Adafruit CharliePlex LED Matrix Bonnet

Created by Kattni Rembor

https://learn.adafruit.com/adafruit-charlieplex-bonnet

Last updated on 2021-11-15 07:34:39 PM EST

Table of Contents

Overview

You won't be able to look away from the mesmerizing patterns created by this Adafru it CharliePlex LED Matrix Display Bonnet. This 16x8 LED display can be placed atop any Raspberry Pi computer with a 2x20 connector, for a beautiful, bright grid of 128 CharliePlexed LEDs. It even comes with a built-in CharliePlex driver that is run over I2C.

We carry these Bonnets in a few vivid colors (https://adafru.it/EcS).
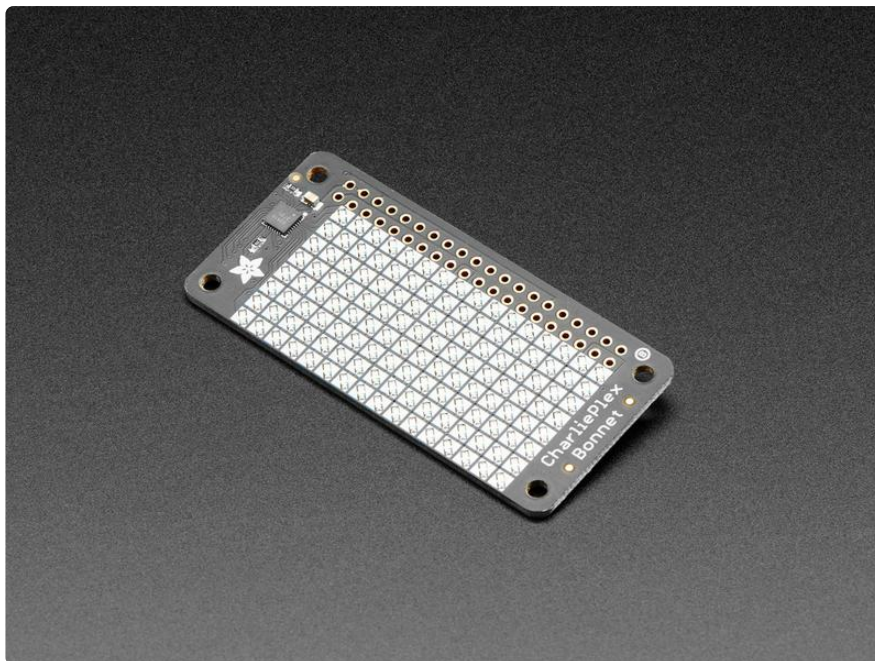
What is particularly nice about this Bonnet is the I2C LED driver chip has the ability to PWM each individual LED in a 16x8 grid so you can have beautiful LED lighting effects, without a lot of pin twiddling. Simply tell the chip which LED on the grid you want lit, and what brightness and it's all taken care of for you. You get 8-bit (256 level) dimming for each individual LED.
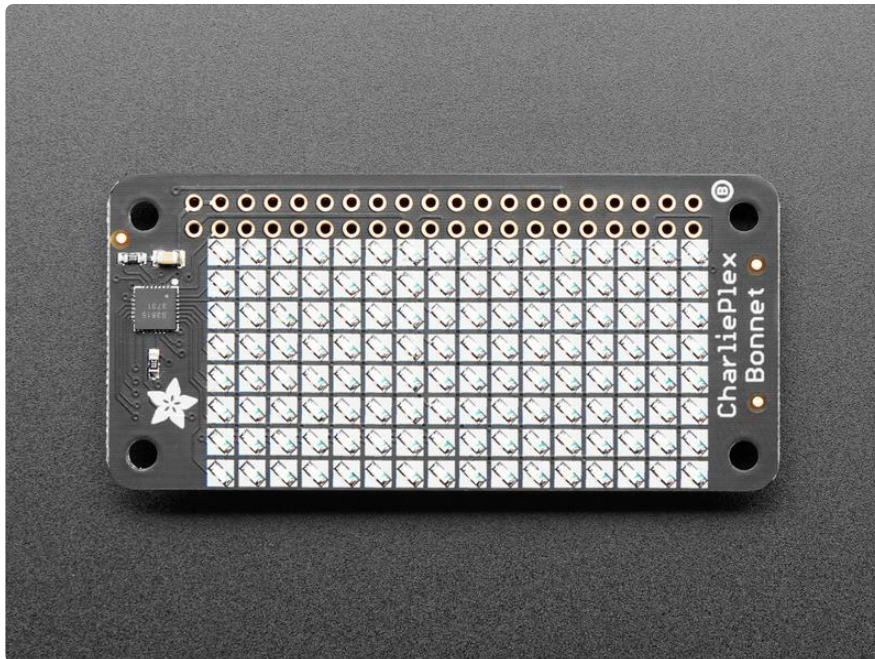
The IS31FL3731 is a nice little chip - and it runs happily over 3.3V power. Inside is enough RAM for 8 separate frames of display memory so you can set up multiple frames of an animation and flip them to be displayed with a single command. Since it uses I2C, it takes up only the SDA/SCL pins on your Pi and can share those pins with other I2C devices and sensors.

The bonnet comes fully assembled, no soldering required, so you can plug it in immediately. To program it, you'll use our CircuitPython library (https://adafru.it/zIE), which works with Linux computers like Raspberry Pi. After a pip install, run some of the examples to display simple graphics or text (https://adafru.it/EcT).
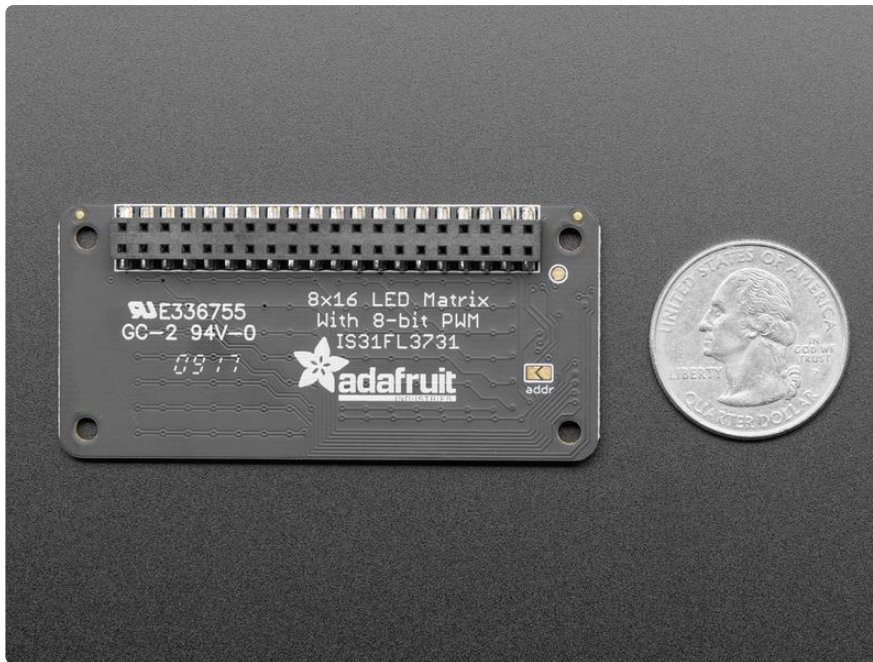
Pinouts
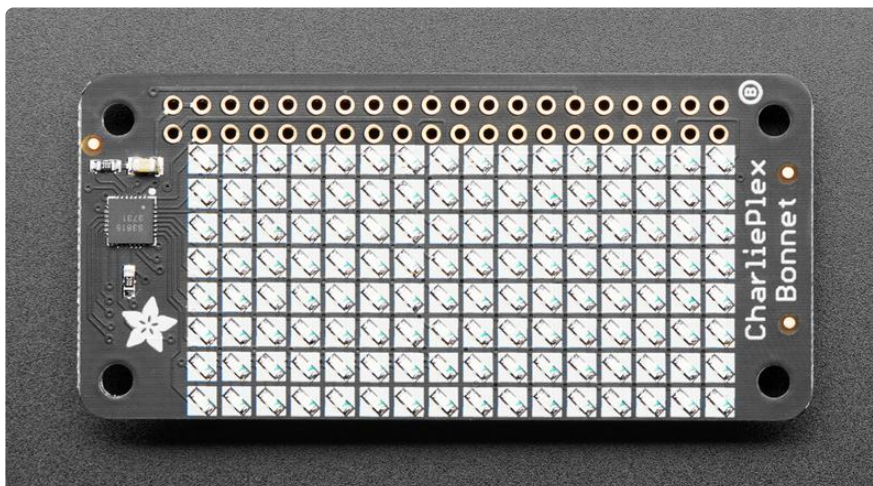
The CharliePlex LED Matrix Bonnet includes the IS31FL3731 driver chip and 128 LEDs in an 8x16 matrix. Let's take a look!

The main pins used by the Bonnet are 5V, GND, SDA and SCL on the Raspberry Pi. We power the LEDs from the 5V supply, but the SDA/SCL logic level is still only 3.3V so it's safe!

IS31FL3731 Driver Chip
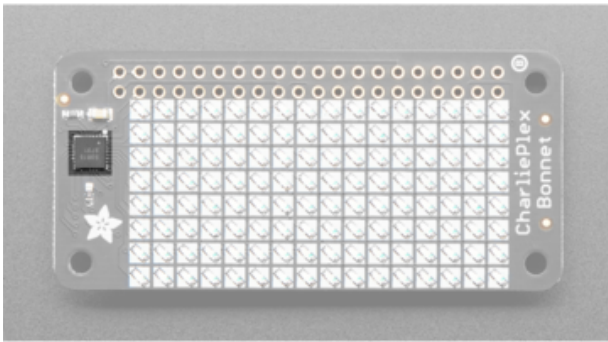
The IS31FL3731 drives the LED matrix via I2C.

8 x 16 LED Matrix
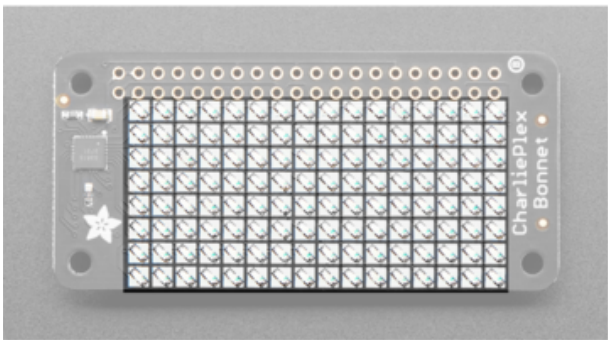
The 8 x 16 LED matrix on the CharliePlex Bonnet provides 128 LEDs.

Address Jumper
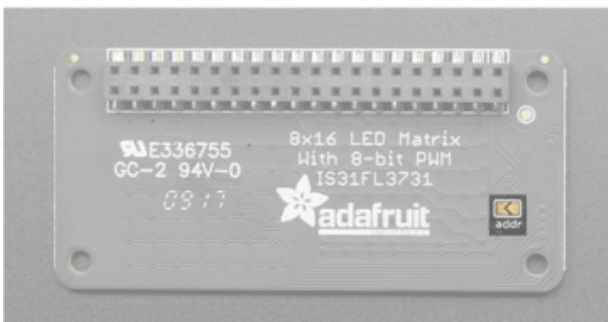
The default I2C address for the Bonnet is `0x74`. Solder this jumper closed to change it to `0x70` to avoid I2C address conflicts, if needed.

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:
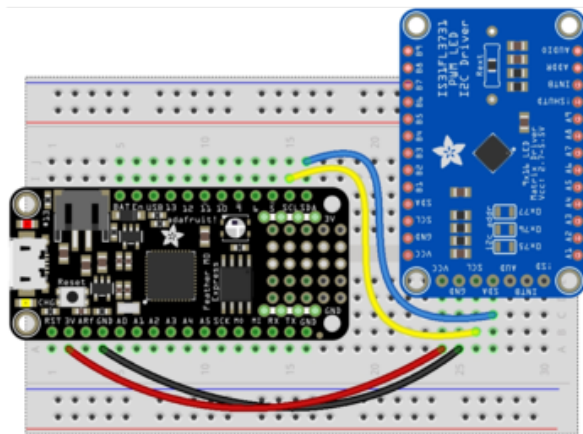
- Board 3V to sensor VCC
- Board GND to sensor GND
- Board SCL to sensor SCL
- Board SDA to sensor SDA

And here is the CharlieWing on a Feather M4:
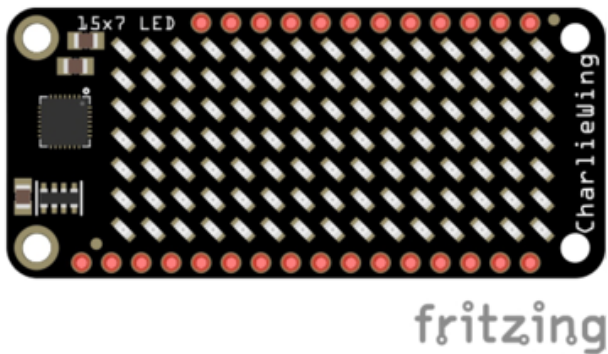
- Assemble the CharlieWing by soldering headers onto the board.
- Once assembled, plug it into a Feather!

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:
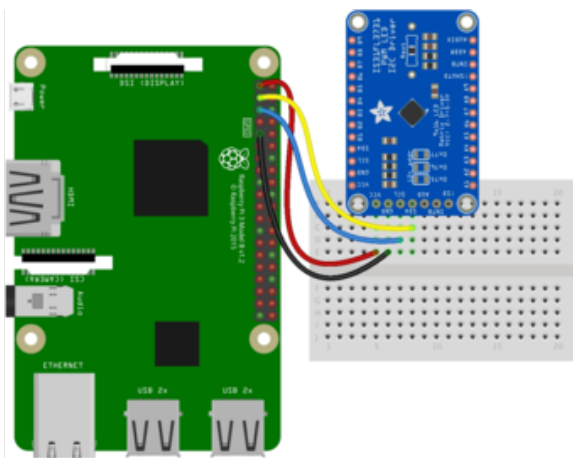
- Pi 3V3 to sensor VIN
- Pi GND to sensor GND
- Pi SCL to sensor SCL
- Pi SDA to sensor SDA

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:

```
import board
import busio
```

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

```
from adafruit_is31fl3731.matrix import Matrix as Display
```

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

```
from adafruit_is31fl3731.charlie_wing import CharlieWing as Display
```

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

```
from adafruit_is31fl3731.charlie_bonnet import CharlieBonnet as Display
```

Now, no matter which board you are using, you'll create the I2C object and pass that into `Display` .

```
i2c = busio.I2C(board.SCL, board.SDA)
display = Display(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)
```
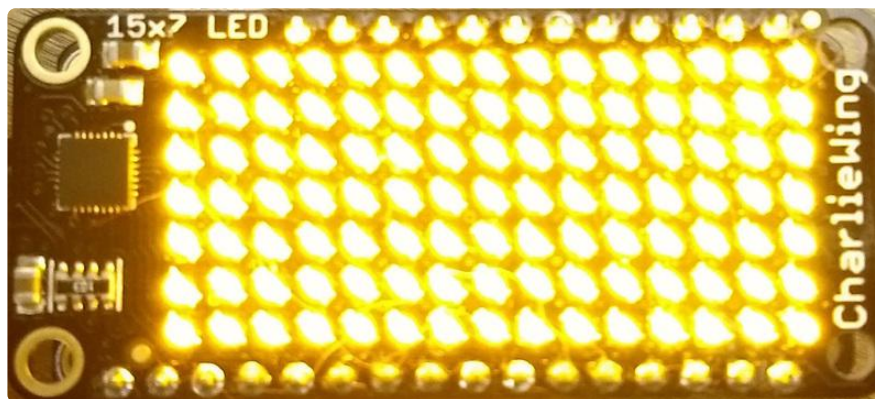

You might notice some buzzing or ringing sounds from the display when all pixels are lit, this is normal as the Charlieplex driver quickly switches LEDs on and off.

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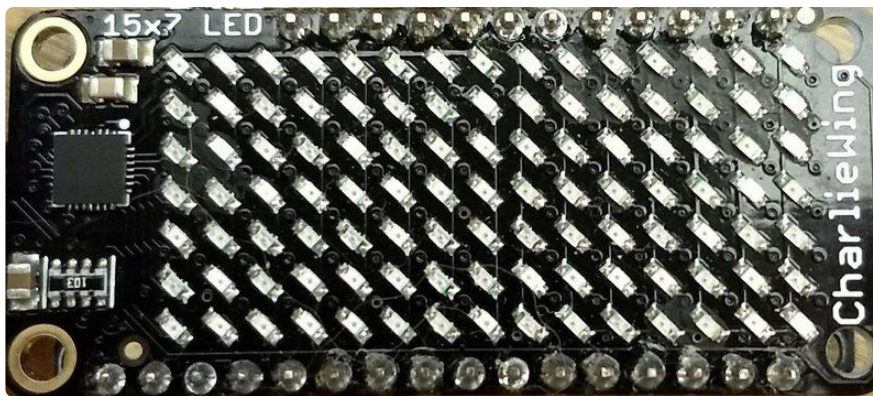
Be careful setting all pixels to 255 maximum brightness! This might pull more power than your computer's USB port can provide if you are powering your board over USB. Use an external powers supply or battery when lighting lots of LEDs to max brightness.

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)
```
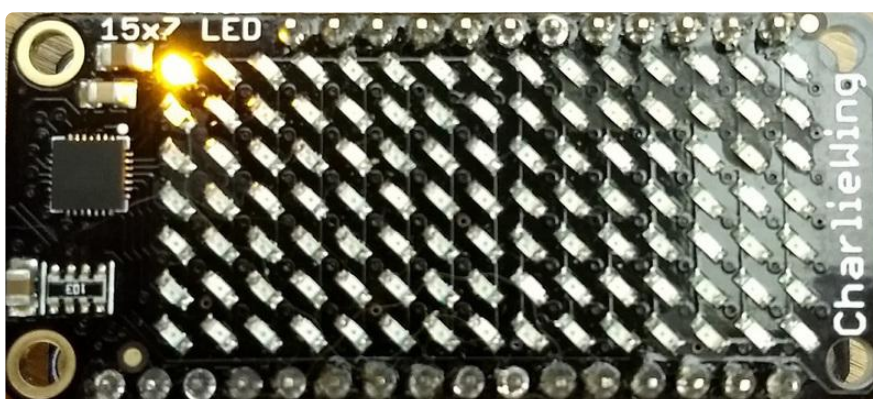

Or to set the pixel next to it horizontally to half brightness run:

```
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)
```

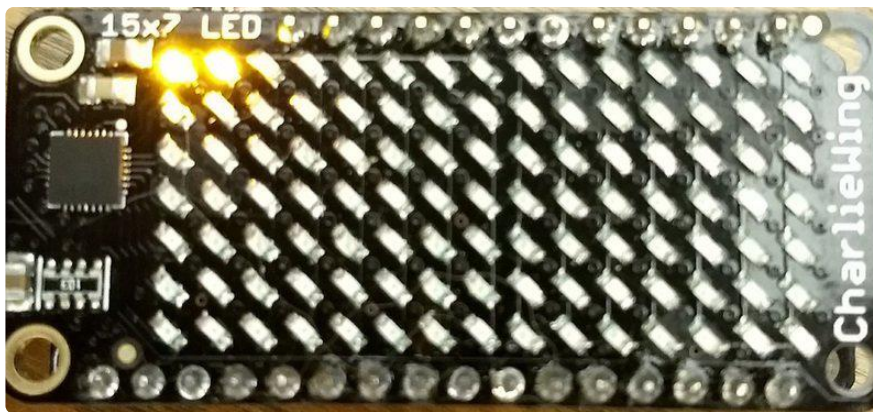You'll notice nothing actually changes on the board. This is because in addition to intensity each LED pixel has a blink state which can be enabled and disabled. The `fill` command can actually set all pixels and turn them on to blink:

```
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!
```
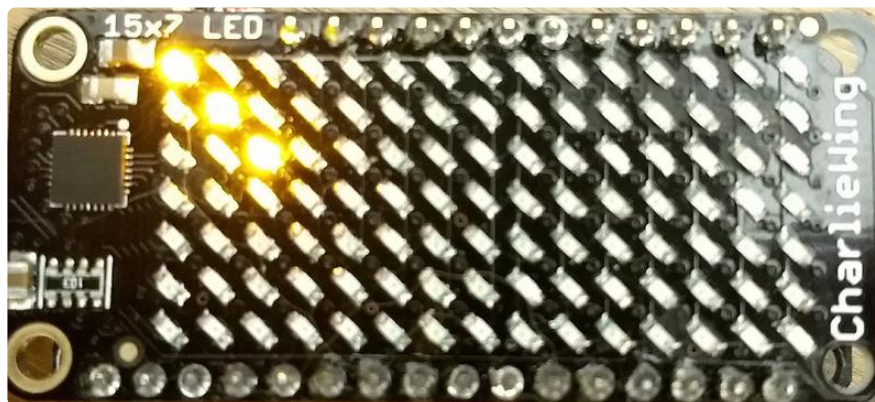

Notice how the first call switches to make frame 1 the active frame but doesn't display it because show is set to false. Then the frame pixel data is changed with fill and pixel commands, and finally the frame is shown by calling frame again but letting the default show = True be used so the frame is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

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import board
```

```
import busio

# uncomment next line if you are using Feather CharlieWing LED 15 x 7
from adafruit_is31fl3731.charlie_wing import CharlieWing as Display

# uncomment next line if you are using Adafruit 16x9 Charlieplexed PWM LED Matrix
# from adafruit_is31fl3731.matrix import Matrix as Display
# uncomment next line if you are using Adafruit 16x8 Charlieplexed Bonnet
# from adafruit_is31fl3731.charlie_bonnet import CharlieBonnet as Display
# uncomment next line if you are using Pimoroni Scroll Phat HD LED 17 x 7
# from adafruit_is31fl3731.scroll_phat_hd import ScrollPhatHD as Display
# uncomment next line if you are using Pimoroni 11x7 LED Matrix Breakout
# from adafruit_is31fl3731.matrix_11x7 import Matrix11x7 as Display

# uncomment this line if you use a Pico, here with SCL=GP21 and SDA=GP20.
# i2c = busio.I2C(board.GP21, board.GP20)

i2c = busio.I2C(board.SCL, board.SDA)

display = Display(i2c)

# draw a box on the display
# first draw the top and bottom edges
for x in range(display.width):
    display.pixel(x, 0, 50)
    display.pixel(x, display.height - 1, 50)
# now draw the left and right edges
for y in range(display.height):
    display.pixel(0, y, 50)
    display.pixel(display.width - 1, y, 50)
```

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
```

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import board
import busio
import adafruit_framebuf

# uncomment next line if you are using Feather CharlieWing LED 15 x 7
# from adafruit_is31fl3731.charlie_wing import CharlieWing as Display
# uncomment next line if you are using Adafruit 16x9 Charlieplexed PWM LED Matrix
# from adafruit_is31fl3731.matrix import Matrix as Display
# uncomment next line if you are using Adafruit 16x8 Charlieplexed Bonnet
from adafruit_is31fl3731.charlie_bonnet import CharlieBonnet as Display

# uncomment next line if you are using Pimoroni Scroll Phat HD LED 17 x 7
# from adafruit_is31fl3731.scroll_phat_hd import ScrollPhatHD as Display
# uncomment next line if you are using Pimoroni 11x7 LED Matrix Breakout
# from adafruit_is31fl3731.matrix_11x7 import Matrix11x7 as Display

# uncomment this line if you use a Pico, here with SCL=GP21 and SDA=GP20.
# i2c = busio.I2C(board.GP21, board.GP20)
```

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

display = Display(i2c)

text_to_show = "Adafruit!!"

# Create a framebuffer for our display
buf = bytearray(32)  # 2 bytes tall x 16 wide = 32 bytes (9 bits is 2 bytes)
fb = adafruit_framebuf.FrameBuffer(
    buf, display.width, display.height, adafruit_framebuf.MVLSB
)


frame = 0  # start with frame 0
while True:
    for i in range(len(text_to_show) * 9):
        fb.fill(0)
        fb.text(text_to_show, -i + display.width, 0, color=1)

        # to improve the display flicker we can use two frame
        # fill the next frame with scrolling text, then
        # show it.
        display.frame(frame, show=False)
        # turn all LEDs off
        display.fill(0)
        for x in range(display.width):
            # using the FrameBuffer text result
            bite = buf[x]
            for y in range(display.height):
                bit = 1 << y & bite
                # if bit > 0 then set the pixel brightness
                if bit:
                    display.pixel(x, y, 50)

        # now that the frame is filled, show it.
        display.frame(frame, show=True)
        frame = 0 if frame else 1
```

Python Examples

If you want to expand the capabilities of the CharliePlex LED Matrix even more, we can add Pillow into the mix. So we'll show you how to add Pillow and then go over a couple of examples that use Pillow.

Additional Setup

If you haven't already installed the library, follow the setup section on the Python & CircuitPython page. If you have, then continue.

DejaVu TTF Font

Raspberry Pi usually comes with the DejaVu font already installed, but in case it didn't, you can run the following to install it:

- `sudo apt-get install ttf-dejavu`

Pillow Library

We also need Pillow, also known as PIL, the Python Imaging Library, to allow using text with custom fonts. There are several system libraries that PIL relies on, so installing via a package manager is the easiest way to bring in everything:

- `sudo apt-get install python3-pil`

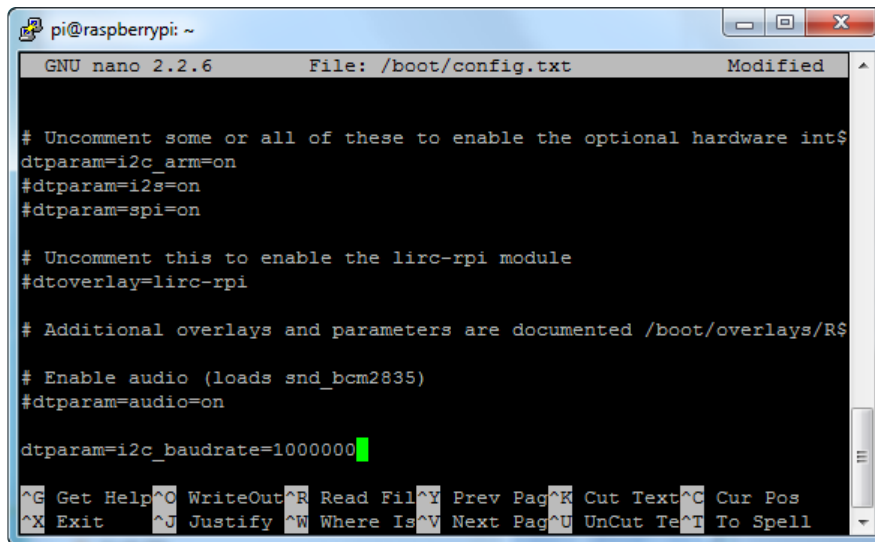That's it. You should be ready to go.

Speeding up the Display on Raspberry Pi

For the best performance, especially if you are doing fast animations, you'll want to tweak the I2C core to run at 1MHz. By default it may be 100KHz or 400KHz

To do this, edit the config with `sudo nano /boot/config.txt`

and add to the end of the file

`dtparam=i2c_baudrate=1000000`

Reboot to 'set' the change.

Scrolling Marquee Example

The first example, we're going to take a look at an example that will take some text, draw it in a TrueType font and then scroll the rendered text. Let's start by looking at the code in each section.

We start by importing the libraries we need which include the board and a few Pillow modules.

```
import board
from PIL import Image, ImageDraw, ImageFont
```

Next we do the import for the IS31FL3731 driver for the matrix itself. Since the different boards have been split into their own modules, we just import the appropriate module and alias it as `Display`.

For instance, if you have the breakout instead of the bonnet, you'll want to uncomment that and comment out the bonnet line.

```
# uncomment next line if you are using Adafruit 16x9 Charlieplexed PWM LED Matrix
# from adafruit_is31fl3731.matrix import Matrix as Display
# uncomment next line if you are using Adafruit 16x8 Charlieplexed Bonnet
from adafruit_is31fl3731.charlie_bonnet import CharlieBonnet as Display

# uncomment next line if you are using Pimoroni Scroll Phat HD LED 17 x 7
# from adafruit_is31fl3731.scroll_phat_hd import ScrollPhatHD as Display
```

Next we set a couple of variables. We have the `SCROLLING_TEXT` variable. Go ahead and change the text if you would like. It shouldn't matter how long, though you probably shouldn't make it too long if you want to see it loop. You can set `BRIGHTNE SS` as well, in case you want to adjust the intensity.

```
SCROLLING_TEXT = "You can display a personal message here..."
BRIGHTNESS = 64  # Brightness can be between 0-255
```

Next we do the basic setup for the display by declaring the I2C object and passing that into the display.

```
i2c = board.I2C()

display = Display(i2c)
```

Next we go ahead and load up the Deja Vu font into an object. We are going with an 8 pixel high font because that's the largest we can fit on the display and still see everything.

```
# Load a font
font = ImageFont.truetype('/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf', 8)
```

In this next part, we first start by getting the width and height of what the text would be when rendered with the font we chose. Then we create a virtual image of that width and height and draw the text onto it.

```
# Create an image that contains the text
text_width, text_height = font.getsize(SCROLLING_TEXT)
text_image = Image.new('L', (text_width, text_height))
text_draw = ImageDraw.Draw(text_image)
text_draw.text((0, 0), SCROLLING_TEXT, font=font, fill=BRIGHTNESS)
```

Next we create a virtual image that's the same size as the display. This will be where we draw what we want to actually display.

```
# Create an image for the display
image = Image.new('L', (display.width, display.height))
draw = ImageDraw.Draw(image)
```

Finally we get to our main loop. We start by drawing a rectangle to be sure we are not leaving any existing text behind. Then we `paste` our image of the text onto the image we are going to display using the value of `x`, which represents the left offset we want to use to give a nice scrolling effect. We have a `for` loop which will scroll

the complete text plus empty display width by one iteration. That's all placed inside an infinite `while` loop for endless iterations.

```python
while True:
    for x in range(text_width + display.width):
        draw.rectangle((0, 0, display.width, display.height), outline=0, fill=0)
        image.paste(text_image, (display.width - x, display.height // 2 -
text_height // 2 - 1))
        display.image(image)
```

Now go ahead and run the example code.

`python3 is31fl3731_pillow_marquee.py`

You should see the display showing a message scrolling from right to left.
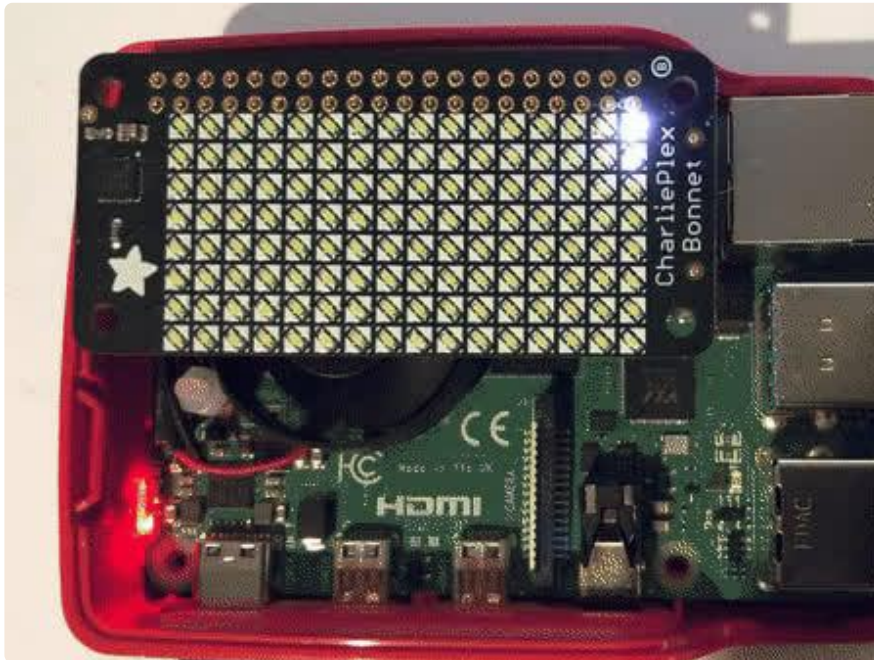
Full Source Code

```python
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
Example to scroll some text as a marquee

This example is for use on (Linux) computers that are using CPython with
Adafruit Blinka to support CircuitPython libraries. CircuitPython does
not support PIL/pillow (python imaging library)!

Author(s): Melissa LeBlanc-Williams for Adafruit Industries
"""

import board
from PIL import Image, ImageDraw, ImageFont
```

```
# uncomment next line if you are using Adafruit 16x9 Charlieplexed PWM LED Matrix
# from adafruit_is31fl3731.matrix import Matrix as Display
# uncomment next line if you are using Adafruit 16x8 Charlieplexed Bonnet
from adafruit_is31fl3731.charlie_bonnet import CharlieBonnet as Display

# uncomment next line if you are using Pimoroni Scroll Phat HD LED 17 x 7
# from adafruit_is31fl3731.scroll_phat_hd import ScrollPhatHD as Display

SCROLLING_TEXT = "You can display a personal message here..."
BRIGHTNESS = 64  # Brightness can be between 0-255

i2c = board.I2C()

display = Display(i2c)

# Load a font
font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 8)

# Create an image that contains the text
text_width, text_height = font.getsize(SCROLLING_TEXT)
text_image = Image.new("L", (text_width, text_height))
text_draw = ImageDraw.Draw(text_image)
text_draw.text((0, 0), SCROLLING_TEXT, font=font, fill=BRIGHTNESS)

# Create an image for the display
image = Image.new("L", (display.width, display.height))
draw = ImageDraw.Draw(image)

# Load the text in each frame
while True:
    for x in range(text_width + display.width):
        draw.rectangle((0, 0, display.width, display.height), outline=0, fill=0)
        image.paste(
            text_image, (display.width - x, display.height // 2 - text_height // 2 -
1)
        )
        display.image(image)
```

Animated GIF Example

Next let's take a look at an animated GIF player example. First we'll start by downloading an animated GIF and copying that into the same folder as the script as a dafruit-star-rotating.gif. It looks tiny and that's because it is. It is 8x8 pixels which works out nicely for the CharliePlex matrix.

Download Rotating Star

https://adafru.it/HAk

Now let's start with the first section, the imports. You may be surprised that this code uses fewer Pillow modules than the previous example. We are also adding `sys`, which we mostly use for passing the name of an animated gif.

```
import sys
import board
from PIL import Image
import adafruit_is31fl3731
```

Next we do the usual setup for the CharliePlex display.

```
i2c = board.I2C()

# uncomment line if you are using Adafruit 16x9 Charlieplexed PWM LED Matrix
#display = adafruit_is31fl3731.Matrix(i2c)
# uncomment line if you are using Adafruit 16x9 Charlieplexed PWM LED Matrix
display = adafruit_is31fl3731.CharlieBonnet(i2c)
```

Now we make sure the user specified a gif file, so we have something to work with that's not hard-coded and open the file. If the file wasn't specified, we are using `sys.exit()`, since that is the preferred way to do it if you are importing `sys` anyways.

```
# Check that the gif was specified
if len(sys.argv) &lt; 2:
    print("No image file specified")
    print("Usage: python3 is31fl3731_pillow_animated_gif.py animated.gif")
    sys.exit()

# Open the gif
image = Image.open(sys.argv[1])
```

We need to check that this is an animated gif. While we could have just displayed it as a static gif in this case, the point was to show how to display the animation.

```
# Make sure it's animated
if not image.is_animated:
    print("Specified image is not animated")
    sys.exit()
```

Next we get some gif animation information such as the delay. Only the duration of the first frame is extractable at the time of this writing with Pillow.

```
# Get the autoplay information from the gif
delay = image.info['duration']
```

The loop number is a little trickier because it means different things between the IS31FL3731 chip and an animated gif. With an animated gif, it is guaranteed to play at least once and then loop by the number of times is provided by the loop value, unless it is zero, which means forever.

With the IS31FL3731, loops mean exactly the number of loops to play the animation, unless it is zero, in which case it will play forever.

So if `loop` is 0, we just pass it on. If we only want to play the animation once, then `loop` is not provided in the image information. If it is more than once, we need to count the first time it plays plus the number of times to loop the animation.

```python
# Figure out the correct loop count
if "loop" in image.info:
    loops = image.info['loop']
    if loops > 0:
        loops += 1
else:
    loops = 1
```

Next, we need to make sure these values are in the ranges that the driver likes. The number of frames in the animation is available from the property `n_frames` and the IS31FL3731 can handle a maximum of 8 frames, so if a longer animation is provided, only the first 8 frames are used.

```python
# IS31FL3731 only supports 0-7
if loops > 7:
    loops = 7

# Get the frame count (maximum 8 frames)
frame_count = image.n_frames
if frame_count > 8:
    frame_count = 8
```

Now that we have a frame count, we will go through each of those frames and load the frame image into the IS31FL3731 using the `paste` function and center the image. First the frame is converted to a 256-grayscale image, which is what mode L is, and then it is copied into a centered position, which is calculated from the difference in size between the display and image. After that, it is inserted as the current frame number.

```python
# Load each frame of the gif onto the Matrix
for frame in range(frame_count):
    image.seek(frame)
    frame_image = Image.new('L', (display.width, display.height))
    frame_image.paste(image.convert("L"), (display.width // 2 - image.width // 2,
                                            display.height // 2 - image.height // 2))
    display.image(frame_image, frame=frame)
```

Finally, we call the `auto_play` function using the delay and loop information from the animated gif.

```python
display.autoplay(delay=delay, loops=loops)
```

Now go ahead and run the example code.

```
python3 is31fl3731_pillow_animated_gif.py adafruit-star-rotating.gif
```

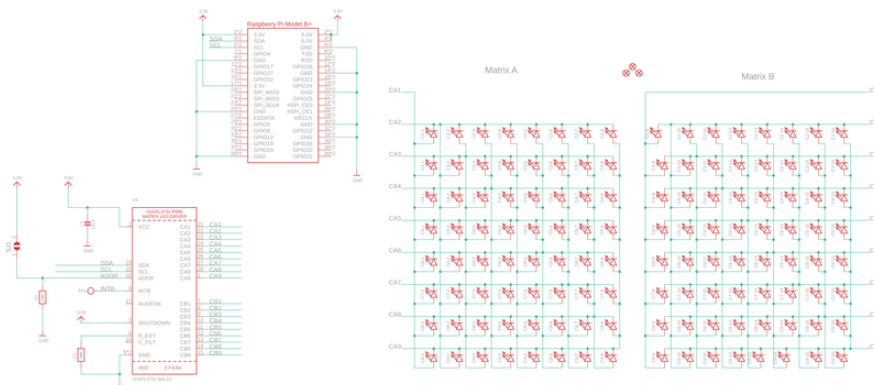You should see the rotating star appear on the display.

Python Docs

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

Downloads

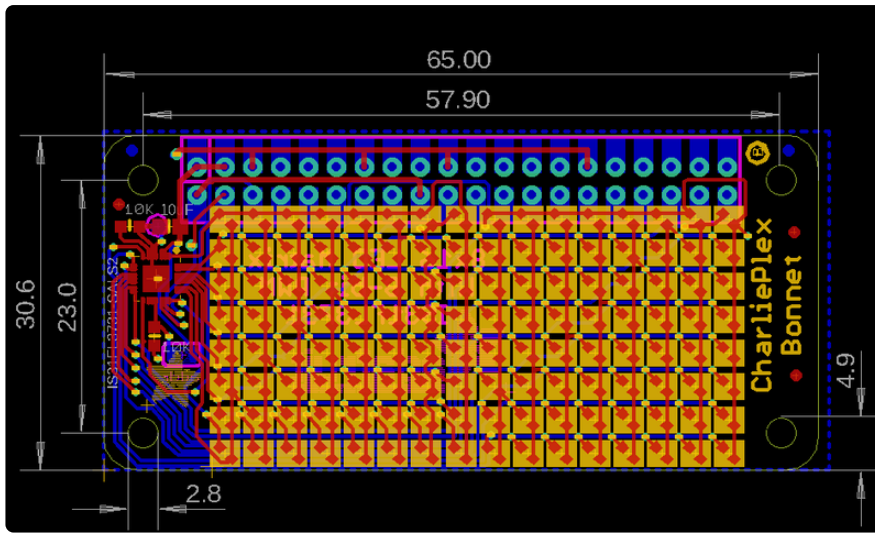## Files

- IS31FL3731 Datasheet (https://adafru.it/EcU)
- Adafruit CharliePlex Bonnet EagleCAD files on GitHub (https://adafru.it/EcV)
- Fritzing object in Adafruit Fritzing Library (https://adafru.it/EcW)

Schematic and Fab Print

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for LED Lighting Development Tools *category:*

Click to view products by Adafruit *manufacturer:*

Other Similar products are found below :

MIC2870YFT EV ADP8860DBCP-EVALZ LM3404MREVAL ADM8843EB-EVALZ TDGL014 ISL97682IRTZEVALZ LM3508TLEV EA6358NH MAX16826EVKIT MAX16839EVKIT+ TPS92315EVM-516 MAX1698EVKIT MAX6956EVKIT+ OM13321,598 DC986A DC909A DC824A STEVAL-LLL006V1 IS31LT3948-GRLS4-EB 104PW03F PIM526 PIM527 MAX6946EVKIT+ MAX20070EVKIT# MAX21610EVKIT# MAX20090BEVKIT# MAX20092EVSYS# PIM498 AP8800EV1 ZXLD1370/1EV4 MAX6964EVKIT MAX25240EVKIT# MAX25500TEVKITC# MAX77961BEVKIT06# 1216.1013 TPS61176EVM-566 TPS61197EVM TPS92001EVM-628 1270 1271.2004 1272.1030 1273.1010 1278.1010 1279.1002 1279.1001 1282.1000 1293.1900 1293.1800 1293.1700 1293.1500