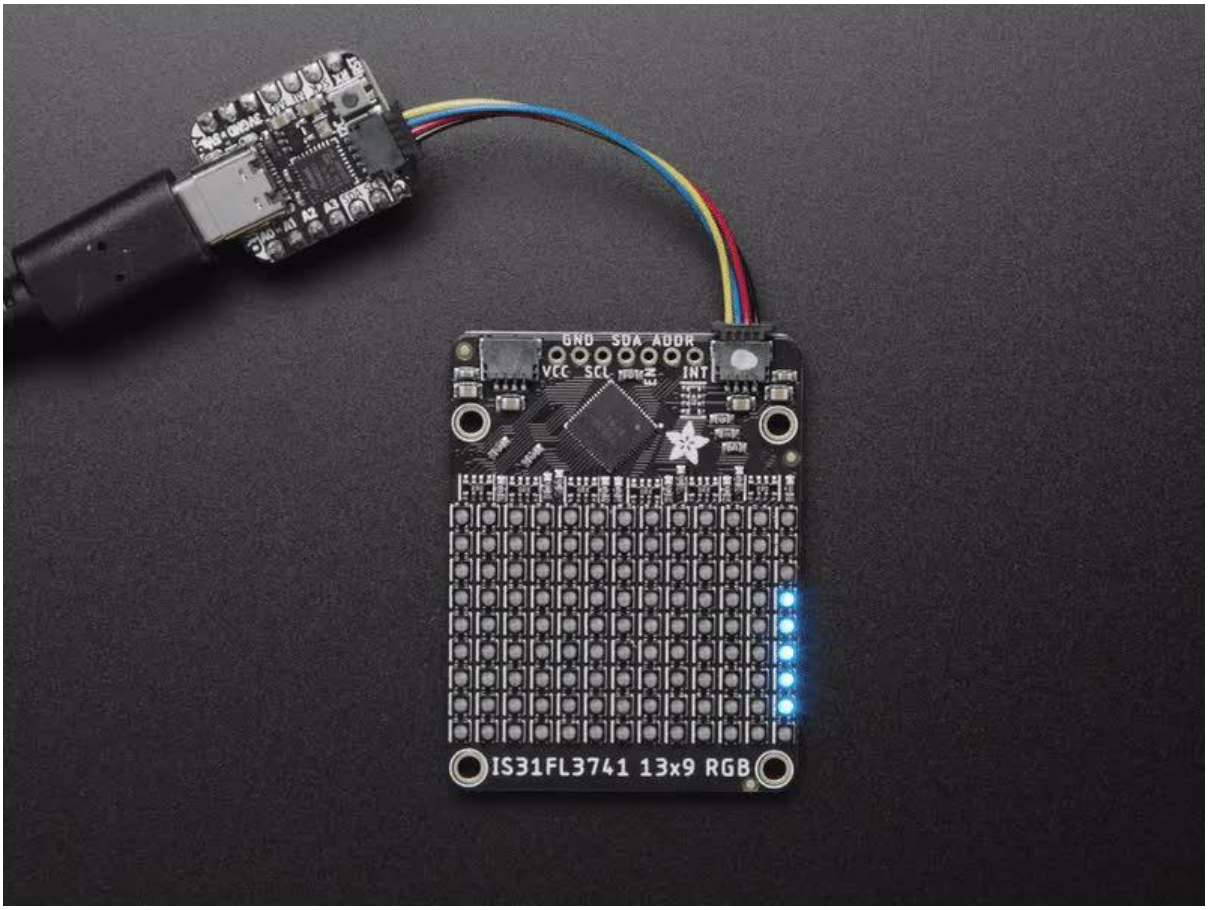
Adafruit IS31FL3741

Created by Kattni Rembor

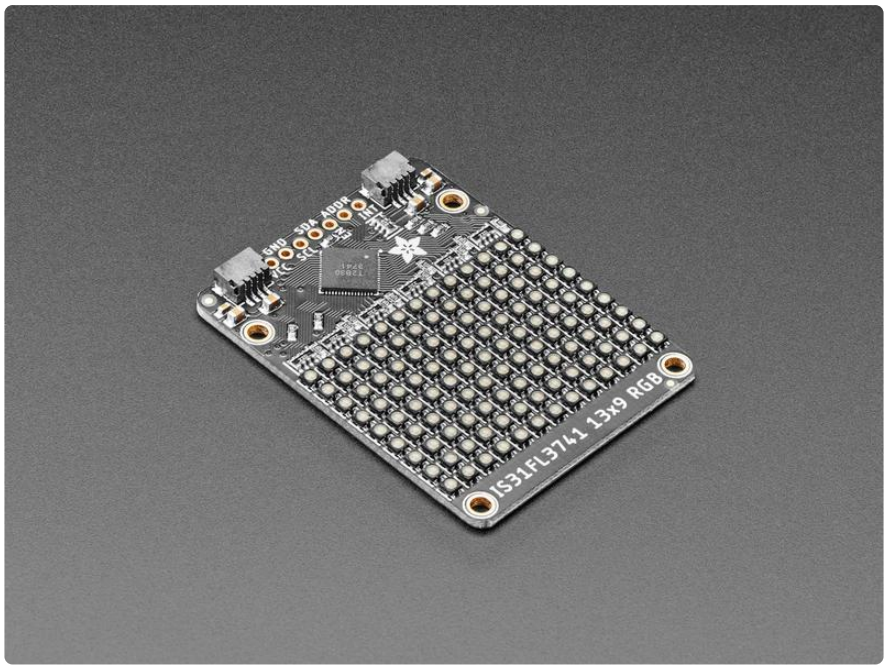


https://learn.adafruit.com/adafruit-is31fl3741

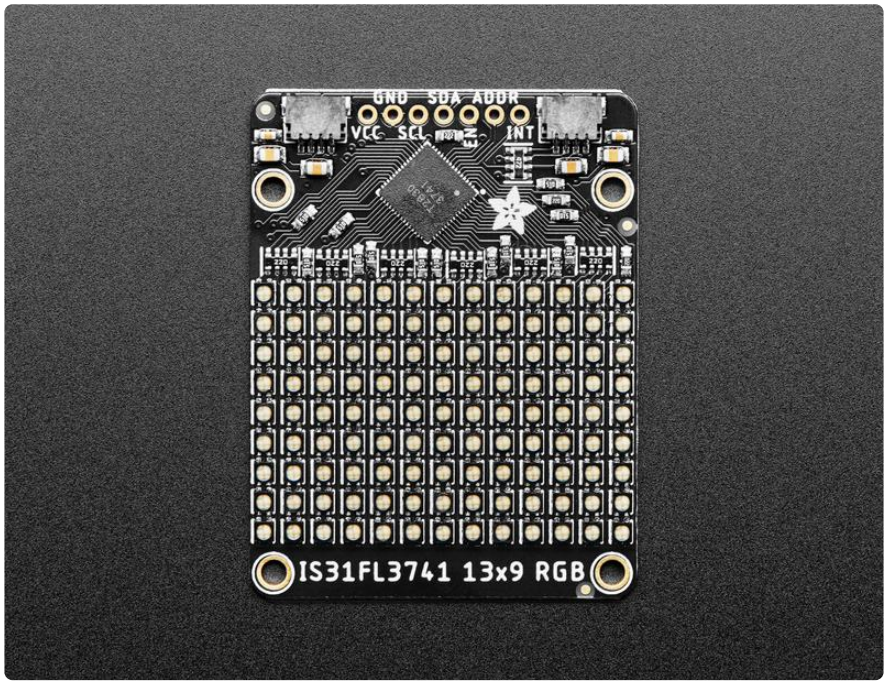
Last updated on 2022-12-01 04:07:33 PM EST

Table of Contents

Overview

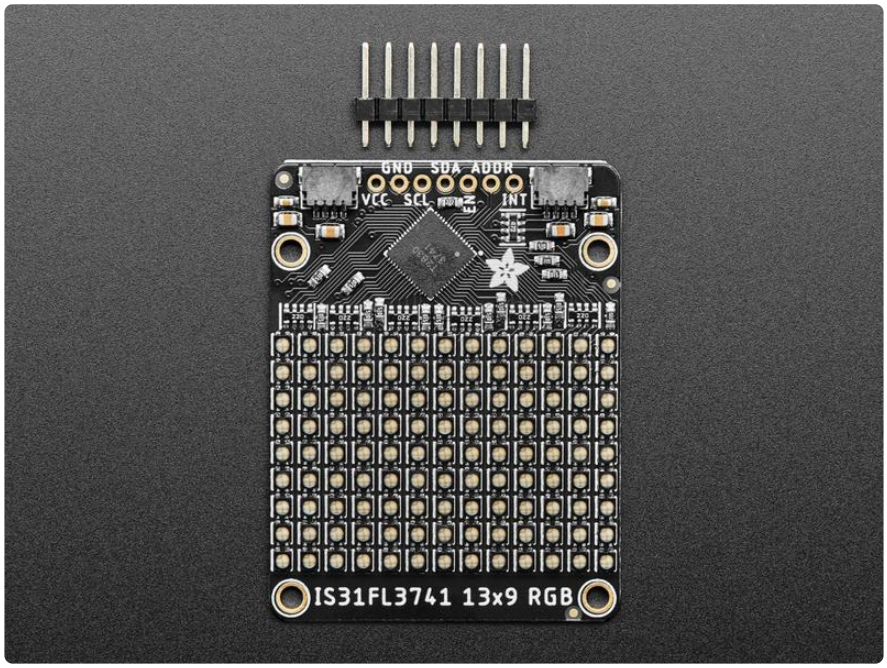


Add a splash of RGB LEDs to a project you're working on, with this adorable 13x9 RGB LED matrix breakout. It features -- no surprise -- 117 RGB LEDs, each one 2x2mm in size, in a 13x9 grid with 3mm pitch spacing.

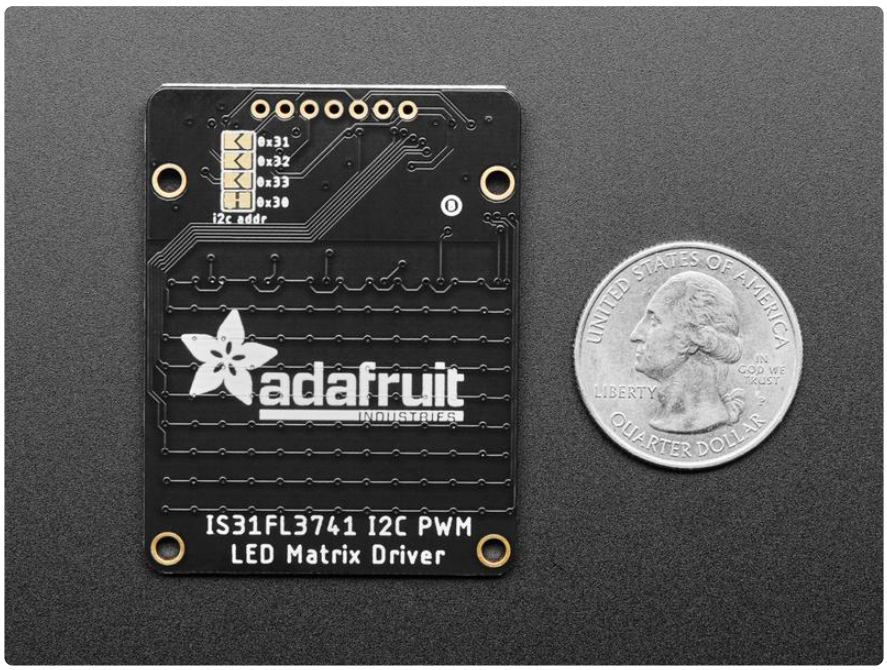


Unlike our 8x8 DotStar grid here (), these are not NeoPixel or DotStar or other 'smart' RGB LEDs. Instead of having a li'l chip in each LED, there's one large controller chip that handles all the PWM for you. The ISSI IS32FL3741 communicates over I2C and can set each LED element with 8 bit PWM for 24-bit color across the RGB elements,

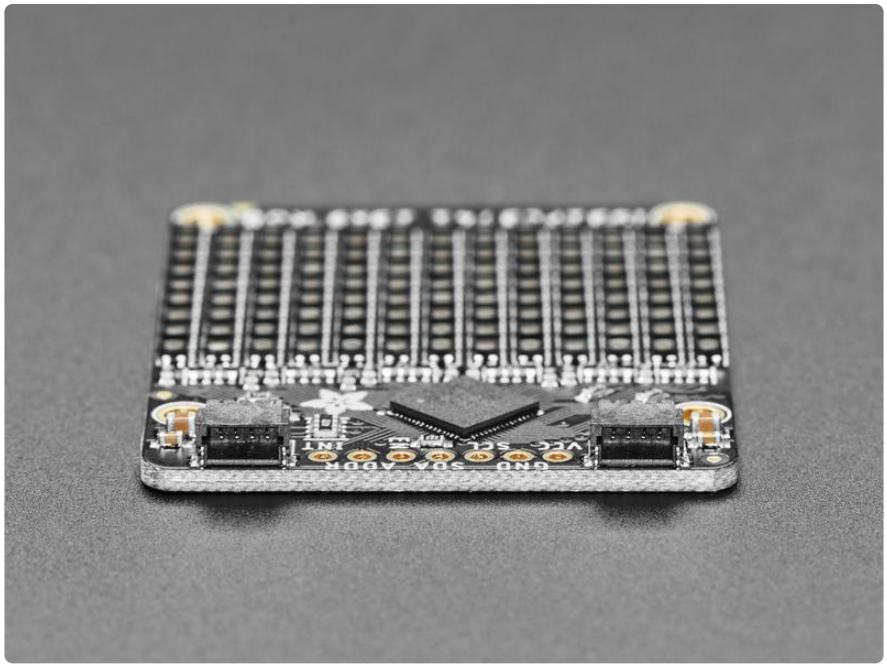
for beautiful color! There's an adjustable current driver, so you can brighten or dim the whole display without losing color resolution.



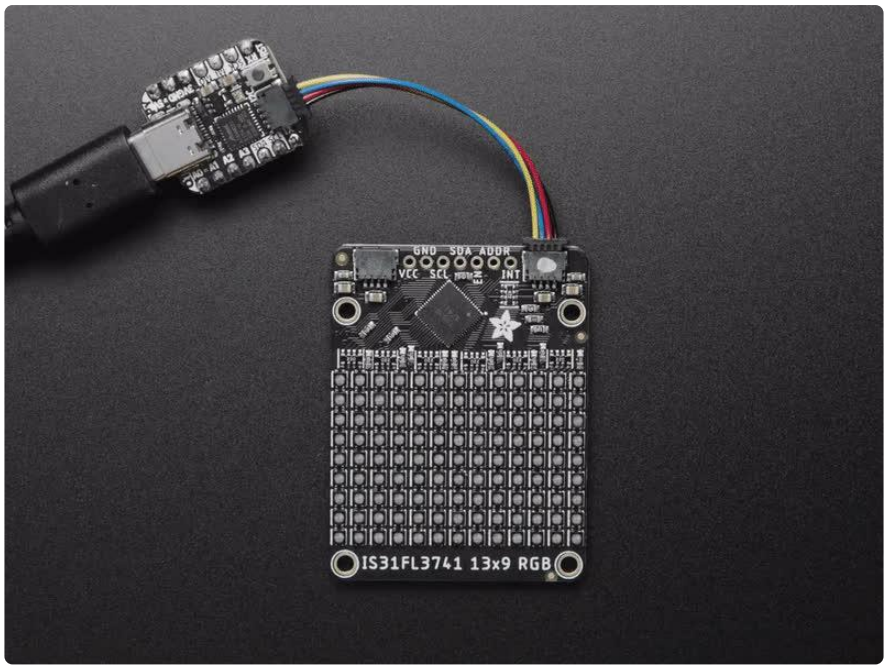
Each assembled board comes with the grid, four mounting holes, and the IS31FL3741 chip with all supporting circuitry. The board can run from 3.3 to 5V DC power and logic - powering from 5V is recommended since the green and blue LEDs look better with the extra headroom. We designed the PCB so you can tile the boards side-to-side if you desire, you'll just have to cut/solder the jumpers on the bottom to change the I2C address: Up to 4 boards can share one I2C bus.



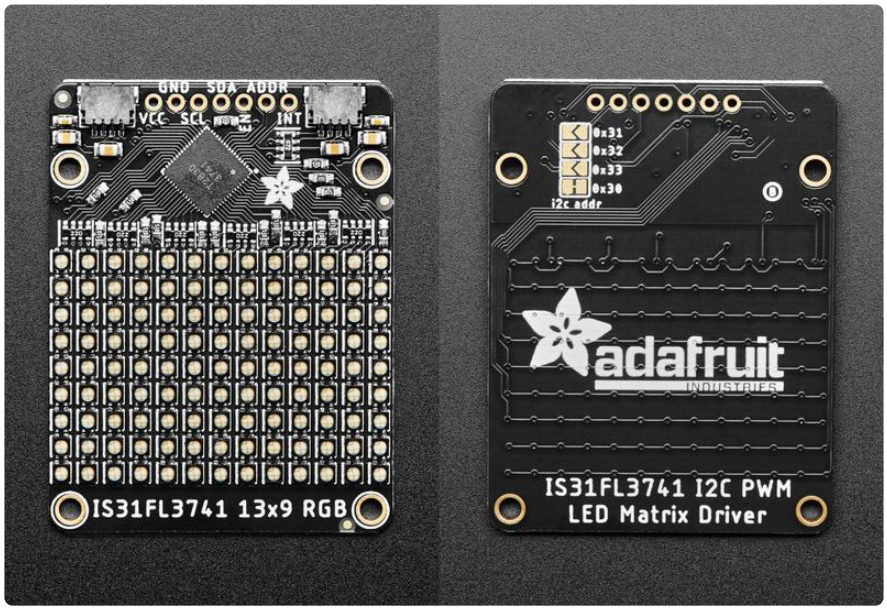
Use Arduino () or CircuitPython/Python to quickly set pixels () to any color you desire. Note that I2C makes this board very easy to wire up compared to our "HUB75"-style RGB LED Matrices () but that also makes the display slower because each pixel must be written over I2C. For a small display with simple animations, it's fine - but if you want to do video or larger graphics, we recommend upgrading to the HUB75 style.



We made it easy for you to get some color right into your next project. The LED matrix and circuitry is soldered onto a custom-made PCB with two STEMMA QT connectors () on the top, and are compatible with SparkFun Qwiic () I2C connectors. This allows you to make solderless connections between your development board and the IS31FL3741 or to chain it with a wide range of other sensors and accessories using a compatible cable ().



Pinouts



Power Pins

The chip on the breakout requires between a 2.7V and 5.5V, and can be easily used with most microcontrollers, such as an Arduino or a Feather.

- VIN - This is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V, or for a Feather use 3.3V.

- GND - This is common ground for power and logic.

I2C Pins

Default address is 0x30.

- SCL - This is the I2C clock pin, connect to your microcontroller's I2C clock line. There's a 10K pullup on this pin.
- SDA - This is the I2C data pin, connect to your microcontroller's I2C data line. There's a 10K pullup on this pin.
- STEMMA QT () - These connectors allow you to connect to development boards with STEMMA QT connectors, or to other things, with various associated accessories ().

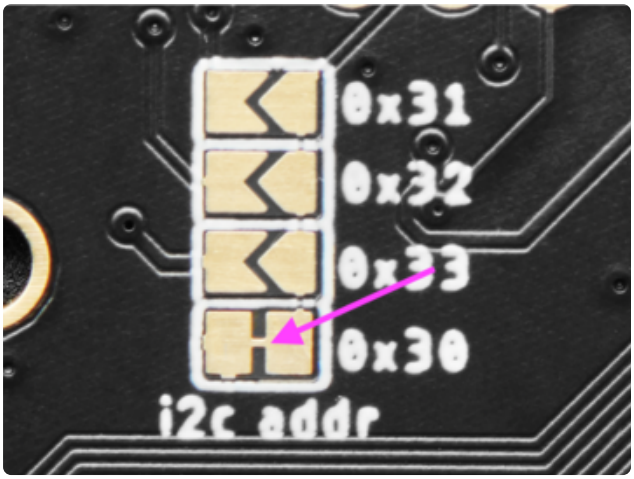
Other Pins

- EN - This is the shutdown pin.
- ADDR - This is the address pin. It is connected to GND by default. Connect this pin to one of three other pins (VCC, SCL, SDA) to change the I2C address. See below for more details.
- INT - This is the interrupt pin. This pin is active low when the interrupt event happens.

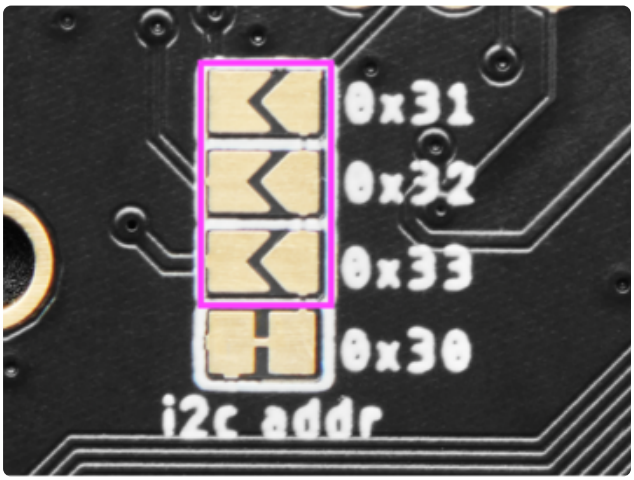
Address Pin and Jumpers

The default I2C address is 0x30. The address pin is connected to ground by default. You can connect the address pin to one of three other pins to change the address, allowing for up to four boards to be connected on the same I2C bus.

To change the I2C address, you must follow the steps below.



First, cut the bottom jumper on the back of the board (indicated by the arrow in the image). Use a hobby knife to cut the connection between the two pads.



Solder ONE of the other three sets of jumper pads (highlighted in the image) to change the address. Solder only one at a time. The addresses are listed on the back of the board next to each jumper. Here are the details:

The top jumper changes the address to 0x31. It connects the ADDR pin to SCL. The second jumper changes the address to 0x32. It connects the ADDR pin to SDA. The third jumper changes the address to 0x33. It connects the ADDR pin to VCC.

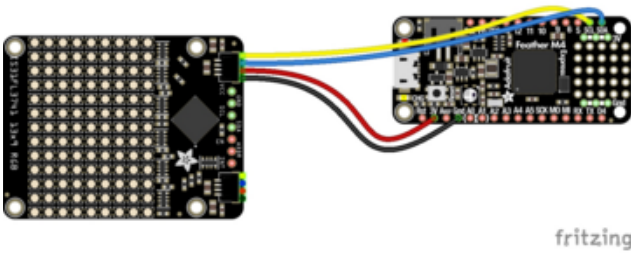
Python & CircuitPython

It's easy to use the IS31FL3741 matrix with Python and CircuitPython, and the Adafruit CircuitPython IS31FL3741 () module. This module allows you to easily write Python code that controls the LEDs.

You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library ().

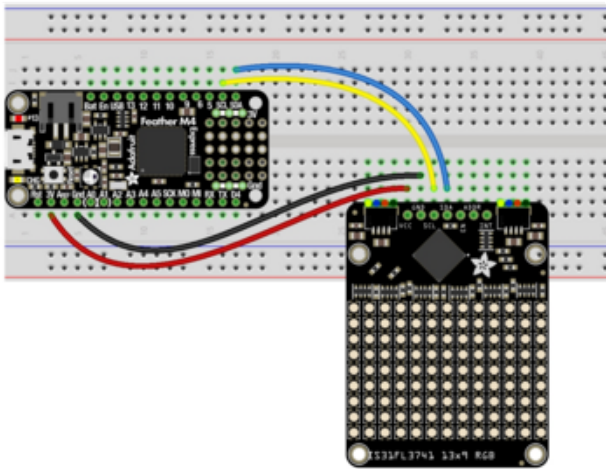
CircuitPython Microcontroller Wiring

First wire up a IS31FL3741 matrix to your board exactly as shown below. Here's an example of wiring a Feather M4 to the matrix driver breakout with I2C using one of the handy STEMMA QT () connectors:



Board 3V to matrix VCC (red wire)
Board GND to matrix GND (black wire)
Board SCL to matrix SCL (yellow wire)
Board SDA to matrix SDA (blue wire)

You can also use the standard 0.100" pitch headers to wire it up on a breadboard:

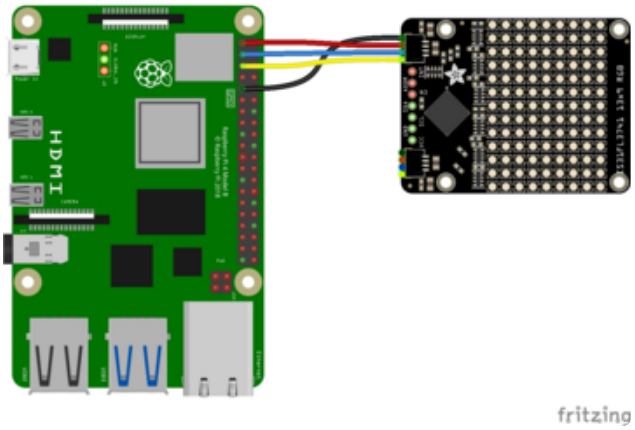


Board 3V to matrix VCC (red wire)
Board GND to matrix GND (black wire)
Board SCL to matrix SCL (yellow wire)
Board SDA to matrix SDA (blue wire)

Python Computer Wiring

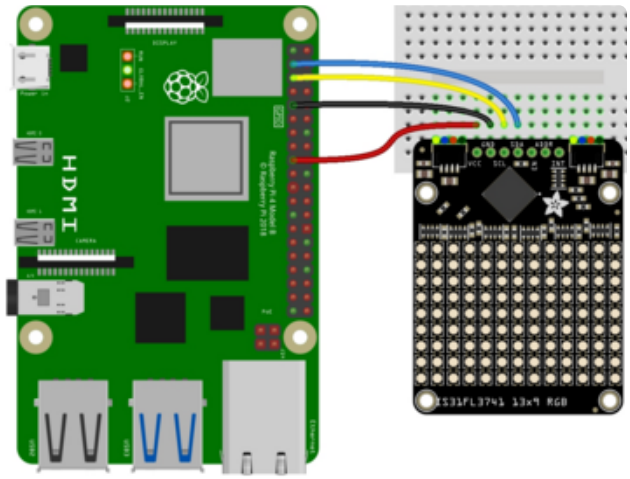
Since there's dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

Here's the Raspberry Pi wired to the matrix driver breakout using I2C and a STEMMA QT () connector:



Pi 3V3 to matrix VCC (red wire)
Pi GND to matrix GND (black wire)
Pi SCL to matrix SCL (yellow wire)
Pi SDA to matrix SDA (blue wire)

Finally here is an example of how to wire up a Raspberry Pi to the sensor using a solderless breadboard:



Pi 3V3 to matrix VCC (red wire)
Pi GND to matrix GND (black wire)
Pi SCL to matrix SCL (yellow wire)
Pi SDA to matrix SDA (blue wire)

Python Installation of IS31FL3741 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 ()!

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

- `pip3 install adafruit-circuitpython-is31fl3741`

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

CircuitPython Usage

To use with CircuitPython, you need to first install the IS31FL3741 library, and its dependencies, into the lib folder on your CIRCUITPY drive. Then you need to update code.py with the example script.

Thankfully, we can do this in one go. In the example below, click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the c ode.py file to your CIRCUITPY drive.

Your CIRCUITPY/lib folder should contain the following folders and file:

- adafruit_bus_device/
- adafruit_register/
- adafruit_is31fl3741/
- adfruit_framebuf.mpy

Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing code.py with whatever you named the file: `python3 code.py`

Basic Example Code:

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

import time
import board
```

```
import adafruit_is31fl3741

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
is31 = adafruit_is31fl3741.IS31FL3741(i2c)

is31.set_led_scaling(0xFF)  # turn on LEDs all the way
is31.global_current = 0xFF  # set current to max
is31.enable = True  # enable!

# light up every LED, one at a time
while True:
    for pixel in range(351):
        is31[pixel] = 255
        time.sleep(0.01)
        is31[pixel] = 0
```

The LEDs light up one at a time, various colors, and in an arbitrary order!

First you import the necessary modules and library. Then you create the `IS31FL3741` object, provide it the `board.I2C()` object, and save it to `is31`.

Next, you do the basic setup needed to control the LEDs. You set all the LEDs to maximum brightness with `set_led_scaling()`, set the current to max with `global_current`, and enable them by setting `enable` to `True`.

Then, inside your loop, you have a `for` loop iterating over the entire set of pixels (351 total!), and turning on each one individually for 0.01 seconds before turning it off and moving on to the next one.

Rainbow Example Code:

To run this example, you need to update your code.py file.

Click the Download Project Bundle button below to download the code.py file (and the libraries, though you don't need to update them again) in a zip file. Extract the contents of the zip file, and copy the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should resemble the image below.


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

import board
from rainbowio import colorwheel

from adafruit_is31fl3741.adafruit_rgbmatrixqt import Adafruit_RGBMatrixQT
import adafruit_is31fl3741

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
is31 = Adafruit_RGBMatrixQT(i2c, allocate=adafruit_is31fl3741.PREFER_BUFFER)
is31.set_led_scaling(0xFF)
is31.global_current = 0xFF
# print("Global current is: ", is31.global_current)
is31.enable = True
# print("Enabled? ", is31.enable)

wheeloffset = 0
while True:
    for y in range(9):
        for x in range(13):
            is31.pixel(x, y, colorwheel((y * 13 + x) * 2 + wheeloffset))
    wheeloffset += 1
    is31.show()
```

The matrix lights up in a slow-moving rainbow pattern!

That's all there is to using the IS31FL3741 LED matrix driver with CircuitPython!

Python Docs

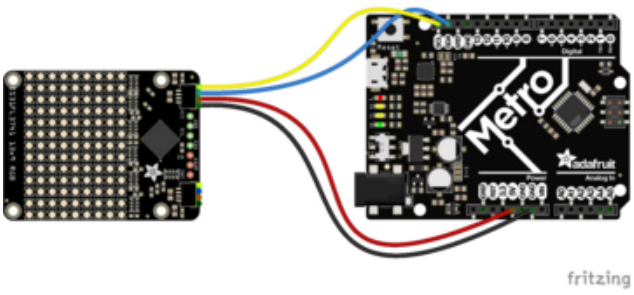
Python Docs ()

Arduino

Using the IS31FL3741 with Arduino involves wiring up the matrix to your Arduino-compatible microcontroller, installing the Adafruit IS31FL3741 () library, and running the provided example code.

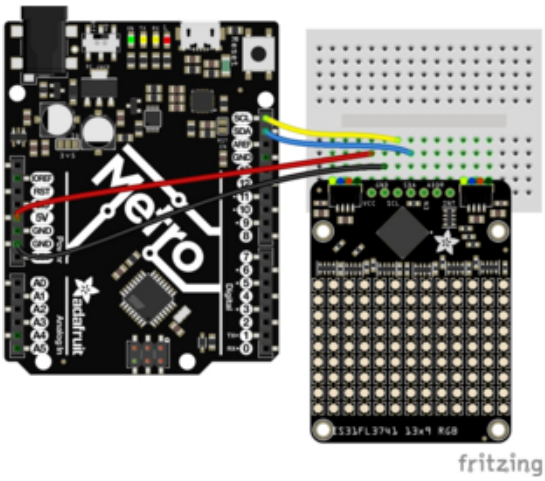
I2C Wiring

Here is how to wire up the matrix using one of the STEMMA QT () connectors. The examples show a Metro but wiring will work the same for an Arduino or other compatible board.



Connect matrix VCC (red wire) to Arduino 5V if you are running a 5V board Arduino (Uno, etc.). If your board is 3V, connect to that instead.
Connect matrix GND (black wire) to Arduino GND
Connect matrix SCL (yellow wire) to Arduino SCL
Connect matrix SDA (blue wire) to Arduino SDA

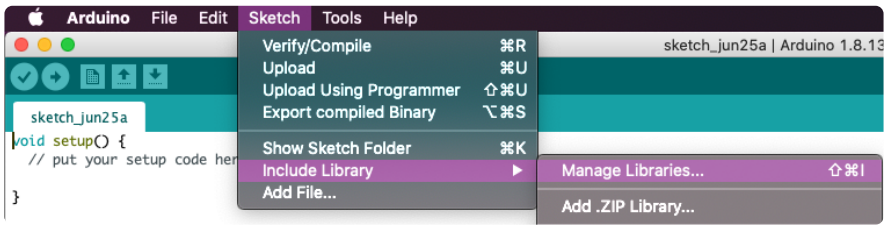
Here is how to wire the matrix to a board using a solderless breadboard:



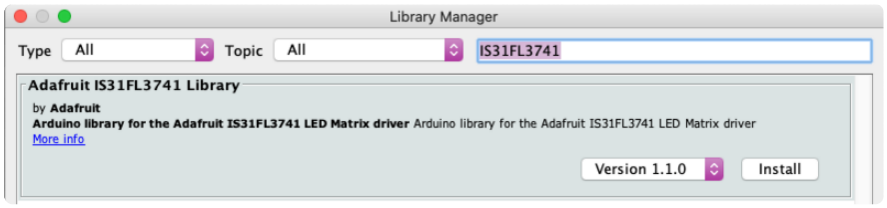
Connect matrix VCC (red wire) to Arduino 5V if you are running a 5V board Arduino (Uno, etc.). If your board is 3V, connect to that instead.
Connect matrix GND (black wire) to Arduino GND
Connect matrix SCL (yellow wire) to Arduino SCL
Connect matrix SDA (blue wire) to Arduino SDA

Library Installation

You can install the Adafruit IS31FL3741 library for Arduino using the Library Manager in the Arduino IDE.



Click the Manage Libraries ... menu item, search for IS31FL3741 , and select the Adafru it IS31FL3741 library, and install the latest version:



If asked about dependencies, click "Install all".

Load Rainbow Example

Open up File -> Examples -> Adafruit IS31FL3741 Library -> qtmatrix-rgbswirl

```
// Rainbow swirl example for the Adafruit IS31FL3741 13x9 PWM RGB LED
// Matrix Driver w/STEMMA QT / Qwiic connector. This is the simplest
// version and should fit on small microcontrollers like Arduino Uno.
// Tradeoff is that animation isn't always as smooth as seen in the
// buffered example. Each LED changes state immediately when accessed,
// there is no show() or display() function as with NeoPixels or some
// OLED screens.

#include <Adafruit_IS31FL3741.h>

Adafruit_IS31FL3741_QT ledmatrix;
// If colors appear wrong on matrix, try invoking constructor like so:
// Adafruit_IS31FL3741_QT ledmatrix(IS3741_RBG);

// Some boards have just one I2C interface, but some have more...
TwoWire *i2c = &Wire; // e.g. change this to &Wire1 for QT Py RP2040

void setup() {
  Serial.begin(115200);
  Serial.println("Adafruit QT RGB Matrix Simple RGB Swirl Test");

  if (! ledmatrix.begin(IS3741_ADDR_DEFAULT, i2c)) {
    Serial.println("IS41 not found");
    while (1);
  }

  Serial.println("IS41 found!");

  // By default the LED controller communicates over I2C at 400 KHz.
  // Arduino Uno can usually do 800 KHz, and 32-bit microcontrollers 1 MHz.
```

```
  i2c->setClock(800000);

  // Set brightness to max and bring controller out of shutdown state
  ledmatrix.setLEDscaling(0xFF);
  ledmatrix.setGlobalCurrent(0xFF);
  Serial.print("Global current set to: ");
  Serial.println(ledmatrix.getGlobalCurrent());
  ledmatrix.enable(true); // bring out of shutdown
}

uint16_t hue_offset = 0;

void loop() {
  uint32_t i = 0;
  for (int y=0; y<ledmatrix.height(); y++) {
    for (int x=0; x<ledmatrix.width(); x++) {
      uint32_t color888 = ledmatrix.ColorHSV(i * 65536 / 117 + hue_offset);
      uint16_t color565 = ledmatrix.color565(color888);
      ledmatrix.drawPixel(x, y, color565);
      i++;
    }
  }

  hue_offset += 256;

  ledmatrix.setGlobalCurrent(hue_offset / 256); // Demonstrate global current
}
```

After opening the demo file, upload to your Arduino wired up to the matrix. Once you upload the code, you'll see the matrix light up in a rainbow pattern!

This demo does a simple example showing how to set the color of each pixel using `ledmatrix.drawPixel(x, y, color565)` - where `x` and `y` are the 0-12 and 0-8 indexed location indicator, and `color565` is the 16-bit packed color. We use `ledmatrix.ColorHSV` to get a rainbow color across the spectrum.

Load GFX Example

Open up File -> Examples -> Adafruit IS31FL3741 Library -> qtmatrix-text

```
// Scrolling text example for the Adafruit IS31FL3741 13x9 PWM RGB LED
// Matrix Driver w/STEMMA QT / Qwiic connector. This is the simplest
// version and should fit on small microcontrollers like Arduino Uno.
// Tradeoff is that animation isn't always as smooth as seen in the
// buffered example. Each LED changes state immediately when accessed,
// there is no show() or display() function as with NeoPixels or some
// OLED screens.

#include <Adafruit_IS31FL3741.h>

Adafruit_IS31FL3741_QT matrix;
// If colors appear wrong on matrix, try invoking constructor like so:
// Adafruit_IS31FL3741_QT matrix(IS3741_RBG);

// Some boards have just one I2C interface, but some have more...
TwoWire *i2c = &Wire; // e.g. change this to &Wire1 for QT Py RP2040

char text[] = "ADAFRUIT!";   // A message to scroll
int text_x = matrix.width(); // Initial text position = off right edge
```

```
int text_y = 1;
int text_min;                // Pos. where text resets (calc'd later)

void setup() {
  Serial.begin(115200);
  Serial.println("Adafruit QT RGB Matrix Scrolling Text Test");

  if (! matrix.begin(IS3741_ADDR_DEFAULT, i2c)) {
    Serial.println("IS41 not found");
    while (1);
  }

  Serial.println("IS41 found!");

  // By default the LED controller communicates over I2C at 400 KHz.
  // Arduino Uno can usually do 800 KHz, and 32-bit microcontrollers 1 MHz.
  i2c->setClock(800000);

  // Set brightness to max and bring controller out of shutdown state
  matrix.setLEDscaling(0xFF);
  matrix.setGlobalCurrent(0xFF);
  Serial.print("Global current set to: ");
  Serial.println(matrix.getGlobalCurrent());

  matrix.fill(0);
  matrix.enable(true); // bring out of shutdown
  matrix.setRotation(0);
  matrix.setTextWrap(false);

  // Get text dimensions to determine X coord where scrolling resets
  uint16_t w, h;
  int16_t ignore;
  matrix.getTextBounds(text, 0, 0, &ignore, &ignore, &w, &h);
  text_min = -w; // Off left edge this many pixels
}

void loop() {
  matrix.setCursor(text_x, text_y);
  for (int i = 0; i < (int)strlen(text); i++) {
    // set the color thru the rainbow
    uint32_t color888 = matrix.ColorHSV(65536 * i / strlen(text));
    uint16_t color565 = matrix.color565(color888);
    matrix.setTextColor(color565, 0); // backound is '0' to erase previous text!
    matrix.print(text[i]); // write the letter
  }

  if (--text_x < text_min) {
    text_x = matrix.width();
  }

  delay(25);
}
```

After opening the demo file, upload to your Arduino wired up to the matrix. Once you upload the code, a scrolling rainbow ADAFRUIT! will show up on your matrix!

This demo uses more of the Adafruit_GFX library to draw text, check out the guide for Adafruit GFX to learn how to draw circles, lines, triangles, etc! ()

Adafruit GFX Library

Adafruit GFX Library ()

Arduino Docs

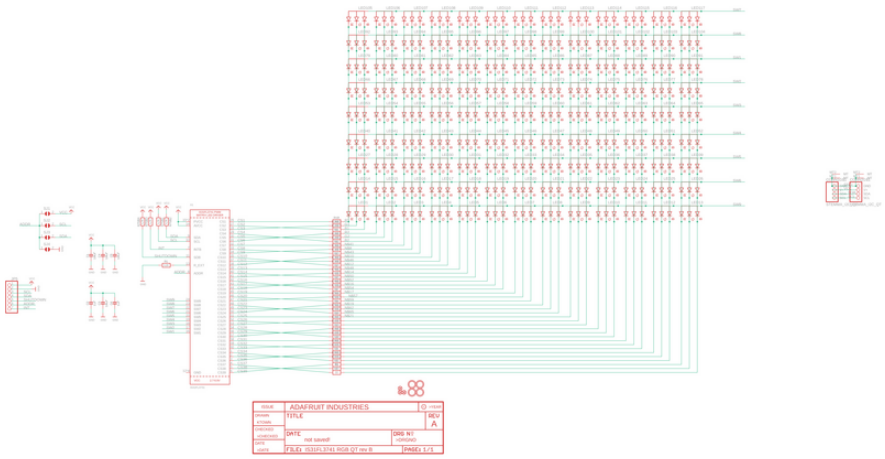
Arduino Docs ()

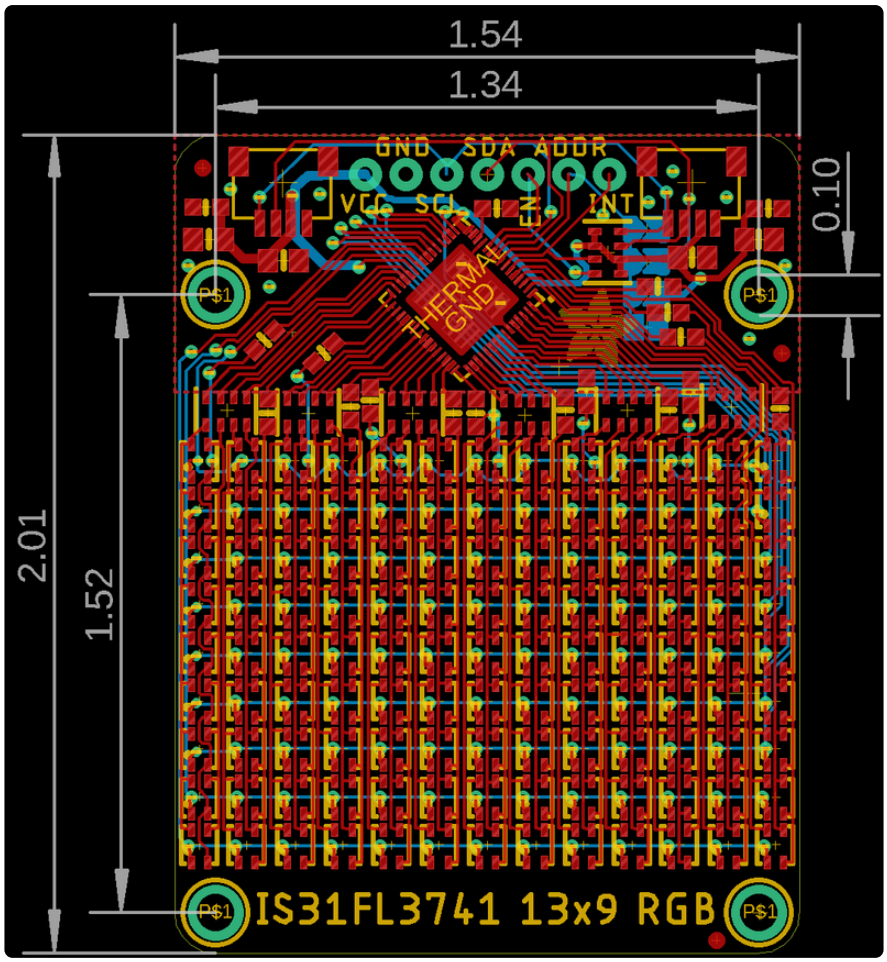
Downloads

Files

- IS31FL3741 Datasheet ()
- EagleCAD PCB files on GitHub ()
- Matrix Fritzing object in the Adafruit Fritzing Library ()
- 3D Models on GitHub ()

Schematic and Fab Print for LED Matrix




1.54
1.34
2.01
1.52
0.10
GND SDA ADDR
VCC SCL
INT
THERMAL GND
P$1
P$1
P$1
P$1
IS31FL3741 13x9 RGB