWITCH2));  
    currentButtonState[pos+2] = !(switches & (1<<SWITCH3));  
    currentButtonState[pos+3] = !(switches & (1<<SWITCH4));
  }
  for (int i = 0; i < 4*NUM_BOARDS; i++) {
    bool state = currentButtonState[i];  
    if(state != lastButtonState[i]) {
      if( state == HIGH ) { //pressed  
        // ---button functions---  
        // ---root notes---  
        if (i < 13) { // these are the piano keys for picking root notes  
          root_note = 0 + i; // MIDI note  
          lightLED(i);
        }
      }
    }
  }
}
if (i == 13) { // arp pattern button on front panel
    if (!arp_on_off_state) {
        arp.on();
        ledArcades[3].analogWrite(PWM2, led_med);
        arp_on_off_state = true;
    } else {
        arp.off();
    }
}

if (i == 14) { // arp range button on front panel
    ledArcades[3].analogWrite(PWM3, led_high);
    arp_octaves = arp_octaves + 1;
    if (arp_octaves == 4) { arp_octaves = 1; }
    arp.setTransposeSteps(arp_octaves);
    Serial.printf("arp steps:%d\n", arp_octaves);
    ledArcades[3].analogWrite(PWM3, led_low);
}

if (i == 15) { // pattern button
    ledArcades[3].analogWrite(PWM4, led_high);
    arp.nextArpId();
    ledArcades[3].analogWrite(PWM4, led_low);
}

for (int i = 0; i < 4 * NUM_BOARDS; i++) {
    lastButtonState[i] = currentButtonState[i];
}

void loop()
{
    arcadeButtonCheck(); // see if any buttons are pressed, send notes or adjust parameters
    joyDown.update();
    joyUp.update();
    joyLeft.update();
    joyRight.update();

    if (joyUp.fell() ) { // read a joystick single tap
        ledArcades[3].analogWrite(PWM3, led_high); // feedback on front panel button
        octave_offset = octave_offset + 1;
        if (octave_offset > 7) { octave_offset = 7; }
        arp.setOctaveOffset(octave_offset);
        ledArcades[3].analogWrite(PWM3, led_low);
    }

    if (joyDown.fell() ) {
        ledArcades[3].analogWrite(PWM3, led_high); // feedback on front panel button
        octave_offset = octave_offset - 1;
        if (octave_offset < 0) { octave_offset = 0; }
        arp.setOctaveOffset(octave_offset);
        ledArcades[3].analogWrite(PWM3, led_low);
    }

    int joyLeftVal = joyLeft.read(); // read a held joystick (autorepeat) instead of single tap
    if( joyLeftVal == LOW ) {
        bpm = bpm - 1;
        if(bpm < 100) { bpm = 100; }
        ledArcades[3].analogWrite(PWM4, led_high);
    }
}
MIDI

The Arcade_Synth_Controller.ino code plays notes via the Audio Library synthesizer, as well as sending out MIDI via USB and Classic DIN-5 connection. To adjust which channels are used, change the `MIDIusb.sendNoteOn` or `0ff` lines here from channel 1 to whichever you need.

```cpp
void noteOn(uint8_t note) {
    waves[0] -> begin( 0.9, tune_frequencies2_PGM[note], waveform);
    waves[1] -> begin( 0.9, tune_frequencies2_PGM[note] * 1.01, waveform); // detune
    waves[2] -> begin( 0.9, tune_frequencies2_PGM[note] * 1.005, waveform); // detune
    waves[3] -> begin( 0.9, tune_frequencies2_PGM[note] * 1.025, waveform); // detune
    filterf = filterf_max;
    env0.noteOn();
    MIDIusb.sendNoteOn(note, 127, 1);
    MIDIclassic.sendNoteOn(note, 127, 1);
}

void noteOff(uint8_t note) {
    env0.noteOff();
    MIDIusb.sendNoteOn(note, 0, 1);
    MIDIclassic.sendNoteOn(note, 0, 1);
}

void midiPanic(){
    for( uint8_t m = 0; m &lt; 128; m++ ){
        MIDIusb.sendNoteOn(m, 0, 1);
        MIDIclassic.sendNoteOn(m, 0, 1);
        yield(); // keep usb midi from flooding
    }
}
```
Arpeggios

The Arpy.h class (https://adafru.it/YaX), design by awesome guy Tod Kurt (https://adafru.it/YaY), handles the playing of arpeggio patterns based on the root note played on the keyboard.

You may want to edit the existing patterns or create your own. This is where to make those changes in the Arpy.h code:

```c
int8_t arps[arp_count][arp_len] = {
    {0, 4, 7, 12},  // major
    {0, 3, 7, 10},  // minor 7th
    {0, 3, 6, 3},   // Diminished
    {0, 5, 7, 12},  // Suspended 4th
    {0, 12, 0, -12}, // octaves
    {0, 12, 24, -12}, // octaves 2
    {0, -12, -12, 0}, // octaves 3 (bass)
    {0, 0, 0, 0},   // root
};
```

Synthesizer Design

The Audio System Design Tool (https://adafru.it/Dkn) can be used to design your synth modules and patch connections. In the example used here, the patch is contained in the ADT.h file.

```c
AudioSynthWaveform wave0;  //xy=502.74795150756836,82.7552137374878
AudioSynthWaveform wave1;  //xy=504.28649139404297,117.86524295806885
AudioSynthWaveform wave2;  //xy=503.2865982055664,153.0081024169922
AudioSynthWaveform wave3;  //xy=502.8580284118653,188.6524295806885
AudioMixer4 mixer0;  //xy=633.7151184082031,100.00811004638672
AudioEffectEnvelope env0;  //xy=758.612813949585,54.04482841491699
AudioFilterStateVariable filter0;  //xy=888.6010780334473,60.850419998168945
AudioMixer4 mixerA;  //xy=1010.7359161376953,171.30673599243164
AudioMixer4 mixerL;  //xy=1196.8192749023438,210.86235809326172
AudioMixer4 mixerR;  //xy=1198.2637329101562,277.8345947265625
AudioOutputAnalogStereo audioOut;  //xy=1360.3193969726562,250.61236572265625
AudioConnection patchCord1(wave0, 0, mixer0, 0);
AudioConnection patchCord2(wave3, 0, mixer0, 3);
AudioConnection patchCord3(wave2, 0, mixer0, 2);
AudioConnection patchCord4(wave1, 0, mixer0, 1);
AudioConnection patchCord5(mixer0, env0);
AudioConnection patchCord6(env0, 0, filter0, 0);
AudioConnection patchCord7(filter0, 0, mixerA, 0);
AudioConnection patchCord8(mixerA, 0, mixerL, 0);
AudioConnection patchCord9(mixerA, 0, mixerR, 0);
AudioConnection patchCord10(mixerL, 0, audioOut, 0);
AudioConnection patchCord11(mixerR, 0, audioOut, 1);
```
Build the Arcade Synth Circuit

The Circuit

The Proto Board

The FeatherWing Proto board is great for adding on a small audio output circuit, as well as connections for the STEMMA QT ports over I2C, joystick wiring, and the reset and enable switches.

The schematic below shows these connections as they relate to the Feather board, but you’ll make them on the Proto board and rely on the FeatherWing Tripler to connect those points to the Feather.
The Proto Board
Following the schematic above, as well as these photos, solder the parts in place for the:

- audio output RC circuit (resistor capacitor circuit for removing noise from the audio)
- joystick wiring (note the use of pin D6 as common joystick switch ground)
- headers for the enable and reset switches
- STEMMA QT/Qwiic connector to SDA, SCL, 3v3, and GND.
The Feather Tripler
Add headers and M2.5 nylon standoffs to the Feather Tripler as shown here.

Insert the MIDI FeatherWing (with either DIN-5 or TRS plugs on it, your choice) to the Tripler, followed by the Proto FeatherWing and the Feather M4 Express.

Fasten the boards to the Tripler standoffs so they don't wiggle around when inserting cables later.
LED Arcade Button
STEMMA QT Boards

To use the four LED Arcade Button boards on I2C they must each have a unique address.

You will cut the traces on three of the boards to set them.

Set them to:

- 0x3A (default, don't cut any traces)
- 0x3B -- cut trace A0
- 0x3C -- cut trace A1
- 0x3D -- cut traces A0 and A1

Then, wire them to each other in order from left to right using 100mm STEMMA QT cables.
Arcade Button Wiring

Plug two sets of quick connect cables into each arcade button -- one for the switch, the other pair for the LED.

The polarity doesn't matter for the switch tabs (these are the ones protruding from the gray plastic switch base), however, you must connect the - and + properly for the LED to work. These are marked both on the Arcade Button board LED ports and on the molding of the switch base.

To avoid confusion, I marked these wires in advance using heat shrink tubing. I also used heat shrink to order each set of four buttons yellow, red, green, blue, as shown here. This type of organization helps immensely with wiring things properly and later troubleshooting!
Reset and Enable Button Wiring

I decided not to wire the LEDs of these switches, instead wiring only the switches. Polarity doesn't matter.

Remember, the yellow momentary switch will be plugged into the Feather's Reset pin, and the green toggle switch into Enable.

Build the Arcade Enclosure
The Enclosure

The Arcade Synth Controller enclosure is designed to be laser cut from 3mm thick acrylic. The .svg file below can be used on a laser cutter or sent to a service provider for cutting.

Arcade_Synth_case.svg
https://adafruit.it/YaZ
Enclosure Parts
Cut the parts from 3mm ("1/8") acrylic as shown. Finished surface side of material is place facing upwards for cutting.
Affix the Arcade Boards
Use M2.5 nylon standoffs, screws, and nuts to affix the Arcade Button boards to the base plate. The first board (address 0x3A) is on the left.

Note the orientation of the board so that everything else lines up later.
Affix the Tripler
Use the standoffs and nuts to attach the tripler as shown. Connect the STEMMA QT cable port to the Arcade Button Boards.
Rear Panel Buttons
Making sure the finish side of the back panel is facing out (so the MIDI out ports align with the holes properly) feed the reset and enable buttons into their holes, then thread on their retaining rings.

Plug the switches into their respective ports on the Feather Proto board.

Place the back panel notches into the base.
Side Panels
Place the side panel notches into the base.
Front Panel Buttons
With the finish side facing out, feed the three front panel arcade buttons into their holes as shown.

Thread on their retaining rings, then plug them into the fourth LED Arcade Button breakout board. Double-check that you've plugged in the LED cables and switch cables into their proper respective ports.
Top Layers Buttons
Sandwich the top panels together (NOTE: only the lower of the two has finger joint notches), then feed in the buttons, minding their yellow, red, green, blue ordering.

Thread on their retaining rings.
Joystick Attach
Unscrews the joystick topper and set it aside. It looks like an everlasting gobstopper, but don't eat it. It will last forever. And taste pretty bland.

With the interface pins facing inward toward the center of the controller (it won't fit otherwise!) feed the joystick stem up through the top panels, then screw them into place with the four M4 x 40mm screws and nuts.

Plug in the joystick wiring harness.
Plug in the Buttons

It's a plugging party!

Plug all of the remaining arcade buttons, following left-to-right order, into the LED Arcade Button breakout boards.
TEST: this is an excellent time to test that everything is working before you close things up!
Close the Case

Carefully close the case top, aligning the finger joint tabs and slots.

Fasten the six M4 x 80mm screws through the holes, using nuts to secure them at the bottom.
Joystick Washer and Ball
Replace the joystick's disk and ball topper.

Feet
You can use large rubber bumper feet to prevent the exposed screws from marring your desk, or get extra fancy and use four or even six M4 threaded keyboard feet as shown here.
Play the Arcade Synth Controller

Plug in a micro USB cable to the Feather, and plug the other end into your computer (particularly for USB MIDI usage) or a USB power supply.

For audio out, plug in a stereo 3.5mm TRS cable into the jack on the Protoboard -- plug the other end into your amplifier, powered speakers, or audio interface.

Optionally, if you’re sending out classic MIDI signals, plug a MIDI cable into the MIDI out, which is the MIDI port closest to the reset button.

Power on the Arcade Synth Controller by pressing the green switch after a moment the buttons will light up and you’re ready to play.

Start the arpeggiator by pressing the white button on the left on the front of the controller -- this button is unlit when the arps are stopped and lit up when they’re playing.

You will hear the synth playing a C major arpeggio, as the first keyboard button on the controller is lit by default. Success!

Here are the other controls to try:

- pick a different root note by pressing any red keyboard button. The root note will light up brighter than the others
• speed up or slow down the tempo with the joystick right/left
• increase or decrease the octave offset with the joystick up/down
• expand the arpeggio range from one, two, or three octaves of the pattern with the white middle button on the front panel
• cycle among eight different built-in arp patterns with the white right button on the front panel