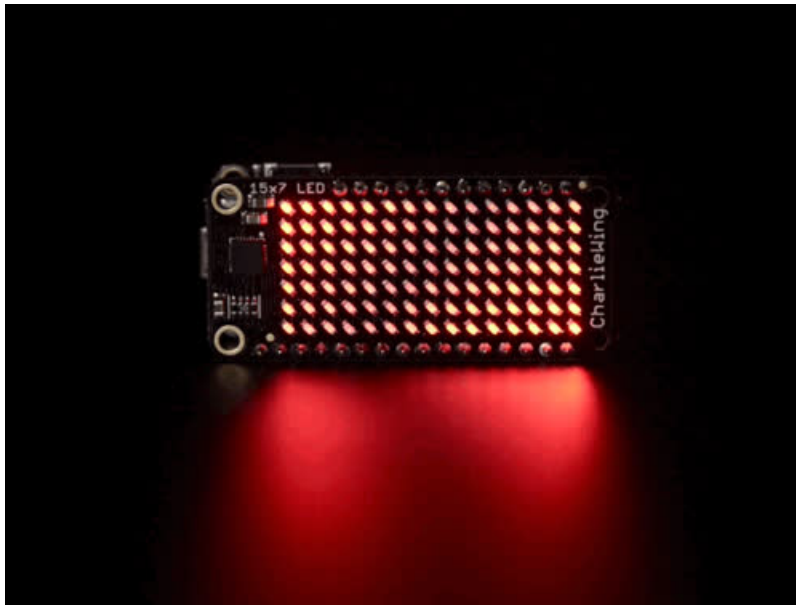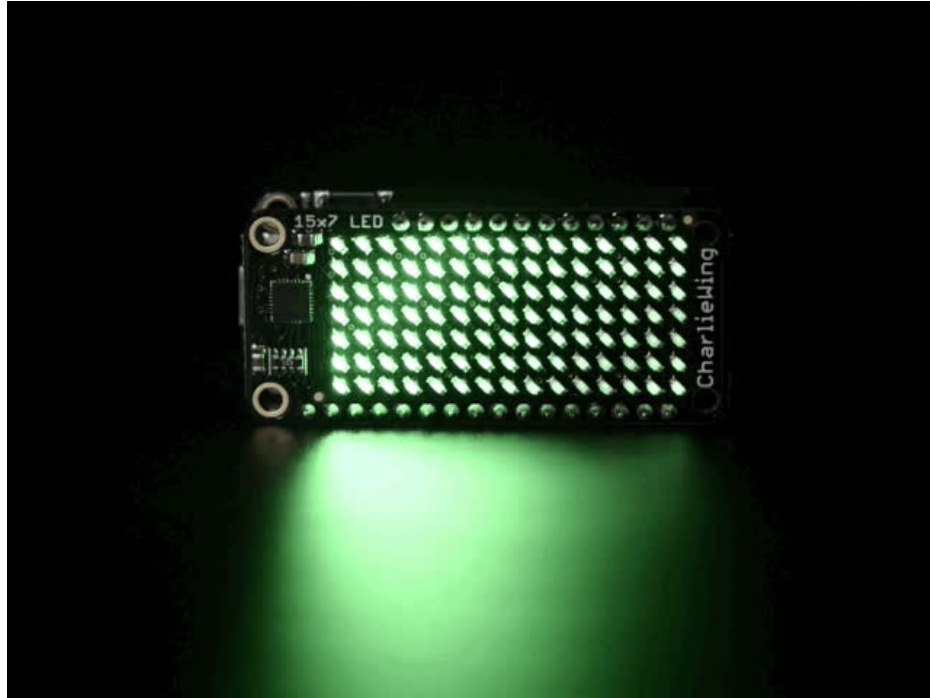# Adafruit 15x7 CharliePlex FeatherWing
Created by lady ada


Last updated on 2019-05-12 07:13:28 PM UTC

Overview

You wont be able to look away from the mesmerizing patterns created by this **Adafruit 15x7 CharliePlex LED Matrix Display FeatherWing**. This 15x7 LED display can be paired with with any of our Feather boards (https://adafru.it/nEt) for a beautiful, bright grid of 105 charlieplexed LEDs. It even comes with a built-in charlieplex driver that is run over I2C.
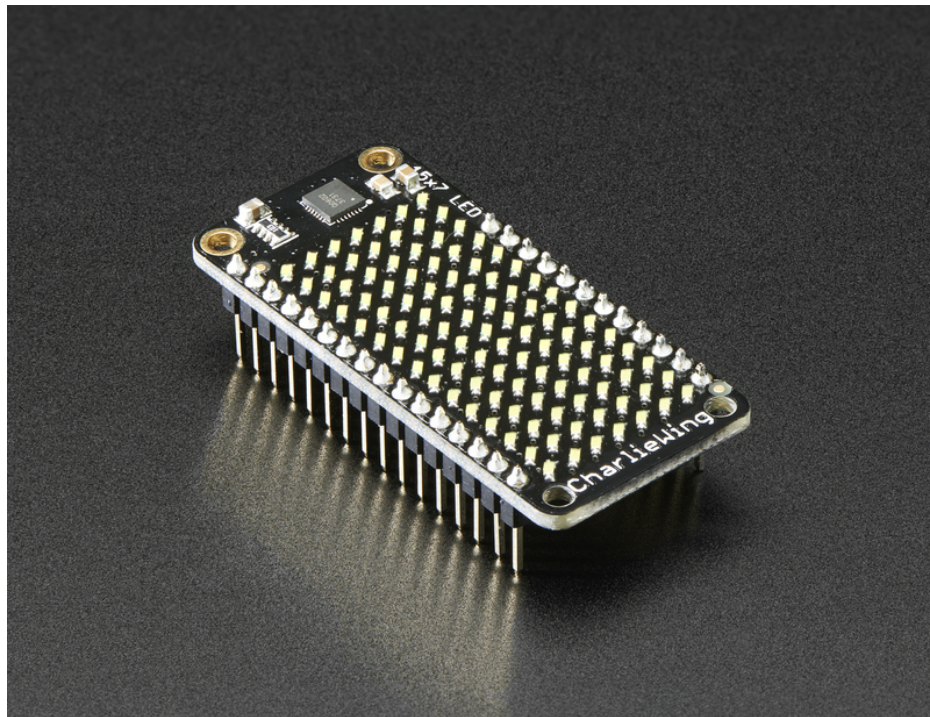
We carry these FeatherWings in five vivid colors (http://adafru.it/29732520)

What is particularly nice about this Wing is the I2C LED driver chip has the ability to PWM each individual LED in a 15x7

grid so you can have beautiful LED lighting effects, without a lot of pin twiddling. Simply tell the chip which LED on the grid you want lit, and what brightness and it's all taken care of for you. You get 8-bit (256 level) dimming for each individual LED.

The IS31FL3731 is a nice little chip - and it runs happily over 3.3V power. Inside is enough RAM for 8 separate frames of display memory so you can set up multiple frames of an animation and flip them to be displayed with a single command. Since it uses I2C, it takes up only the SDA/SCL pins on your Feather and can share those pins with other I2C devices and sensors.
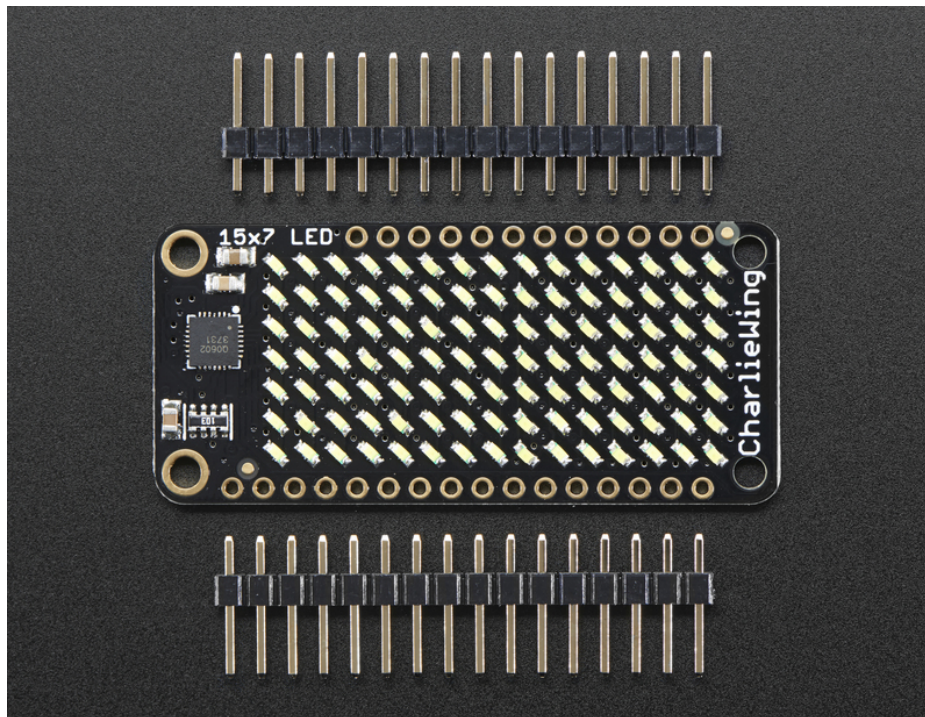
Our graphics library makes it easy to draw shapes, text and bitmaps for some strikingly beautiful LED displays.
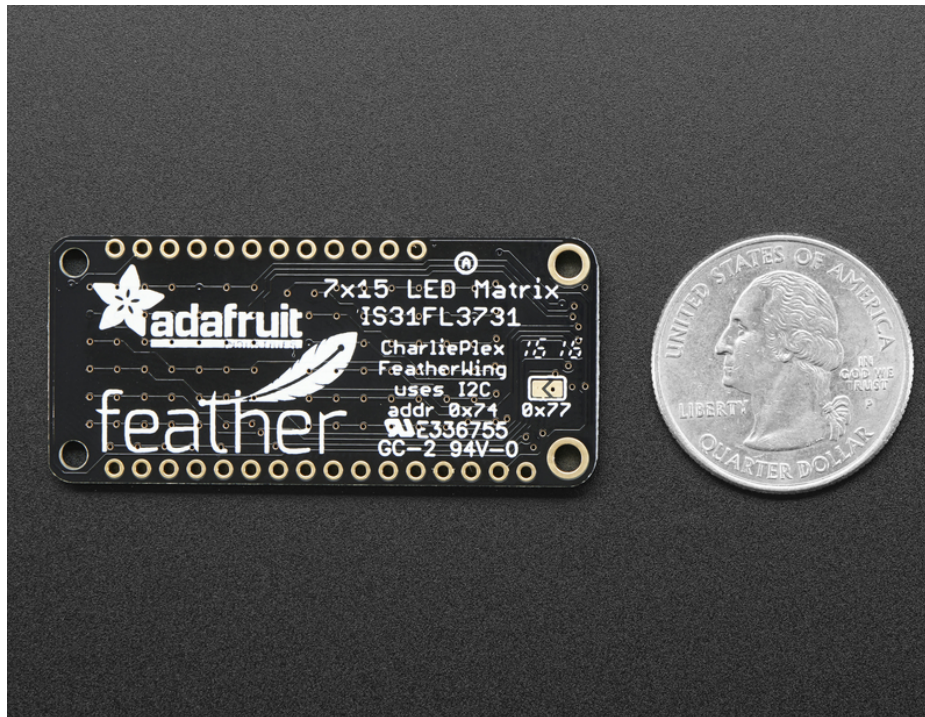
Pinouts
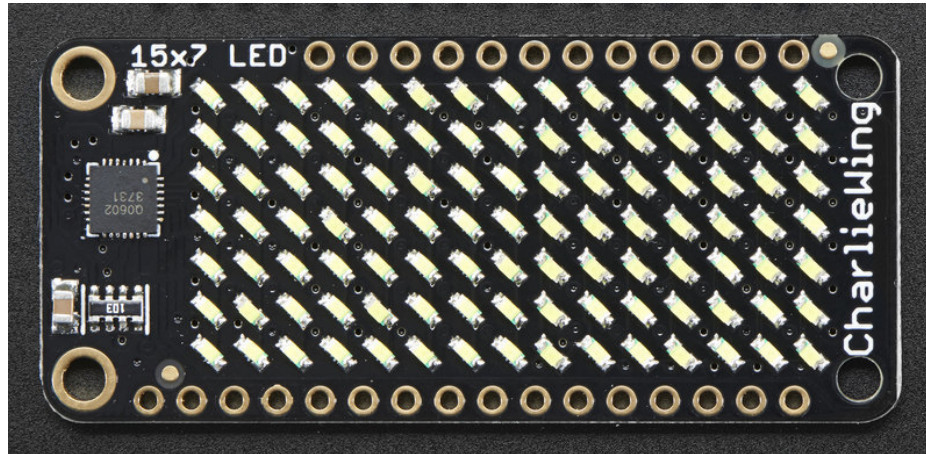
The 7x15 backpack makes it really easy to add a 105-LED display

The LEDs themselves do not connect to the Feather. Instead, a matrix driver chip (IS31FL3731) does the multiplexing for you. The Feather simply sends i2c commands to the chip to tell it what LEDs to light up & how bright, and it is handled for you. This takes a lot of the work and pin-requirements off the Feather. Since it uses only I2C for control, it works with any Feather and can share the I2C pins for other sensors or displays.

Power Pins
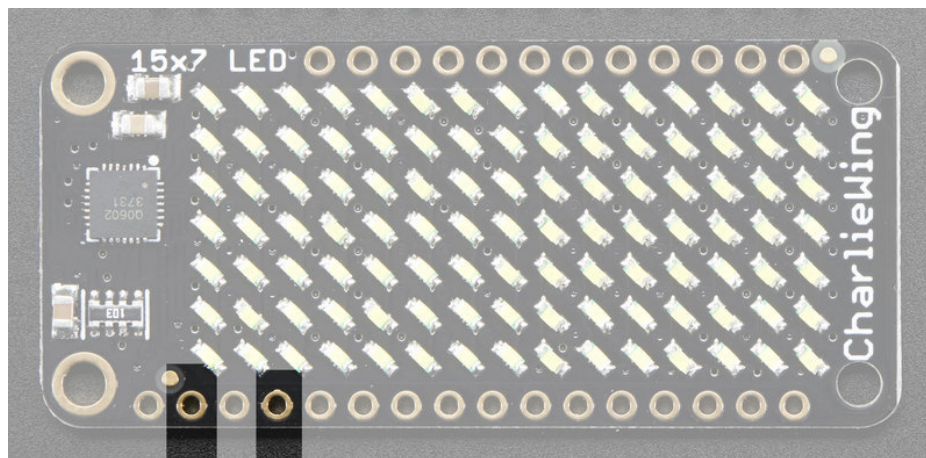
The LED matrix uses only the **3V** (second from left) and **GND** (fourth from left) pins for power and logic. Current draw depends on how many LEDs are lit but you can approximate it as about 40mA for most uses.

Note that the 3.3V power supply is a tiny bit lower than the forward voltage for the pure green, blue and white LED matrices but we didn't find any significant degredation in brightness. Really, they're still very bright.

I2C pins
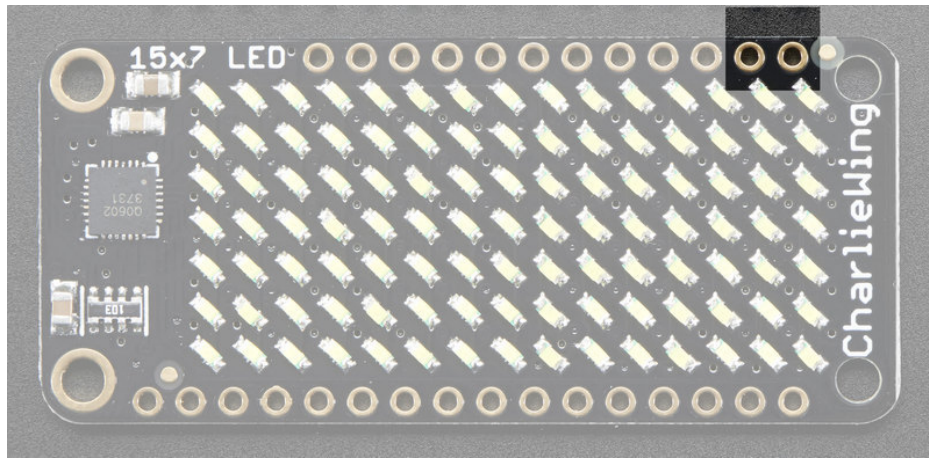
All LED control is done over I2C using the IS31FL3731 interface library. This means **SDA** (rightmost) and **SCL** (to the left of SDA) must be connected, see above for those pins.

The default address is **0x74** but you can change the address to **0x77** by bridging solder onto the address jumper
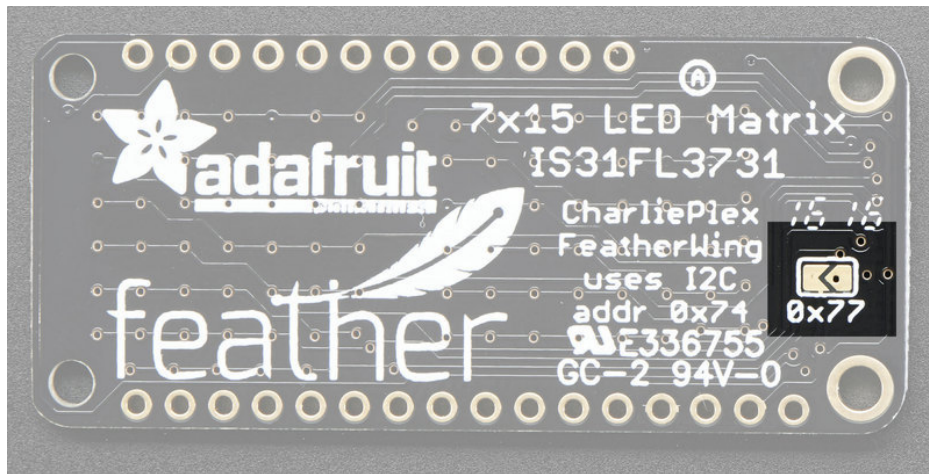
Arduino Usage

Download Adafruit_IS31FL3731 library

To begin reading sensor data, you will need to download Adafruit_IS31FL3731 from our github repository (https://adafru.it/lWA). You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

<div style="background-color:green; color:white; text-align:center;">https://adafru.it/lWB</div>

https://adafru.it/lWB

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 **arduinosketchfolder/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)

<div style="background-color:green; color:white; text-align:center;">https://adafru.it/cBB</div>

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 *arduinosketchfolder*/**libraries/** folder like you did with the IS31FL library

Load Demo

Open up **File->Examples->Adafruit_IS31FL3731->featherswirldemo** and upload to your Feather + CharliePlex Matrix Feather

<div style="background-color:#a5a520; color:white;">⚠   You must update the code before uploading!</div>

At the top of the sketch, find the code that says

```
// If you're using the full breakout...
Adafruit_IS31FL3731 ledmatrix = Adafruit_IS31FL3731();
// If you're using the FeatherWing version
//Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
```

and uncomment the last line and recomment the second line so you have:

```
// If you're using the full breakout...
//Adafruit_IS31FL3731 ledmatrix = Adafruit_IS31FL3731();
// If you're using the FeatherWing version
Adafruit_IS31FL3731_Wing ledmatrix = Adafruit_IS31FL3731_Wing();
```
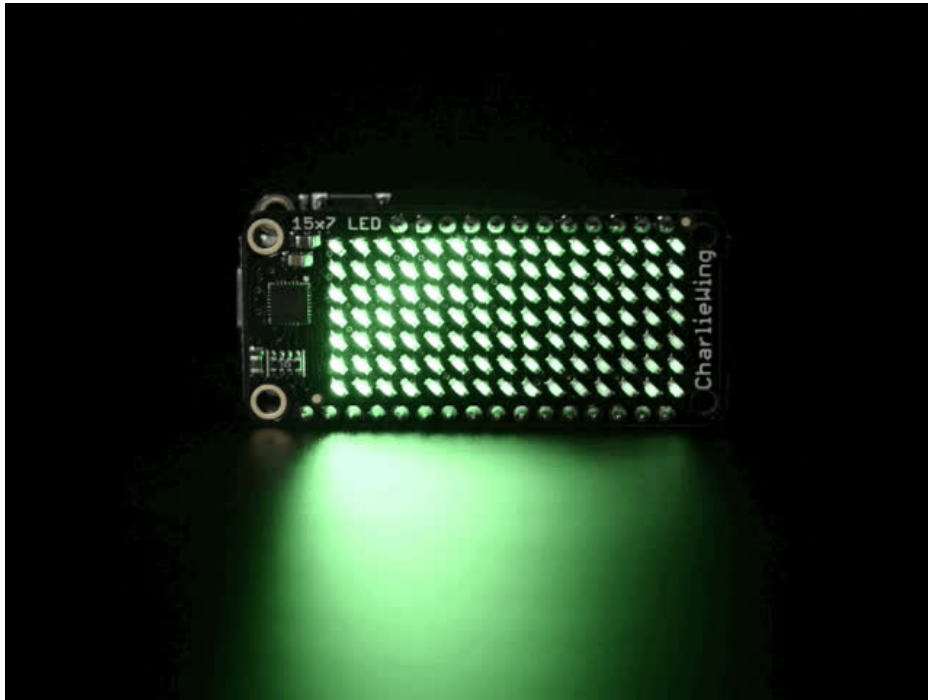

Upload to your Arduino, you'll see the LED display swirl different brightnesses!

Library Reference

Now that you have the demo working, you can control the matrix directly.

Initialize

Start by creating a new matrix object with something like:

```
Adafruit_IS31FL3731 ledmatrix = Adafruit_IS31FL3731();
```

There's no arguments to the constructor

Then in your setup, call **begin(*address*)** to initialize the driver. Begin() will return false if the matrix was not found, and true if initialization worked out

```
if (! ledmatrix.begin()) {
  Serial.println("IS31 not found");
  while (1);
}
Serial.println("IS31 found!");
```

Drawing

You can then draw to the display. Note that since we write directly to the driver RAM, any pixels 'drawn' will appear immediately.

You can start with **drawPixel(*x, y, brightness*)** where **x** ranges between 0 and 15 inclusive, and **y** ranges between 0 and 8 inclusive. *Brightness* is the PWM of the LED, 0 is off, and 255 is all the way on.

This loop will light up every LED in increasing brightness:

```
int i = 0;
for (uint8_t x=0; x<16; x++) {
  for (uint8_t y=0; y<9; y++) {
    ledmatrix.drawPixel(x, y, i++]);
  }
}
```

Adafruit GFX

Once you get pixels drawing, you can use Adafruit GFX to draw lines, rectangles, circles, text, etc.

The Adafruit_GFX library for Arduino provides a common syntax and set of graphics functions for all of our LED, TFT, LCD and OLED displays. This allows Arduino sketches to easily be adapted between display types with minimal fuss… and any new features, performance improvements and bug fixes will immediately apply across our complete offering of color displays.

Check out our detailed tutorial here http://learn.adafruit.com/adafruit-gfx-graphics-library (https://adafru.it/aPx) It covers the latest and greatest of the GFX library!

Multiple Buffers

The IS31 has 8 full display frame buffers available. By default you draw and display to frame buffer #0

But! If you want to flip thru different images quickly, you can double buffer by writing to one buffer and then telling the IS31 to switch which one is visible.

To set which frame we are **drawing** to, use **setFrame(**_n_**)**
where _n_ ranges from 0 to 7 inclusive

```
ledmatrix.setFrame(frame);
```

Then when you are ready to display it, to set which frame we are **displaying** to, use **displayFrame(**_n_**)**
where _n_ ranges from 0 to 7 inclusive

```
ledmatrix.displayFrame(frame);
```

Python & CircuitPython

It's easy to use the IS31FL3731 Charlieplex breakout, the Charlieplex FeatherWing, and the CharliePlex Bonnet with Python or CircuitPython and the Adafruit CircuitPython IS31FL3731 (https://adafru.it/zIE) module. This module allows you to easily write Python code that does all sorts of fun things with the LED matrix.

You can use CharliePlex LED matrices with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

First wire up a IS31FL3731 breakout to your board exactly as shown on the previous pages for Arduino.
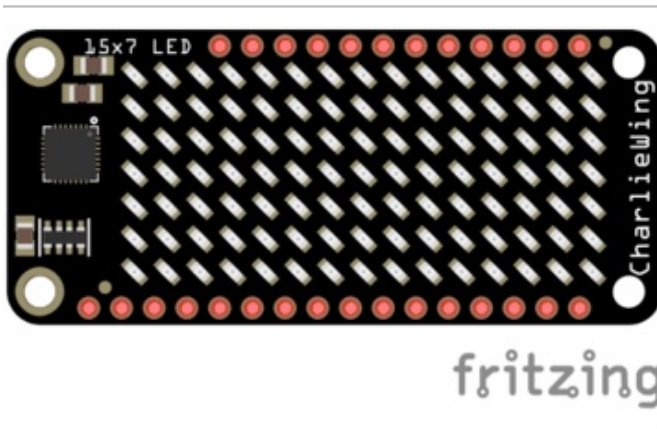
For the FeatherWing, solder on the headers, and attach to the Feather.

Here's an example of wiring a Feather M0 to the breakout with I2C:
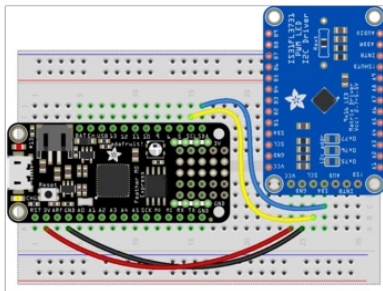
- **Board 3V** to **sensor VCC**
- **Board GND** to **sensor GND**
- **Board SCL** to **sensor SCL**
- **Board SDA** to **sensor SDA**

And here is the CharlieWing on a Feather M4:

- Assemble the CharlieWing by soldering headers onto the board.
- Once assembled, plug it into a Feather!

Python Computer Wiring

Since there's *dozens* of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

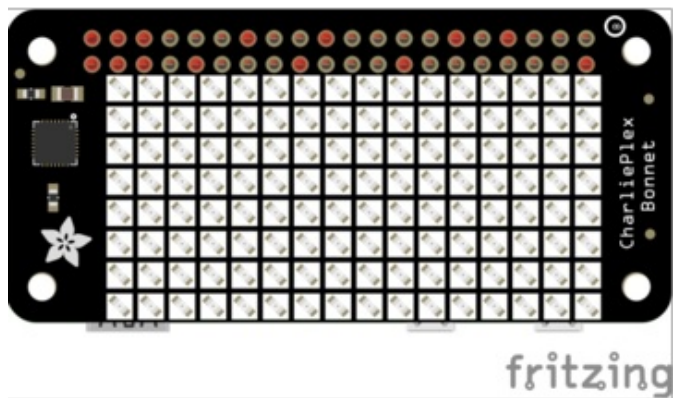For the Bonnet, simply attach the Bonnet to your Raspberry Pi header.

Here's the Raspberry Pi wired to the breakout with I2C:
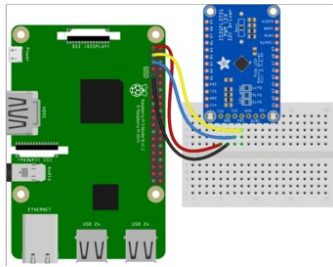
- **Pi 3V3** to **sensor VIN**
- **Pi GND** to **sensor GND**
- **Pi SCL** to **sensor SCL**
- **Pi SDA** to **sensor SDA**

Here is the CharliePlex Bonnet on a Raspberry Pi Zero:

The CharliePlex Bonnet comes fully assembled. Simply plug it into your Raspberry Pi!

CircuitPython Installation of IS31FL3731 Library

You'll need to install the Adafruit CircuitPython IS31FL3731 (https://adafru.it/zIE) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/uap). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- **adafruit_is31fl3731.mpy**
- **adafruit_bus_device**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_is31fl3731.mpy,** and **adafruit_bus_device** files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz) so you are at the CircuitPython >>> prompt.

Python Installation of IS31FL3731 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- sudo pip3 install adafruit-circuitpython-is31fl3731 adafruit-circuitpython-framebuf

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and manipulate the LED matrix from the board's Python REPL.

NOTE: Due to size and design of each CharliePlex matrix form-factor, import and initialisation is different for each. Make sure you're running the correct code for your matrix!

First, run the following code to import the necessary modules and create the I2C object:

```
import board
import busio
import adafruit_is31fl3731
i2c = busio.I2C(board.SCL, board.SDA)
```

If you're using the CharliePlex breakout, initialise it by running the following code:

```
display = adafruit_is31fl3731.Matrix(i2c)
```

If you're using the CharliePlex FeatherWing, run the following code:

```
display = adafruit_is31fl3731.CharlieWing(i2c)
```

If you're using the CharliePlex Bonnet, run the following code:

```
display = adafruit_is31fl3731.CharlieBonnet(i2c)
```

When the display initializes it will go through and clear each frame (there are 8 frames total) of the display. You might see the display momentarily flash and then turn off to a clear no pixel lit image.

You can control all of the board's pixels using the `fill` function. Send to this function a value from `0` to `255` where `0` is every LED pixel turned off and `255` is every LED pixel turned on to maximum brightness. For example to set all the pixels to half their brightness run:

```
display.fill(127)
```
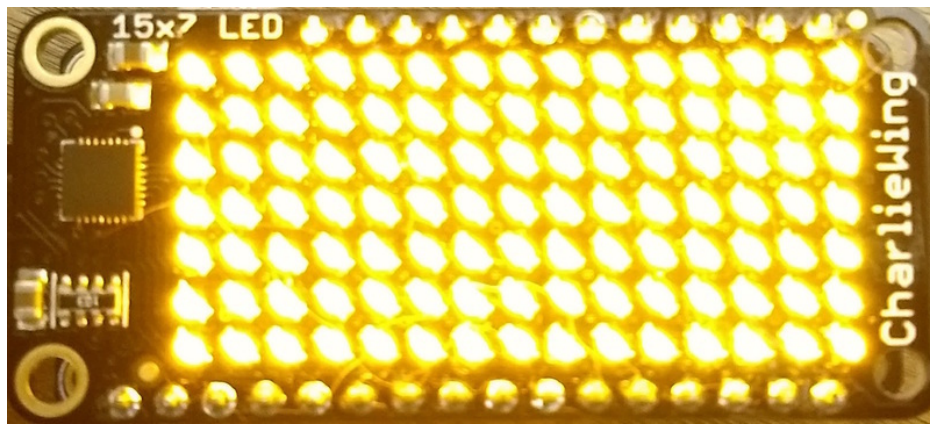
You might notice some buzzing or ringing sounds from the display when all pixels are lit, this is normal as the Charlieplex driver quickly switches LEDs on and off.

If you've used other displays like LED matrices you might notice the Charlieplex module doesn't need to have a show function called to make the changes visible. As soon as you call fill or other display functions the display will update!

You can turn all the pixels off with `fill` set to `0`:

```
display.fill(0)
```
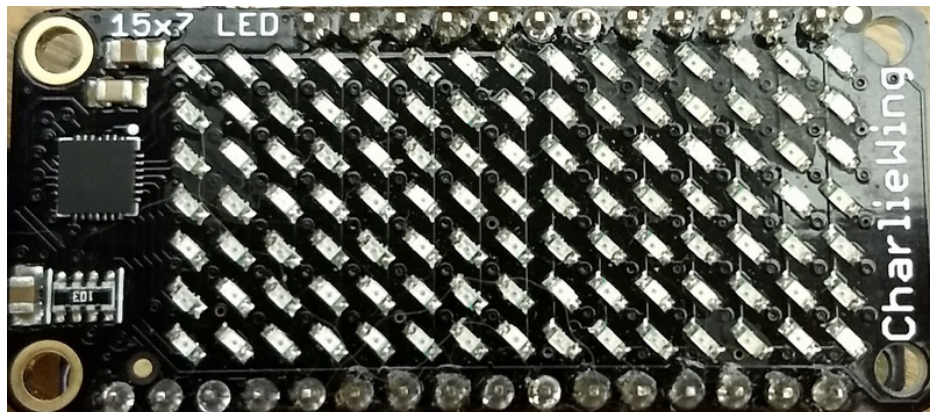

⚠️ Be careful setting all pixels to 255 maximum brightness! This might pull more power than your computer's USB port can provide if you are powering your board over USB. Use an external powers supply or battery when lighting lots of LEDs to max brightness.

Now for some fun! You can set any of the LED pixels using the `pixel` function. This function takes the following parameters:

- **X position** - The location of the horizontal / X pixel position.
- **Y position** - The location of the vertical / Y pixel position.
- **Intensity** - This is a value from `0` to `255` which specifies how bright the pixel should be, 0 is off and 255 is maximum brightness. Use an in-between value to show a less bright pixel.
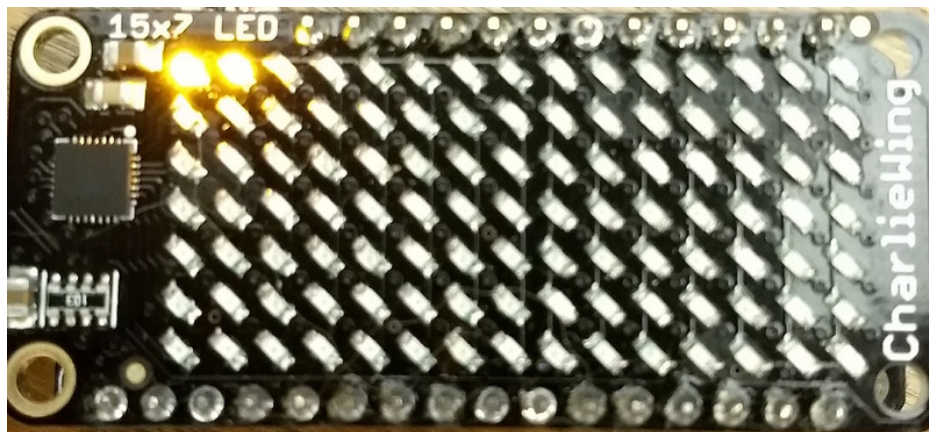
For example to set pixel 0, 0 to full brightness run:

```
display.pixel(0, 0, 255)
```
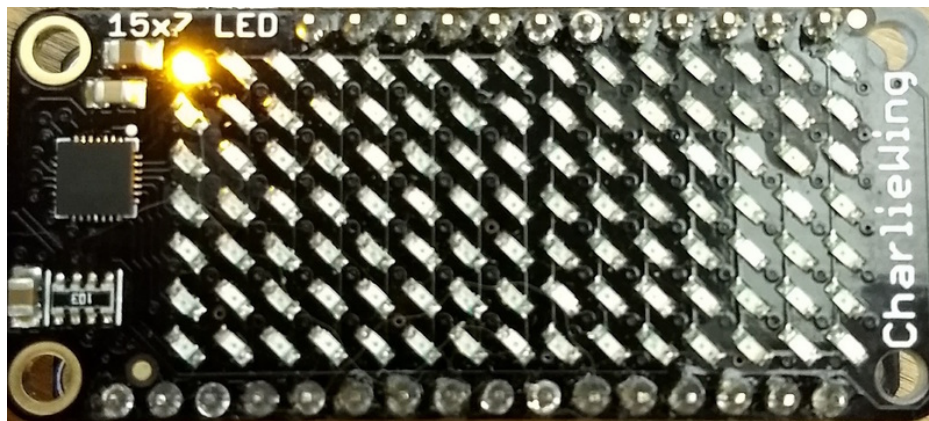

Or to set the pixel next to it horizontally to half brightness run:

```
display.pixel(1, 0, 127)
```


You can turn off individual pixels by setting them to an intensity of zero.

You can even make pixels blink! The board supports a fixed blink rate that you set using the `blink` function. This function takes in the number of milliseconds to use for the blink rate (but internally it can only blink in 270ms increments so you might not get an exact match). For example to blink pixels about once every half second call:

```
display.blink(500)
```

You'll notice nothing actually changes on the board. This is because in addition to intensity each LED pixel has a blink state which can be enabled and disabled. The `fill` command can actually set all pixels and turn them on to blink:

```
display.fill(127, blink=True)
```

You can turn off the blinking by setting `blink=False` .

The `pixel` command supports the blink parameter too! You can turn on and off blinking pixel by pixel as needed. For

example to turn on blinking for pixel `0, 0` :

```
display.pixel(0, 0, 127, blink=True)
```

Currently the Charlieplex module is very simple and only exposes pixel set commands. In the future more advanced graphics commands like line drawing, text display, etc. might be implemented but for now you'll need to manipulate the pixels yourself.

Finally the display supports holding up to 8 frames of pixel data. Each frame contains an entire matrix of LED pixel state (intensity, blinking, etc.) and by default the module starts you on frame 0. You can change to start displaying and drawing on another frame by calling `frame` which takes these parameters:

- **Frame number** - This is the frame number to make the active frame for display or drawing. There are 8 frames total, `0` through `7` .
- `show` - An optional boolean that defaults to `True` and specifies if the frame should be immediately displayed (`True`) or just made active so that pixel and fill commands draw on it but it's not yet shown.

For example to clear frame `1` and draw a few pixels on it, then display it you can run:

```
display.frame(1, show=False)
display.fill(0)
display.pixel(0, 0, 255)
display.pixel(1, 1, 255)
display.pixel(2, 2, 255)
display.frame(1)  # show=True is the default, the frame will be displayed!
```
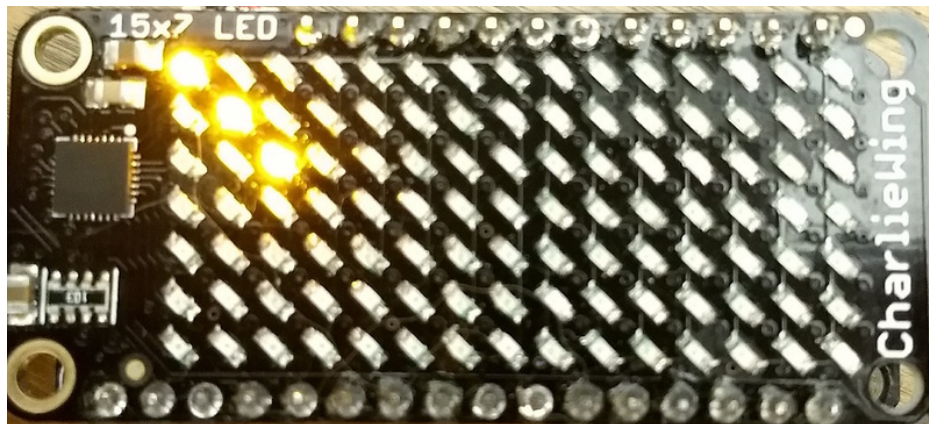

Notice how the first call switches to make frame 1 the active frame but doesn't display it because show is set to false. Then the frame pixel data is changed with fill and pixel commands, and finally the frame is shown by calling frame again but letting the default show = True be used so the frame is displayed.

Using frames you can build simple animations by drawing each frame and swapping between them over time!

That's all there is to the basic Charlieplex driver module usage!

Full Example Code

```
Temporarily unable to load content:
```

Text Scrolling Example

NOTE: When running this example on Raspberry Pi, you must have the font8x5.bin file found here (https://adafru.it/Edh) in the same directory as the program!

wget https://raw.githubusercontent.com/adafruit/Adafruit_CircuitPython_framebuf/master/examples/font5x8.bin

```
Temporarily unable to load content:
```

Python Docs

Python Docs (https://adafru.it/C55)

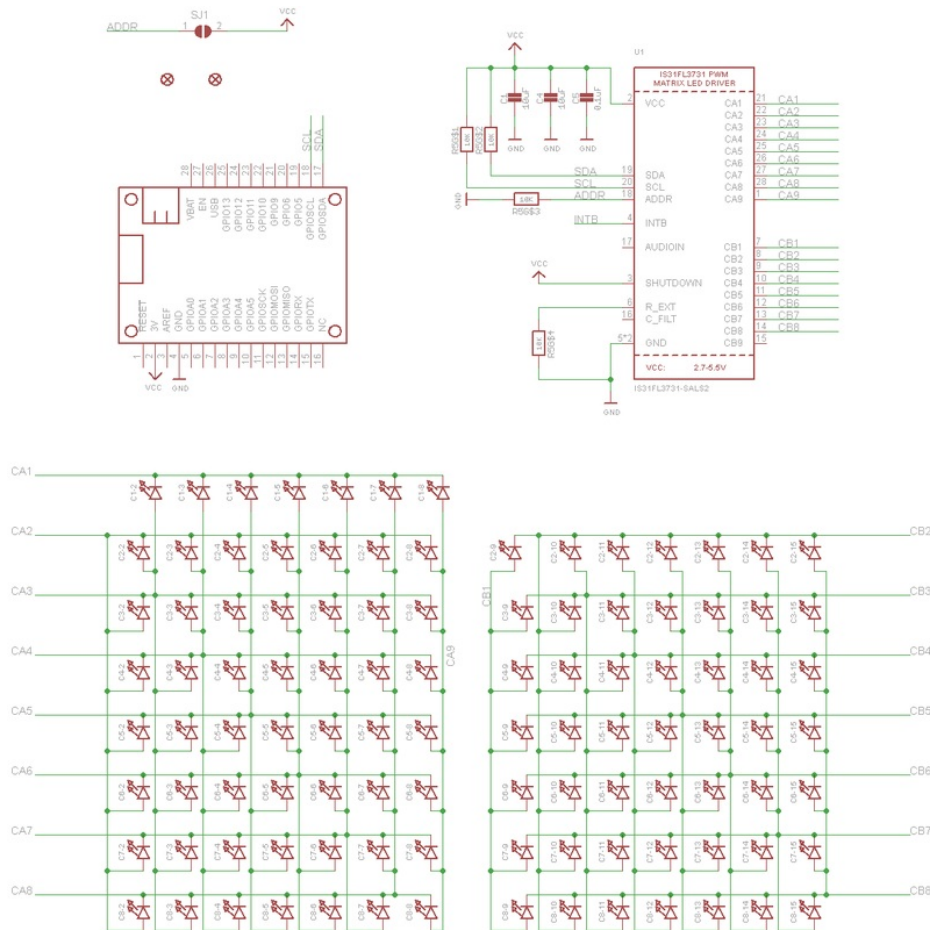# Downloads

Datasheets & Files

- EagleCAD PCB Files on GitHub (https://adafru.it/nEB)
- Fritzing objects in Adafruit Fritzing library (https://adafru.it/aP3)
- IS31FL3731 Library (https://adafru.it/IWA)
- IS31FL3731 Datasheet (https://adafru.it/IWC)

Schematic

Click to embiggen

Fabrication Print

Dimensions in inches