



# Mini Commodore PET with Charlieplexed LED Matrix

Created by Ruiz Brothers



<https://learn.adafruit.com/mini-commodore-pet-with-charlieplexed-led-matrix>

Last updated on 2024-06-03 02:02:04 PM EDT

# Table of Contents

Overview	5
<hr/>	
<ul style="list-style-type: none"><li>• <a href="#">3D Printed Miniature</a></li><li>• <a href="#">Charlieplexed LED Matrix</a></li><li>• <a href="#">Prerequisite Guides</a></li><li>• <a href="#">Parts</a></li><li>• <a href="#">Tools &amp; Supplies</a></li><li>• <a href="#">Banana for Scale</a></li></ul>	
3D Printing	9
<hr/>	
<ul style="list-style-type: none"><li>• <a href="#">3D Printed Parts</a></li><li>• <a href="#">Slice Settings</a></li><li>• <a href="#">Download &amp; Remix</a></li><li>• <a href="#">2D Print the Label</a></li></ul>	
Software	11
<hr/>	
<ul style="list-style-type: none"><li>• <a href="#">Getting Code Onto M0 Feather</a></li><li>• <a href="#">Download Adafruit_IS31FL3731 library</a></li><li>• <a href="#">Install Adafruit GFX</a></li><li>• <a href="#">Lines of Code, The Matrix, and Game of Life</a></li></ul>	
Circuit Diagram	18
<hr/>	
<ul style="list-style-type: none"><li>• <a href="#">Wired Connections</a></li><li>• <a href="#">Battery Power</a></li></ul>	
LED Matrix	19
<hr/>	
<ul style="list-style-type: none"><li>• <a href="#">Charlieplexed Driver and LED Matrix</a></li><li>• <a href="#">Charlieplexed Driver Header Pins</a></li><li>• <a href="#">Mount Parts</a></li><li>• <a href="#">Side Profile</a></li><li>• <a href="#">Solder In Place</a></li><li>• <a href="#">Charlieplexed Matrix Assembly</a></li><li>• <a href="#">Trim Header Pins</a></li></ul>	
Wiring	22
<hr/>	
<ul style="list-style-type: none"><li>• <a href="#">Switch for Feather</a></li><li>• <a href="#">Tin Wires for Slide Switch</a></li><li>• <a href="#">Connect Wires to Slide Switch</a></li><li>• <a href="#">Connect Slide Switch to Feather M0</a></li><li>• <a href="#">Test Switch</a></li><li>• <a href="#">Wires for Charlieplexed Matrix</a></li><li>• <a href="#">Final Circuit Test</a></li><li>• <a href="#">Heat Shrink Tubing</a></li><li>• <a href="#">Connect Wires to Charlieplexed Matrix</a></li><li>• <a href="#">Connect Charlieplexed Matrix to Feather</a></li></ul>	
Assembly	25
<hr/>	
<ul style="list-style-type: none"><li>• <a href="#">Glue Keys to Cover</a></li><li>• <a href="#">Glue KeyCover to Frame</a></li><li>• <a href="#">Glue displayBase to Frame</a></li><li>• <a href="#">Install displayFace</a></li><li>• <a href="#">Tap Mounting Holes</a></li></ul>	

- Mount Feather to Cover
- Install Components into Frame
- Install Battery into Frame
- Install Cover to Frame
- Fasten Screws
- Checkpoint
- Install Charlieplexed Matrix into Display Cover
- Install Charlieplexed Matrix into Display Frame
- Install Switch to Back Cover
- Assembly Checkpoint 2
- Install Display to Frame
- Install Back Cover to Frame
- Crop Sticker Artwork
- Apply Sticker



---

# Overview



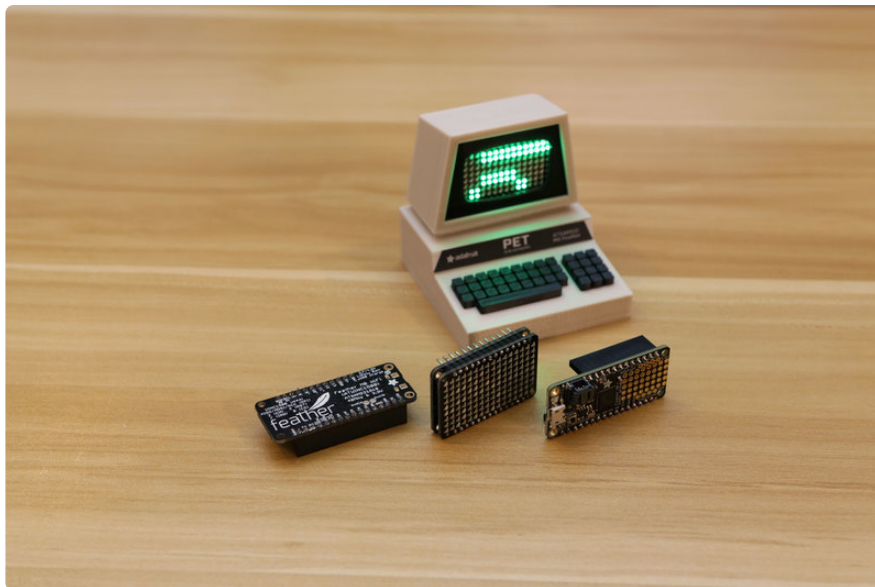
## 3D Printed Miniature

In this tutorial, you'll learn how to build and assemble a mini replica of the PET commodore. This isn't necessarily useful or an actual computer, it's just a cool prop.

You will however, hopefully gain some experience, learn something new and have fun :-)

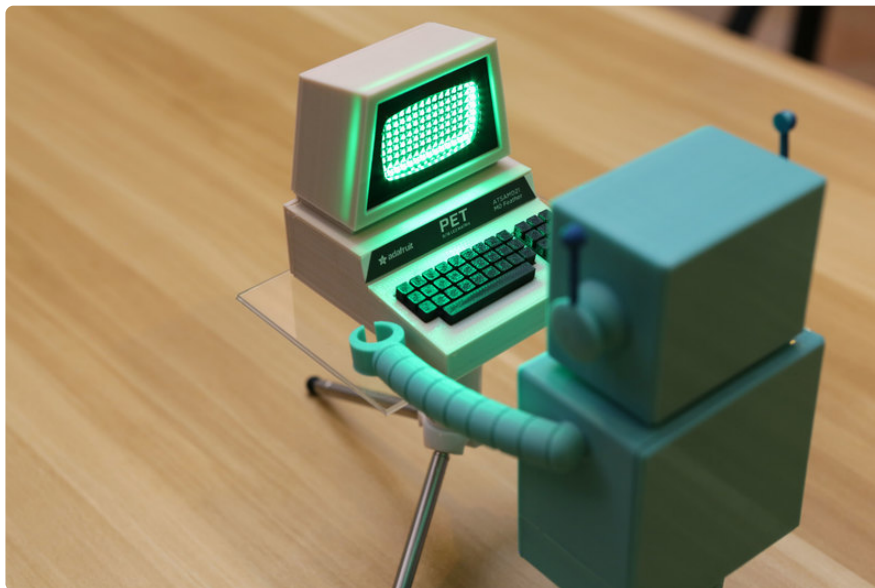
From Wikipedia:

The Commodore PET (Personal Electronic Transactor) is a line of home/personal computers produced starting in 1977 by Commodore International. A top-seller in the Canadian and United States educational markets, it was Commodore's first full-featured computer, and formed the basis for their entire 8-bit product line, including the Commodore 64. The first model, which was named the PET 2001, was the first personal computer ever made available to retail consumers.



## Charlieplexed LED Matrix

There's 144 LEDs in the Adafruit 16x9 charlieplexed matrix. It's a classic LED matrix look that uses the IS31FL3731 chipset and can PWM each individual LED in a 16x9 grid so you can have beautiful LED lighting effects.

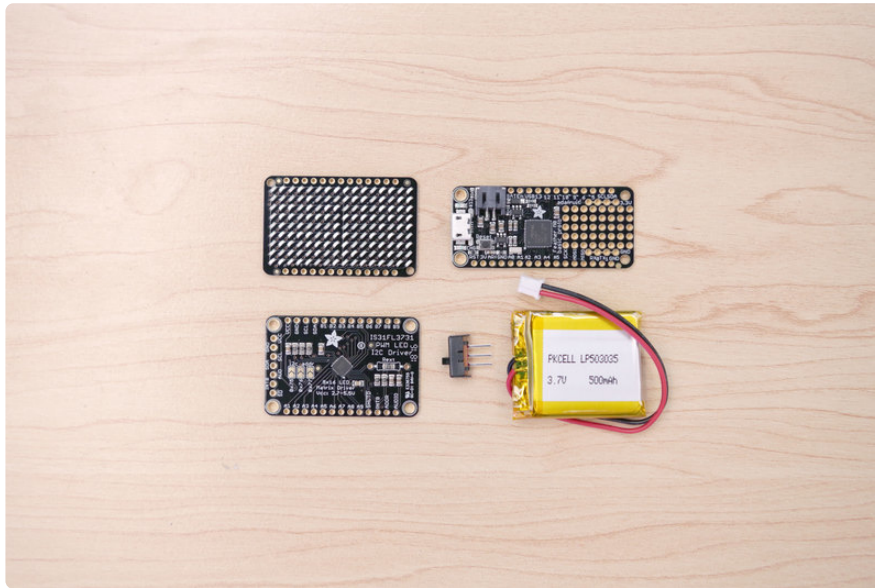


## Prerequisite Guides

We suggest walking through the following guides to get a better understanding of the components and Arduino IDE. We also have great tutorials on learning how to solder.

- [Adafruit Guide to Excellent Soldering \(https://adafru.it/sCa\)](https://adafru.it/sCa)
- [Adafruit Feather M0 Basic Proto \(https://adafru.it/sCb\)](https://adafru.it/sCb)

- [Adafruit 16x9 Charlieplexed PWM LED Matrix Driver – IS31FL3731 \(https://adafru.it/scE\)](https://adafru.it/scE)



## Parts

All of the components used in this project are listed below and in the right hand sidebar.

- [LED Charlieplexed Matrix \(https://adafru.it/scH\)](https://adafru.it/scH) - 9x16 LEDs (any color)
- [Adafruit 16x9 Charlieplexed PWM LED Matrix Driver - IS31FL3731 \(http://adafru.it/2946\)](http://adafru.it/2946)
- [Adafruit Feather M0 Basic Proto - ATSAMD21 Cortex M0 \(http://adafru.it/2772\)](http://adafru.it/2772)
- [500mAh Lipo Battery \(http://adafru.it/1578\)](http://adafru.it/1578)
- [SPDT Slide Switch \(http://adafru.it/805\)](http://adafru.it/805)

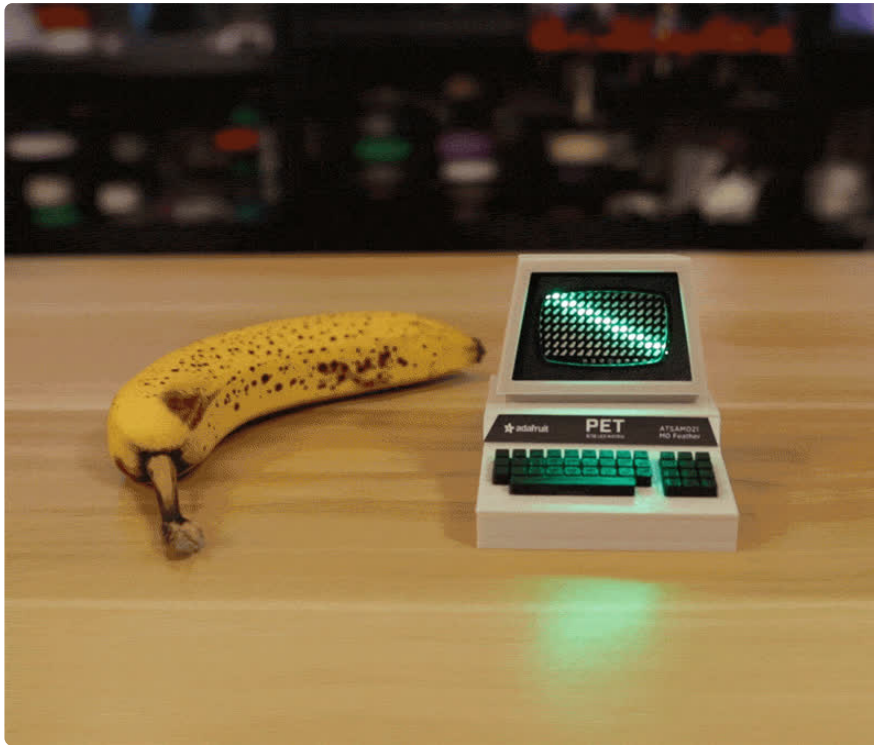


## Tools & Supplies

You'll need some tools and supplies to complete this project.

- [3D Printer \(https://adafru.it/diH\)](https://adafru.it/diH) & [Filament \(http://adafru.it/2080\)](http://adafru.it/2080)
- [Soldering Iron \(https://adafru.it/sFU\)](https://adafru.it/sFU) & [Solder \(https://adafru.it/sFV\)](https://adafru.it/sFV)
- [Panavise Jr. PV-201 \(http://adafru.it/151\)](http://adafru.it/151) or [Helping Third Hands \(http://adafru.it/291\)](http://adafru.it/291)
- [30AWG Wire \(http://adafru.it/2051\)](http://adafru.it/2051)
- [Wire Strippers \(http://adafru.it/527\)](http://adafru.it/527)
- E6000 adhesive
- [Heat Shrink Tubing \(http://adafru.it/1649\)](http://adafru.it/1649)
- 2x 4-40 3/8 (M3 5mm) machine screws
- 4x 2-56 3/8 (M2.5 5mm) machine screws





## Banana for Scale

A yummie system of measurement that is much easier and less snobby than inches, feet or even the metric system.

---

## 3D Printing



## 3D Printed Parts

Below is a list of all the 3D printed parts in this project. There's a total of 10 parts. The parts are oriented to 3D print "as-is" and does not require any support material.

- pet-displayBase.stl
- pet-keyCover.stl
- pet-frame.stl
- pet-frame-bottCover.stl
- pet-frame-backCover.stl
- pet-displayCover.stl
- pet-pad.stl
- pet-keys.stl
- pet-display.stl
- pet-displayFace.stl

## Slice Settings

Most of these settings will need to change depending on your 3D printers configuration, and choice of material. Below are the settings we used on a Sigma BCN3D printer using PLA filament.

- 220C extruder – 65C heated bed
- 20% Infill – 0.2mm layer height
- 60mm/s travel – 150mm/s default speed
- 0.4mm nozzle – 100% extrusion multiplier

**Download STLs from Thingiverse**

<https://adafru.it/sFW>

**Download STLs from Pinshape**

<https://adafru.it/sGc>

**Download STLs from YouMagine**

<https://adafru.it/sGd>

## Download & Remix

If you're interested in making changes to the design, the source is public. You can download the source in many different formats such as IGS, STEP, SAT, Google SketchUp, and more.

Download Fusion 360 Source

<https://adafru.it/sGe>

## 2D Print the Label

The label on the front panel is "2D" printed on sticker paper. You can make your own or download and use the one we made. This is obviously totally optional, but it does make the project resemble the Commodore PET series :-)

led\_matrix\_sticker.psd

<https://adafru.it/sFY>



---

## Software

### Getting Code Onto M0 Feather

In this portion of the guide, we'll get code uploaded to the **Adafruit Feather M0** micro-controller. If you don't write / understand code, don't to worry! You don't need to be a programmer to be able to upload prewritten code :-)

We'll walk you through the whole process.

First, visit the Feather M0 tutorial page by clicking the button below. Follow the instructions to download & setup the Arduino IDE and install drivers.

Getting Started with Adafruit  
Feather M0 Basic Proto

<https://adafru.it/sFZ>

Make sure you are able to get sketches compiled and uploaded, especially the blink example in the tutorial. Once you are comfortable with using the Feather M0, you can continue!

## Download Adafruit\_IS31FL3731 library

Before uploading the code, you will need to [download Adafruit\\_IS31FL3731 from our github repository \(https://adafru.it/IWA\)](https://adafru.it/IWA). You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip.

Download Adafruit\_IS31FL3731  
library

<https://adafru.it/IWB>

Rename the uncompressed folder **Adafruit\_IS31FL3731** and check that the **Adafruit\_IS31FL3731** folder contains **Adafruit\_IS31FL3731.cpp** and **Adafruit\_IS31FL3731.h**

Place the **Adafruit\_IS31FL3731** library folder your **arduinofolder/libraries/** folder.

You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

We also have a great tutorial on Arduino library installation at:  
<http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use> (<https://adafru.it/aYM>)

## Install Adafruit GFX

You will need to do the same for the **Adafruit\_GFX** library available here (<https://adafru.it/aJa>)

Download Adafruit\_GFX Library

<https://adafru.it/cBB>

Rename the uncompressed folder **Adafruit\_GFX** and check that the **Adafruit\_GFX** folder contains **Adafruit\_GFX.cpp** and **Adafruit\_GFX.h**

Place the **Adafruit\_GFX** library folder your **arduinofolder/libraries/** folder like you did with the IS31FL library

If using an older (pre-1.8.10) version of the Arduino IDE, also locate and install [Adafruit\\_BusIO](https://adafru.it/GxD) (<https://adafru.it/GxD>) (newer versions do this automatically when installing Adafruit\_GFX with the Arduino Library Manager).

## Lines of Code, The Matrix, and Game of Life

Written by [Phillip Burgess](https://adafru.it/iPc) (<https://adafru.it/iPc>), this sketch cycles between different effects: typing code, Conway's Game of Life, The Matrix effect, and a blank screen w/ blinking cursor. It doesn't require the GFX or IS31FL3731 library but instead addresses the CharliePlex driver chip directly to achieve smooth full-screen animation.

```
//-----
// Animated Commodore PET plaything. Uses the following parts:
//   - Feather M0 microcontroller (adafruit.com/products/2772)
//   - 9x16 CharliePlex matrix (2972 is green, other colors avail.)
//   - Optional LiPoly battery (1578) and power switch (805)
//
// This is NOT good "learn from" code for the IS31FL3731. Taking a cue from
// our animated flame pendant project, this code addresses the CharliePlex
// driver chip directly to achieve smooth full-screen animation. If you're
// new to graphics programming, download the Adafruit_IS31FL3731 and
// Adafruit_GFX libraries, with examples for drawing pixels, lines, etc.
//
// Animation cycles between different effects: typing code, Conway's Game
// of Life, The Matrix effect, and a blank screen w/blinking cursor (shown
// for a few seconds before each of the other effects; to imply "loading").
//-----

#include <Wire.h>;

#define I2C_ADDR 0x74          // I2C address of Charlieplex matrix
#define WIDTH    16           // Matrix size in pixels
#define HEIGHT   9            // Gamma-correction exponent
#define GAMMA    2.5          // Gamma-correction (brightness) table
uint8_t img[WIDTH * HEIGHT], // 8-bit buffer for image rendering
        bitmap[((WIDTH+7)/8) * HEIGHT], // 1-bit buffer for some modes
        gamma8[256],          // Gamma correction (brightness) table
        page = 0,              // Double-buffering front/back control
        frame = 0;             // Frame counter used by some animation modes
// More globals later, above code for each animation, and before setup()

// UTILITY FUNCTIONS -----

// Begin I2C transmission and write register address (data then follows)
uint8_t writeRegister(uint8_t n) {
  Wire.beginTransmission(I2C_ADDR);
  Wire.write(n); // No endTransmission() - left open for add'l writes
  return 2;      // Always returns 2; count of I2C address + register byte n
}

// Select one of eight IS31FL3731 pages, or the Function Registers
void pageSelect(uint8_t n) {
  writeRegister(0xFD); // Command Register
  Wire.write(n);        // Page number (or 0xB = Function Registers)
  Wire.endTransmission();
}

// Set bit at (x,y) in the bitmap buffer (no clear function, wasn't needed)
void bitmapSetPixel(int8_t x, int8_t y) {
  bitmap[y * ((WIDTH + 7) / 8) + x / 8] |= 0x80 >>> (x & 7);
}
```

```

}

// Read bit at (x,y) in bitmap buffer, returns nonzero (not always 1) if set
uint8_t bitmapGetPixel(uint8_t x, uint8_t y) {
    return bitmap[y * ((WIDTH + 7) / 8) + x / 8] & (0x80 >> (x & 7));
}

// BLINKING CURSOR / LOADING EFFECT -----
// Minimal animation - just one pixel in the corner blinks on & off,
// meant to suggest "program loading" or similar busy effect.

void cursorLoop() {
    img[0] = (frame & 1) * 255;
}

// TERMINAL TYPING EFFECT -----
// I messed around trying to make a random "fake code generator," but it
// was getting out of hand. Instead, the typed "code" is just a bitmap!

const uint16_t codeBits[] = {
    0b1110111110100000,
    0b0011101100000000,
    0b0011011111101000,
    0b0000111111011100,
    0b0000111011100000,
    0b0010000000000000,
    0b0011011111000000,
    0b1000000000000000,
    0b0000000000000000,
    0b1111011010000000,
    0b0011110111101110,
    0b1000000000000000,
    0b0000000000000000,
    0b1110111101000000,
    0b0011101011010000,
    0b0011110111110000,
    0b0011101101110000,
    0b0011011111101000,
    0b0000111011111100,
    0b0010000000000000,
    0b1000000000000000,
    0b0000000000000000,
    0b1110110100000000,
    0b0011011111101000,
    0b0000111101100000,
    0b0010000000000000,
    0b0011110111110000,
    0b0011101111011000,
    0b1000000000000000,
    0b0000000000000000
};

uint8_t cursorX, cursorY, line;

void typingSetup() {
    cursorX = cursorY = line = 0;
}

void typingLoop() {
    img[cursorY * WIDTH + cursorX] = // If bit set, "type" random char
        ((codeBits[line] <<< cursorX) & 0x8000) ? random(32, 128) : 0;

    cursorX++;
    if(!(uint16_t)(codeBits[line] <<< cursorX)) { // End of line reached?
        cursorX = 0;
        if(cursorY >= HEIGHT-1) { // Cursor on last line?
            uint8_t y;

```

```

        for(y=0; y<HEIGHT-1; y++) // Move img[] buffer up one line
            memcpy(&img[y * WIDTH], &img[(y+1) * WIDTH], WIDTH);
        memset(&img[y * WIDTH], 0, WIDTH); // Clear last line
    } else cursorY++;
    if(++line >= (sizeof(codeBits) / sizeof(codeBits[0]))) line = 0;
}
img[cursorY * WIDTH + cursorX] = 255; // Draw cursor in new position
}

// MATRIX EFFECT -----
// Inspired by "The Matrix" coding effect -- 'raindrops' travel down the
// screen, their 'tails' twinkle slightly and fade out.

#define N_DROPS 15
struct {
    int8_t x, y; // Position of raindrop 'head'
    uint8_t len; // Length of raindrop 'tail' (not incl head)
} drop[N_DROPS];

void matrixRandomizeDrop(uint8_t i) {
    drop[i].x = random(WIDTH);
    drop[i].y = random(-18, 0);
    drop[i].len = random(9, 18);
}

void matrixSetup() {
    for(uint8_t i=0; i<N_DROPS; i++) matrixRandomizeDrop(i);
}

void matrixLoop() {
    uint8_t i, j;
    int8_t y;

    for(i=0; i<N_DROPS; i++) { // For each raindrop...
        // If head is onscreen, overwrite w/random brightness 20-80
        if((drop[i].y >= 0) && (drop[i].y < HEIGHT))
            img[drop[i].y * WIDTH + drop[i].x] = random(20, 80);
        // Move pos. down by one. If completely offscreen (incl tail), make anew
        if(++drop[i].y - drop[i].len >= HEIGHT) matrixRandomizeDrop(i);
        for(j=0; j<drop[i].len; j++) { // For each pixel in drop's tail...
            y = drop[i].y - drop[i].len + j; // Pixel Y coord
            if((y >= 0) && (y < HEIGHT)) { // On screen?
                // Make 4 pixels at end of tail fade out. For other tail pixels,
                // there's a 1/10 chance of random brightness change 20-80
                if(j < 4)
                    img[y * WIDTH + drop[i].x] /= 2;
                else if(!random(10)) img[y * WIDTH + drop[i].x] = random(20, 80);
            }
        }
        if((drop[i].y >= 0) && (drop[i].y < HEIGHT)) // If head is
        onscreen,
            img[drop[i].y * WIDTH + drop[i].x] = 255; // draw w/255 brightness
    }
}

// CONWAY'S GAME OF LIFE -----
// The rules: if cell at (x,y) is currently populated, it stays populated
// if it has 2 or 3 populated neighbors, else is cleared. If cell at (x,y)
// is currently empty, populate it if 3 neighbors.

void lifeSetup() { // Fill bitmap with random data
    for(uint8_t i=0; i<sizeof(bitmap); i++) bitmap[i] = random(256);
}

void lifeLoop() {
    static const int8_t xo[] = { -1, 0, 1, -1, 1, -1, 0, 1 },
                        yo[] = { -1, -1, -1, 0, 0, 1, 1, 1 };
    int8_t
        x, y;

```

```

uint8_t          i, n;

// Modify img[] based on old contents (dimmed) + new bitmap
for(i=y=0; y<HEIGHT; y++) {
    for(x=0; x<WIDTH; x++, i++) {
        if(bitmapGetPixel(x, y)) img[i] = 255;
        else if(img[i] > 28)      img[i] -= 28;
        else                      img[i] = 0;
    }
}

// Generate new bitmap (next frame) based on img[] contents + rules
memset(bitmap, 0, sizeof(bitmap));
for(y=0; y<HEIGHT; y++) {
    for(x=0; x<WIDTH; x++) {
        for(i=n=0; (i < sizeof(xo)) && (n < 4); i++)
            n += (img[((y+yo[i])%HEIGHT) * WIDTH + ((x+xo[i])%WIDTH)] == 255);
        if((n == 3) || ((n == 2) && (img[y * WIDTH + x] == 255)))
            bitmapSetPixel(x, y);
    }
}

// Every 32 frames, populate a random cell so animation doesn't stagnate
if(!(frame & 0x1F)) bitmapSetPixel(random(WIDTH), random(HEIGHT));
}

// MORE GLOBAL STUFF - ANIMATION STATES -----
struct { // For each of the animation modes...
    void (*setup)(void); // Animation setup func (run once on mode change)
    void (*loop)(void);  // Animation loop func (renders one frame)
    uint8_t maxRunTime;   // Animation run time in seconds
    uint8_t fps;          // Frames-per-second for this effect
} anim[] = {
    NULL, cursorLoop, 3, 4,
    typingSetup, typingLoop, 15, 15,
    lifeSetup, lifeLoop, 12, 30,
    matrixSetup, matrixLoop, 15, 10,
};

uint8_t seq[] = { 0, 1, 0, 2, 0, 3 }, // Sequence of animation modes
        idx = sizeof(seq) - 1;        // Current position in seq[]
uint32_t modeStartTime = 0x7FFFFFFF;  // micros() when current mode started

// SETUP - RUNS ONCE AT PROGRAM START -----
void setup() {
    uint16_t i;
    uint8_t p, bytes;

    randomSeed(analogRead(A0)); // Randomize w/unused analog pin
    Wire.begin();               // Initialize I2C
    Wire.setClock(400000L);     // 400 KHz I2C = faster updates

    // Initialize IS31FL3731 directly (no library)
    pageSelect(0x0B);           // Access the Function Registers
    writeRegister(0);           // Starting from first...
    for(i=0; i<13; i++) Wire.write(10 == i); // Clear all except Shutdown
    Wire.endTransmission();
    for(p=0; p<2; p++) {        // For each page used (0 & 1)...
        pageSelect(p);          // Access the Frame Registers
        for(bytes=i=0; i<180; i++) { // For each register...
            if(!bytes) bytes = writeRegister(i); // Buf empty? Start xfer @ reg i
            Wire.write(0xFF * (i < 18)); // 0-17 = enable, 18+ = blink+PWM
            if(++bytes >= SERIAL_BUFFER_SIZE) bytes = Wire.endTransmission();
        }
        if(bytes) Wire.endTransmission(); // Write any data left in buffer
    }
}

```



```

}

for(i=0; i<256; i++) // Initialize gamma-correction table:
    gamma8[i] = (uint8_t)(pow(((float)i / 255.0), GAMMA) * 255.0 + 0.5);
}

// LOOP - RUNS ONCE PER FRAME OF ANIMATION -----

uint32_t prevTime = 0xFFFFFFFF; // Used for frame-to-frame animation timing
uint32_t frameUsec = 0L;        // Frame interval in microseconds

void loop() {
    // Wait for FPS interval to elapse (this approach is more consistent than
    // delay() as the animation rendering itself takes indeterminate time).
    uint32_t t;
    while(((t = micros()) - prevTime) < frameUsec);
    prevTime = t;

    // Display frame rendered on prior pass. This is done immediately
    // after the FPS sync (rather than after rendering) to ensure more
    // uniform animation timing.
    pageSelect(0x0B); // Function registers
    writeRegister(0x01); // Picture Display reg
    Wire.write(page); // Page #
    Wire.endTransmission();
    page ^= 1; // Flip front/back buffer index

    anim[seq[idx]].loop(); // Render next frame
    frameUsec = 1000000L / anim[seq[idx]].fps; // Frame hold time

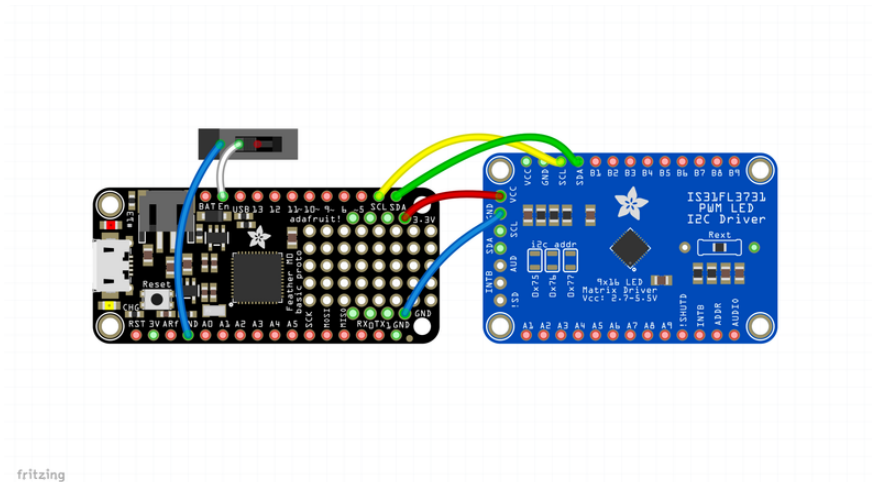
    // Write img[] array to matrix thru gamma correction table
    uint8_t i, bytes; // Pixel #, Wire buffer counter
    pageSelect(page); // Select background buffer
    for(bytes=i=0; i<WIDTH*HEIGHT; i++) {
        if(!bytes) bytes = writeRegister(0x24 + i);
        Wire.write(gamma8[img[i]]);
        if(++bytes >= SERIAL_BUFFER_SIZE) bytes = Wire.endTransmission();
    }
    if(bytes) Wire.endTransmission();

    // Time for new mode?
    if((t - modeStartTime) > (anim[seq[idx]].maxRunTime * 1000000L)) {
        if(++idx >= sizeof(seq)) idx = 0;
        memset(img, 0, sizeof(img));
        if(anim[seq[idx]].setup) anim[seq[idx]].setup();
        modeStartTime = t;
        frame = 0;
    } else frame++;
}

```

---

# Circuit Diagram



## Wired Connections

The circuit diagram above shows how the components will be wired together. This won't be 100% exact in the actual circuit but it's a very close approximation.

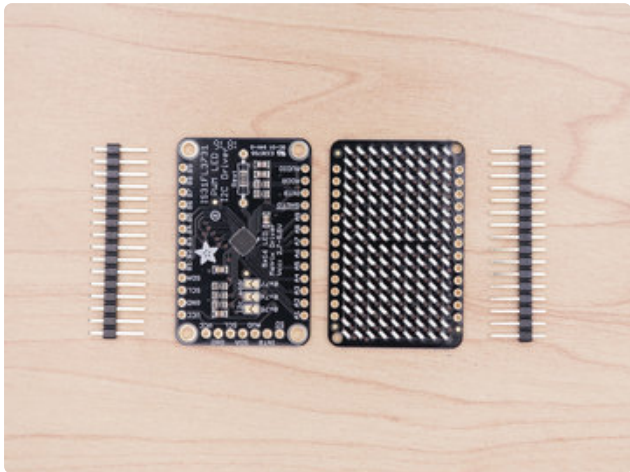
- Slide switch to EN pin on Adafruit Feather M0
- Slide switch to GND pin on Adafruit Feather M0
- SCL from LED Charlieplexed Matrix to SCL on Adafruit Feather M0
- SDA from LED Charlieplexed Matrix to SDA on Adafruit Feather M0
- VCC from LED Charlieplexed Matrix to 3.3V on Adafruit Feather M0
- GND from LED Charlieplexed Matrix to GND on Adafruit Feather M0

## Battery Power

The circuit will be powered by a 3.7V 500mAh Lithium ion battery via JST connection. The battery plugs directly into the Adafruit Feather M0, which also has a built-in charging circuit.

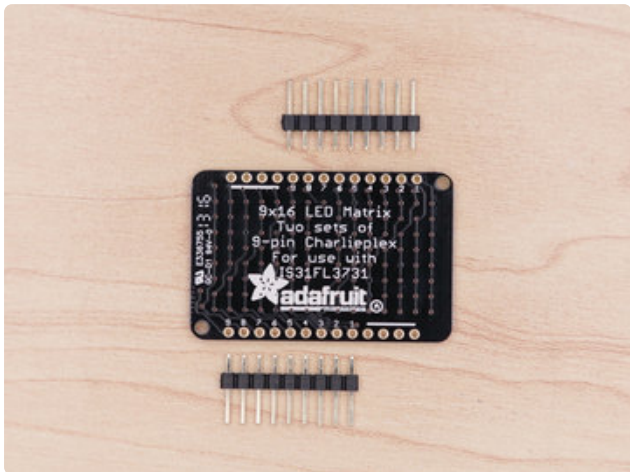
---

# LED Matrix



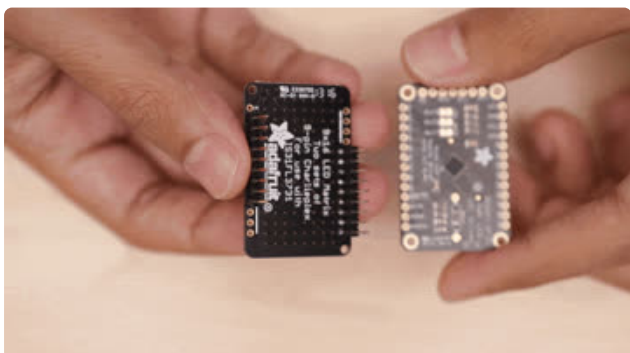
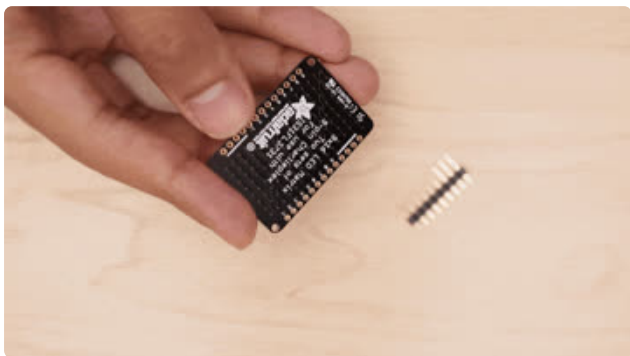
## Charlieplexed Driver and LED Matrix

The charlieplexed driver and 16x9 LED matrix are two separate PCBs that will need to be assembled together. We'll do this that soldering two header pins that shipped with the boards.



## Charlieplexed Driver Header Pins

We aren't required to solder the whole strip of header pins, we can just use what's necessary. Use the labeling on the back of the 9x16 LED matrix PCB to reference the necessary amount of pins. You can break the strip apart using flush cutters (scissors in a pinch!).



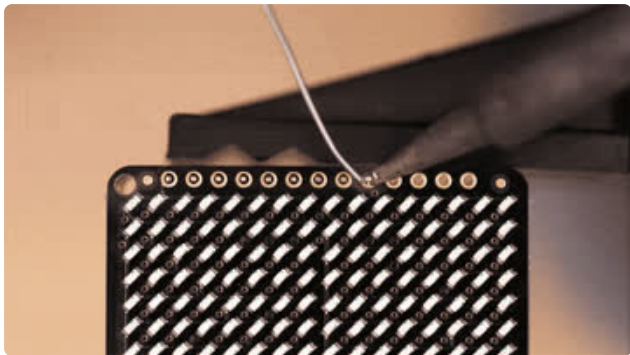
## Mount Parts

Pro Tip: Insert the short ends of the header pins into the LED matrix first, then lay the driver over the long ends.



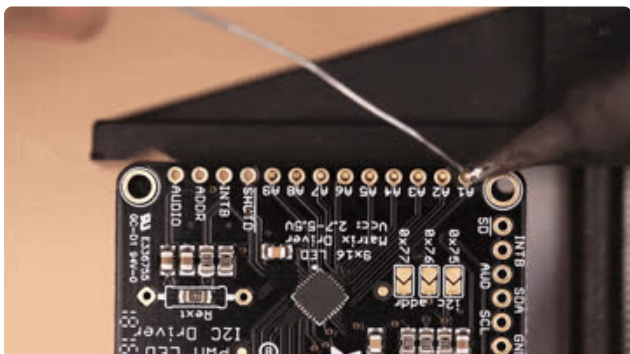
## Side Profile

The assembly should look like this, with a nice and low profile. Once seated, keep them together until we solder the pins.



## Solder In Place

I recommend using a panavise jr. or helping third hands to secure the PCBs in place while soldering.



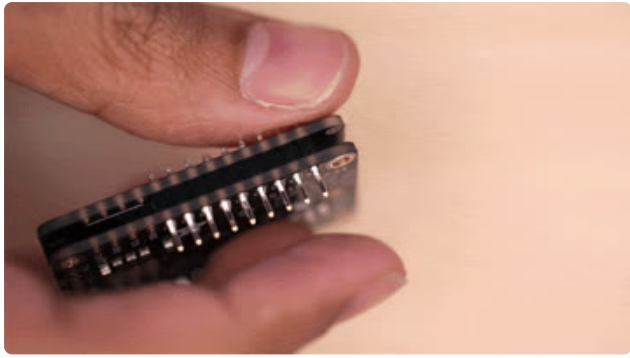
Pro Tip: Secure the driver to the panavise' jaws so the LED matrix can rest on the header pins.

Solder all 18 pins in place.



## Charlieplexed Matrix Assembly

Thoroughly inspect the assembly for any shorts – Ensure each pin has a sufficient amount of solder.



## Trim Header Pins

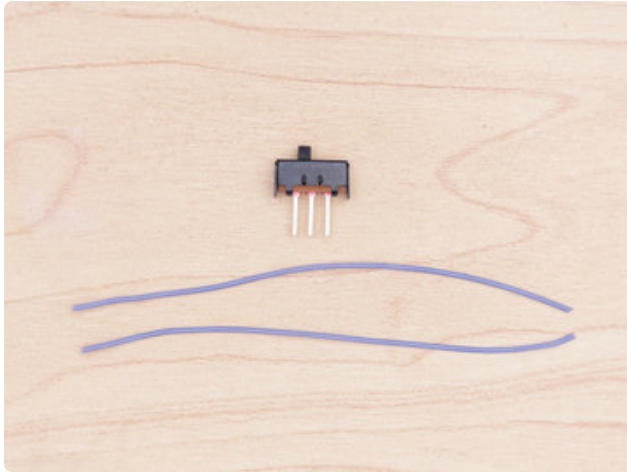
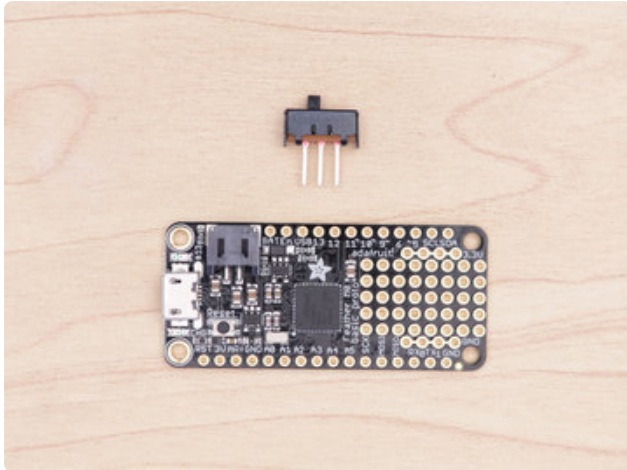
You can trim the access header pins using a pair of flush cutters.



Pro Tip: Hold the tip of your index finger over the access pin while trimming to avoid them from flying across the room.

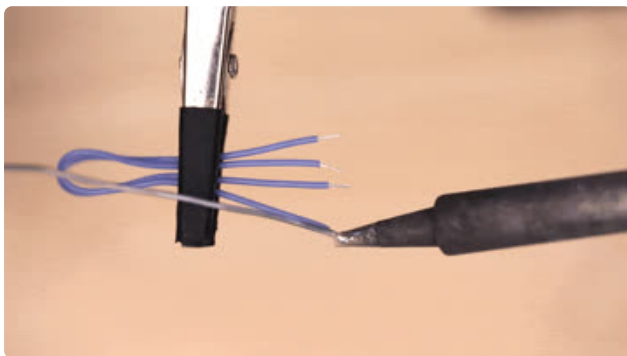
---

# Wiring



## Switch for Feather

To make it easy to turn the circuit on/off, we'll need to connect a switch. This breadboard friendly SPDT slide switch is perfect for the task. We'll need two pieces of short wires to solder onto the leads of the switch. I recommend using [30AWG silicone-coated wires](http://adafru.it/2051) (<http://adafru.it/2051>) for maximum efficiency.



## Tin Wires for Slide Switch

To make it easier to connect the wires to the leads of the slide switch, we can strip the wires and tin the tips with a small amount of solder.

Pro Tips:

1. Add pieces of heat shrink tubing over the teeth of helping third hands to avoid kinking wires.
2. Group wires together in one arm to tin quicker.



## Connect Wires to Slide Switch

Solder the wires to the leads on the slide switch. Only two leads are necessary, one in the middle and one on the outside (doesn't matter which).

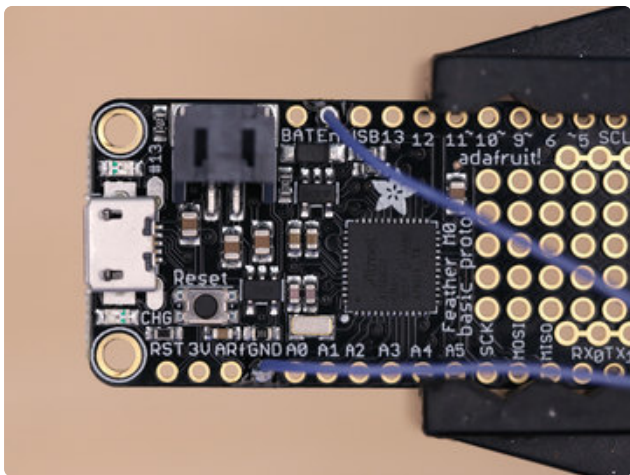


Pro Tip:

1. Snip one of the leads on the far left/right (but not middle).
2. Use the tip of the soldering iron to heat up the lead and apply a small amount of solder to tin them.

## Connect Slide Switch to Feather M0

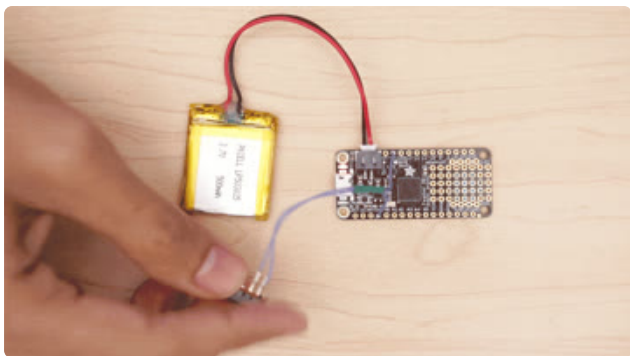
Now we can connect the slide switch to the Adafruit Feather M0. Solder one wire to the **EN** pin and the other to the **GND** pin on the Adafruit Feather M0.

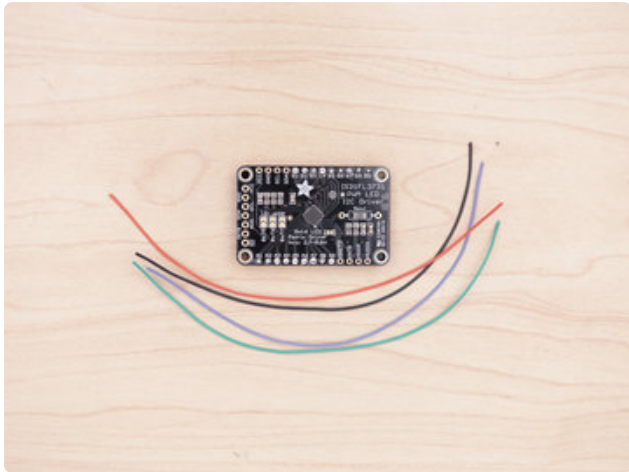


Pro Tip: Tin the pins with a small amount of solder to make it easier to insert wires.

## Test Switch

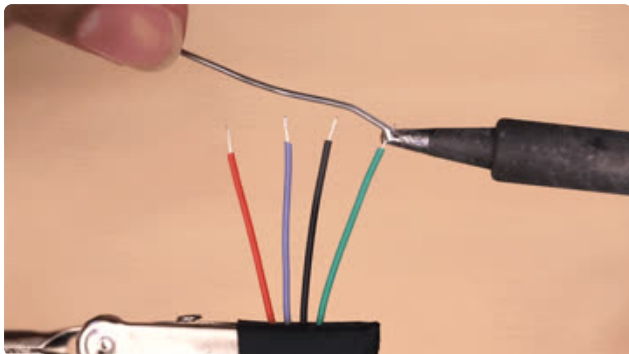
With the slide switch wired to the Adafruit Feather M0, we can then test it out to ensure its working properly. Plug in the male JST connector from the battery and see if the red LED turns on. Sliding the switch should make it turn on/off.





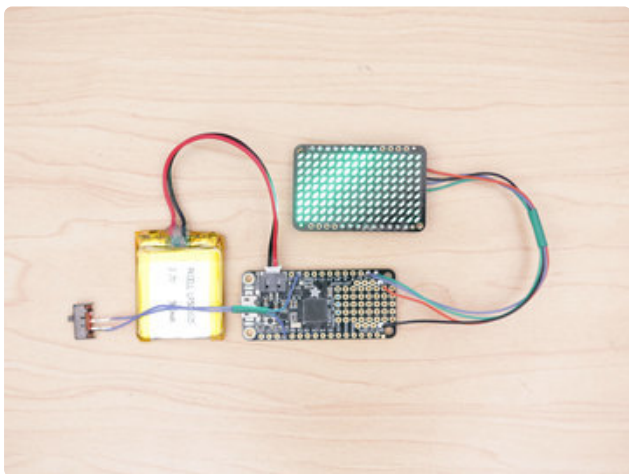
## Wires for Charlieplexed Matrix

We'll need four pieces of wire to connect the charlieplexed LED matrix to the Adafruit Feather M0. These can be about 6cm in length. Then, strip and tin the tips of each wire.



Pro Tip:

1. Use different colored wires to easily tell them apart.



## Final Circuit Test

With all of our connections wired up, we can now test out the circuit! The code should already be uploaded to the Adafruit Feather M0.

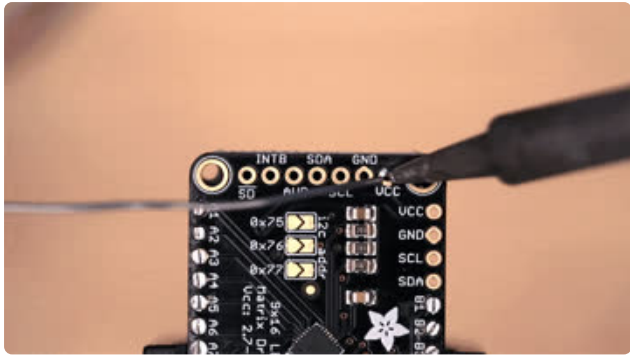


## Heat Shrink Tubing

You can use a piece of heat shrink tubing to keep the wires grouped together. This makes wiring management easier during assembly.

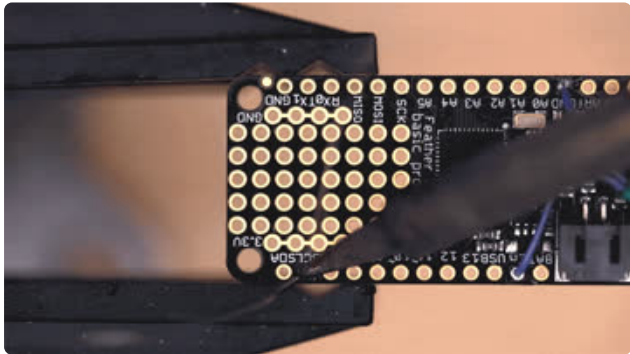
Pro Tip: If you don't have a lighter handy, use the heating element (NOT TIP) of the soldering iron to shrink the tubing.





## Connect Wires to Charlieplexed Matrix

Secure the charlieplexed LED matrix assembly and connect the four wires to the **SDA**, **SLC**, **VCC** and **GND** pins by soldering them in place.



## Connect Charlieplexed Matrix to Feather

Now we can connect the four wires from the charlieplexed LED matrix to the Adafruit Feather M0. Connect VCC to 3.3V, GND to GND, SDA to SDA and SLC to SLC.

# Assembly



## Glue Keys to Cover

The faux keyboard and keypad needs to be glued to the keyboard cover. You can use super glue or E6000. Allow the parts to dry before working further.



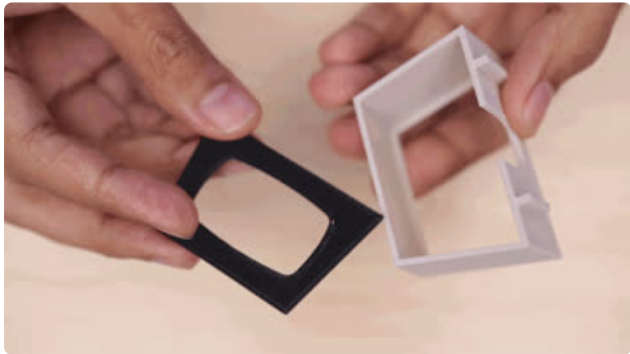
## Glue KeyCover to Frame

The keyboard cover need to be secured to the lower part of the frame. I recommend apply small amounts of glue to the edges and then laying the keyboard cover over top. Again, allow the parts to dry before continuing



## Glue displayBase to Frame

The display base needs to be secured to the upper part of the frame. Use the same method like we did for the keyboard cover.



## Install displayFace

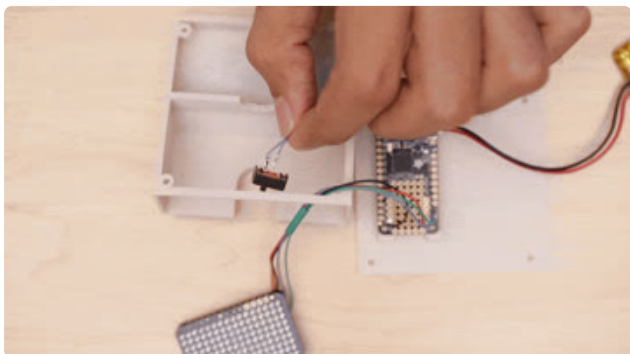
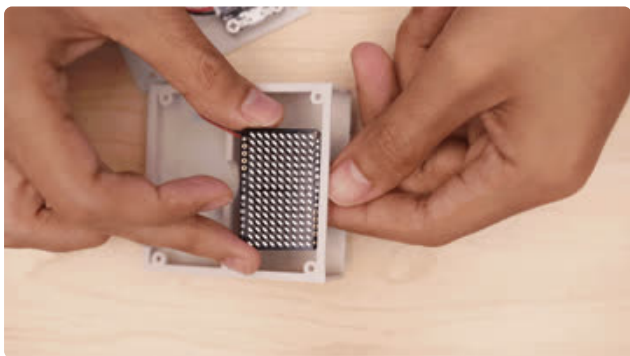
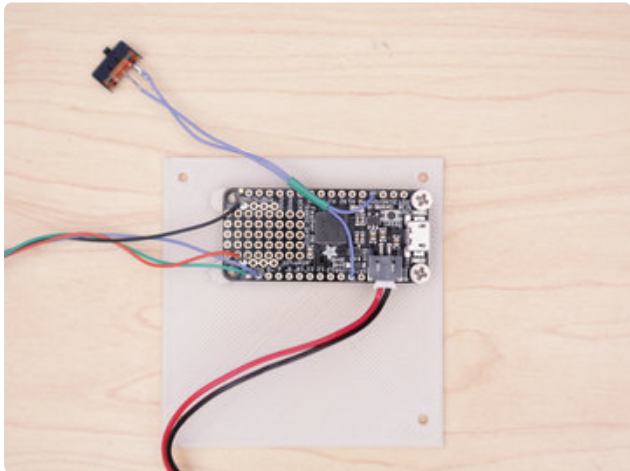
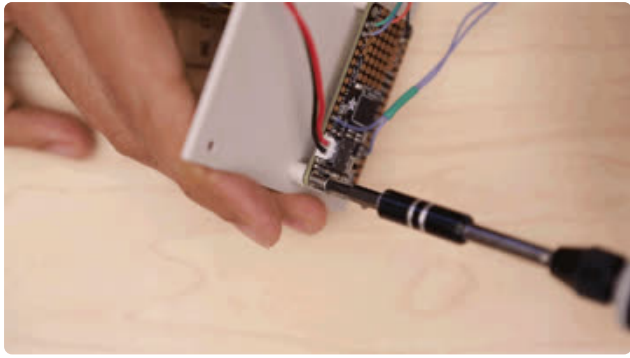
The face of the display can be inserted through the large opening of the display frame. Glue isn't required, but it will help secure the parts together.



## Tap Mounting Holes

We'll secure the Adafruit Feather M0 board to the 3d printed cover using #4-40 (M3) screws.

Pro Tip: Tap the two mounting holes (closest to the microUSB connector) by inserting a single screw and slowly fastening. You only need to make 2-3 full rotations to create the threads. Try to be as straight as possible.

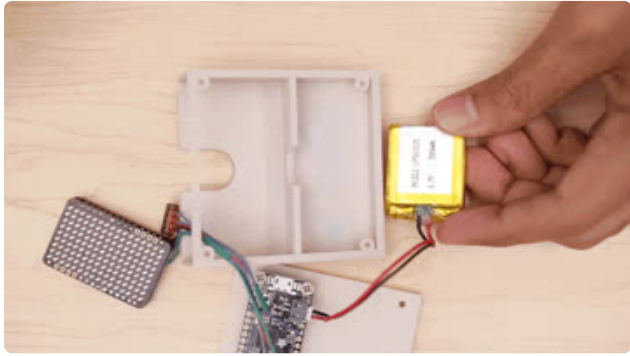


## Mount Feather to Cover

Insert both screws into the two mounting holes until the tip of the screw barely protudes from the bottom. Then, lay the Adafruit Feather M0 over the **frame bottom cover** part with the two mounting screws going into the holes of the two standoffs. Hold the Adafruit Feather M0 in place while fastening the two screws.

## Install Components into Frame

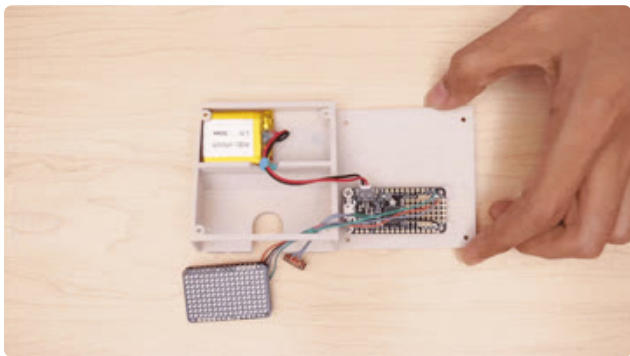
We'll need to route our charlieplexed matrix and slide switch through the frame in order to properly assemble the parts.



## Install Battery into Frame

Now is a good time to mount the battery to the inside of the frame enclosure. Ensure the wire from the lipo battery is positioned through the arch near the center.

**Pro Tip:** Use a piece of mounting tack to secure the lipo battery to the bottom of the keyboard cover.



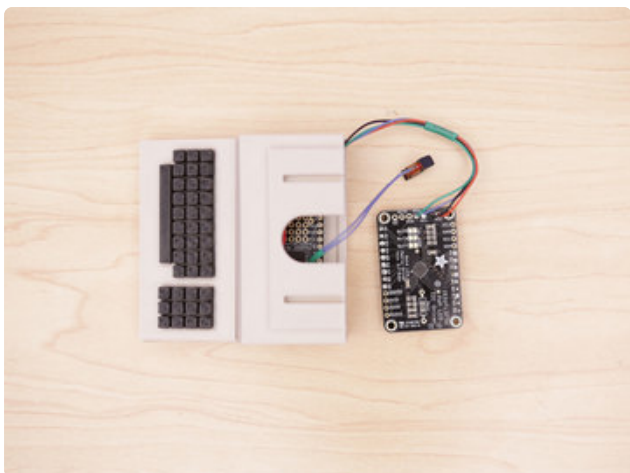
## Install Cover to Frame

The bottom cover of the frame will be secured with machine screws. Carefully lay the cover over the frame and line up the holes in the corners of both pieces.



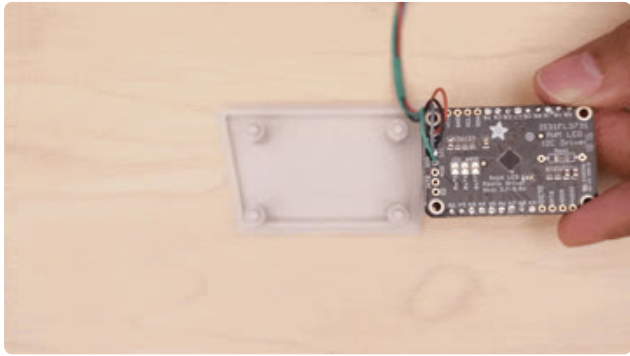
## Fasten Screws

Insert a #2-56 (M2.5) screw into one of the holes on the bottom of the cover. Fasten the screw until it's flush with the surface. Repeat this process for the remaining holes.



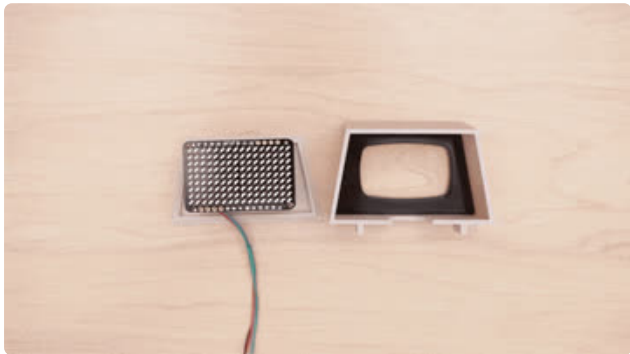
## Checkpoint

OK, now's a good time to take a break and review our work.



## Install Charlieplexed Matrix into Display Cover

The PCB of the charlieplexed LED matrix driver has four mount holes. The four standoffs on the display cover has nubs that fit snugly into the holes. Route the wires so they're posited in the center. Lay the PCB over the standoffs and press the PCB down until the nubs fit into the holes.



## Install Charlieplexed Matrix into Display Frame

The display cover snaps into the back opening of the display frame. The inner lip of the display cover has a tight tolerance.



## Install Switch to Back Cover

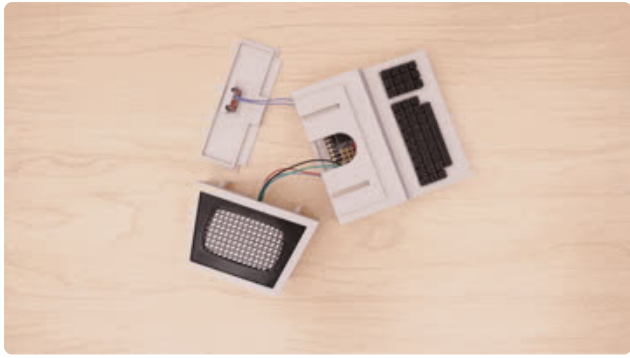
You can insert the slide switch into the little platform in the center of the back cover. Glue it in place if it has a loose tolerance. You should be able to slide the switch from the other side.



## Assembly Checkpoint 2

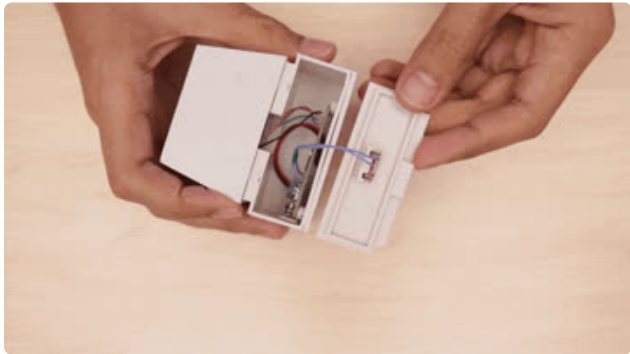
Here's our build so far. All that's left to do is fit the back cover and display into place.





## Install Display to Frame

The display has two keys that fit into the slots on the top of the display base. Press the two parts together until their flush with each other.



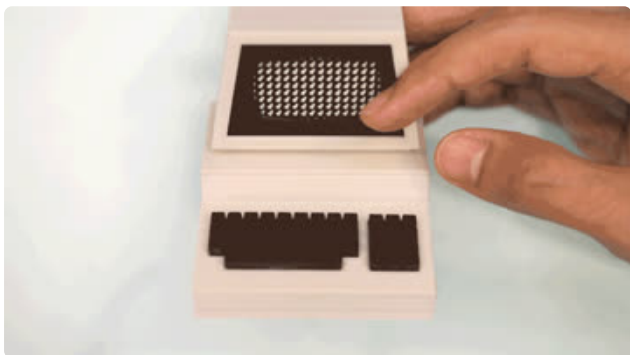
## Install Back Cover to Frame

The back cover that houses the slide switch snaps into the frame. The inner lip will have tight tolerance and should hold in place.



## Crop Sticker Artwork

If you (2D) printed out the label, now is a good time to crop it using an x-acto knife or similar.



## Apply Sticker

Peel off the back to reveal the adhesive and carefully line up the sticker and press to apply. And that's it! Now we have our very own PET ATSAMD21 Feather :-)