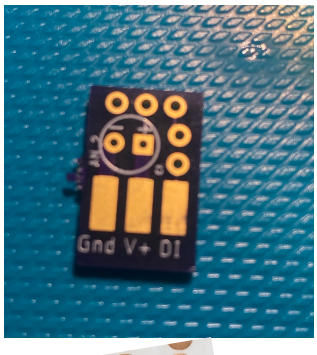
21M.370 Digital Instrument Design

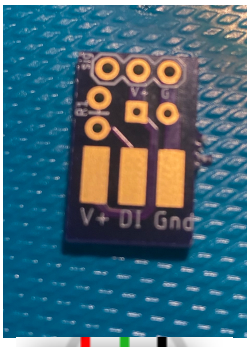
LED strip tutorial

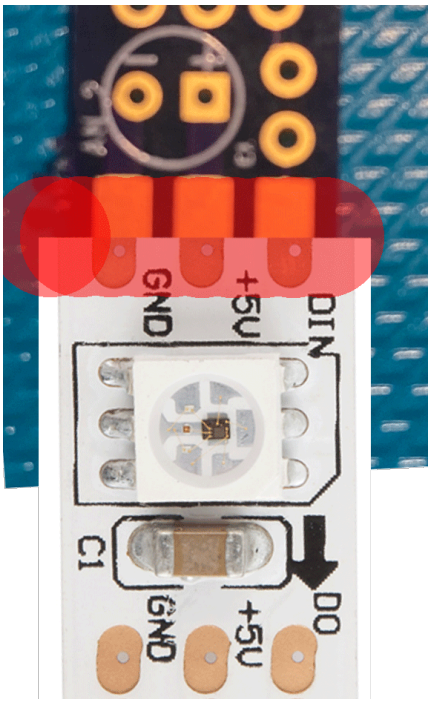
This document covers how to use the an addressable LED strip. These come in several formats and shapes. The PCB that comes with your kit is designed to work with two of the most common formats, and the code for the ESP32 and Python should work with any addressable LED strip.

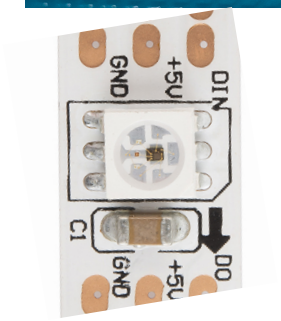
PCB

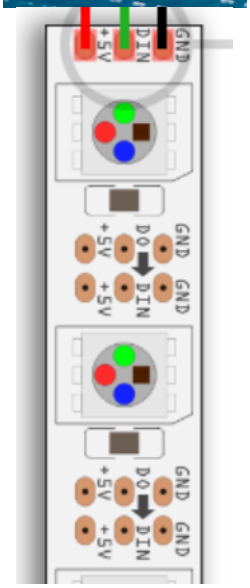
The PCB has two sides, with the two most common pinouts. Look carefully at your LED strip to see which side of the PCB will work better for your strip.





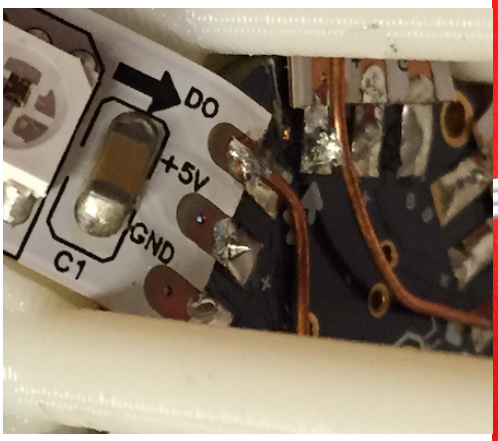


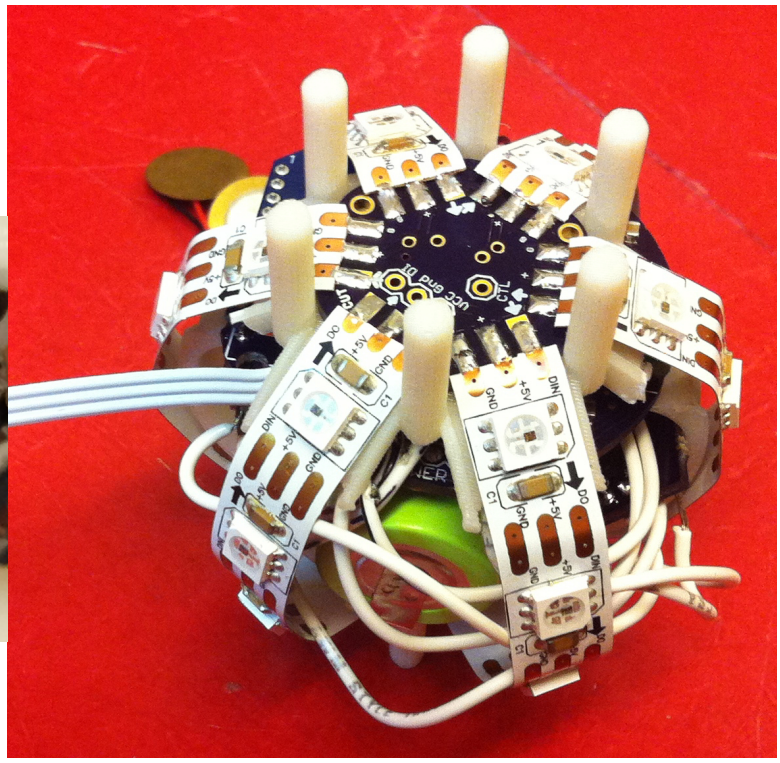




The pads are designed for the strips to be directly soldered to them. I recommend wrapping a piece of strong tape over the solder joint to the edge of the PCB in order to add mechanical strength.

Here are some pics of another project that used a similar approach to mounting LED strips to PCBs.



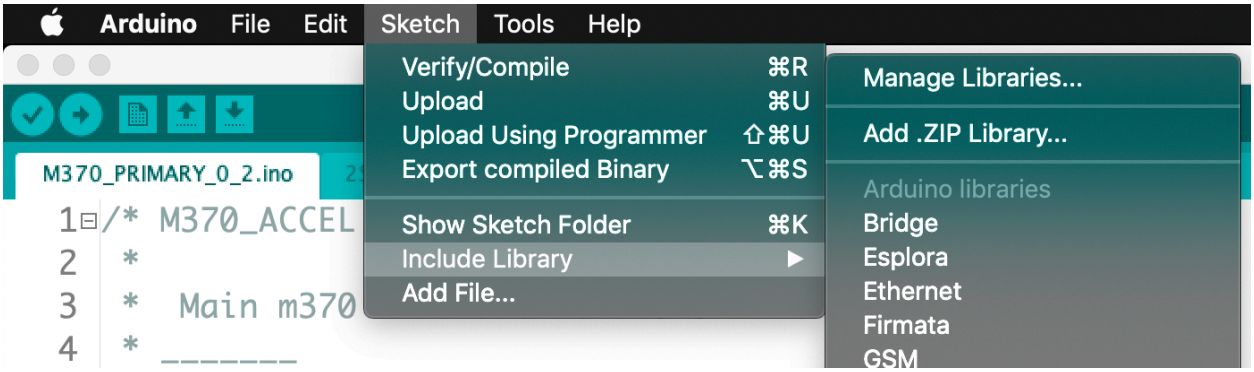


You can always just use wires to make the connection from the led strip to your microcontroller, but using the PCB may help make a more stable connection. As always, the wired connections are always the weakest points!

You should be able to use any pin our PCB to send data to the LEDs. We will cover the details below, but for now connect power and GND to any 3v and GND pins, and the DI to any available IO pin.

ESP32 Support

The easiest way to use these LEDs with the ESP32 is to use an external library. Up to this point I've been including library support for peripherals like the LSM6DS3 and MPR121 within the same folder as the arduino sketch, which is why they show up as tabs. We can also download these libraries and store them in the Arduino library folder (typically /User/documents/Arduino/libraries). The easiest way to do this is to use the Arduino library manager.



To do this, go to the sketch menu and select sketch->Include Library->Manage Libraries… Within the library manager search for FastLED and install the default version by Daniel Garcia.

Once the library is installed you will see a list of examples for fastLED under the file menu. You may have to scroll down quite a ways to find them. Open the blink example.

```
// How many leds in y
#define NUM_LEDS 8

// For led chips like
// need to define DAT
// ground, and power)
// Clock pin only nee
#define DATA_PIN 13
#define CLOCK_PIN 0
```

Within the example sketch, change the number of LEDs and dataPin to the correct values. You will also need to make sure one of the lines in the setup function are uncommented. This is where it gets tricky - make sure you know exactly what kind of LED strip you are using so you can select the right one!

```
void setup() {
    // Uncomment/edit one of the following lines for your leds arrangement.
    // ## Clockless types ##
    FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds, NUM_LEDS);  // GRB ordering is assumed
    // FastLED.addLeds<SM16703, DATA_PIN, RGB>(leds, NUM_LEDS);
    // FastLED.addLeds<TM1829, DATA_PIN, RGB>(leds, NUM_LEDS);
```

75% of the time the default NEOPIXEL setting will be correct, but if it doesn't work take a closer look at the description of your strip and try some other options. Also, occasionally strips will come with the colors in different orders but that is easy to fix in software.

If the blink sketch works, try looking through the different examples to see how they work. Some of them use different approaches to defining boards. FYI WS2811 or WS2812 LEDS are equivalent to NEOPIXELs.

Working with our system

Up to this point most of what we have been doing has been sending data from the ESP32 and routing it to Python. To control the LEDs we need to send data the other direction. Luckily it is pretty easy, and a working implementation is included using the following files:

ledMinimal.pd
370_LED_STRIP.py
M370_LED_STRIP.ino

Check out ledExample.pd to see an automated lighting effect.

The key to the PD patch is the OSC_OUTPUT subpatch, which sends the data to Python and must be included in your main patch (or opened as a separate file I suppose). We send data to OSC_OUTPUT using the 's m370_OSC_SEND' object. Data is always in format:

< led# >< red >< green >< blue >

with the led number being 0-indexed and color values are always 0-255.