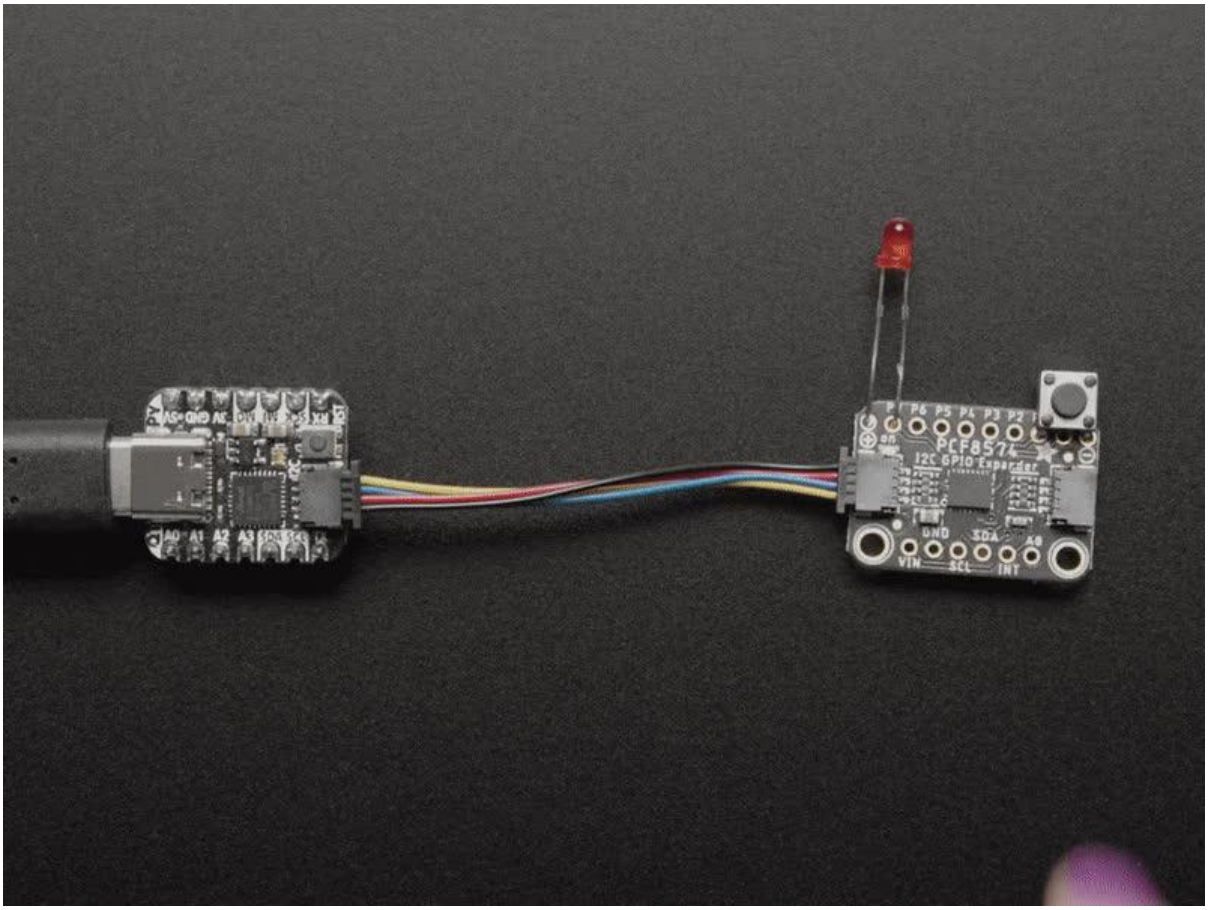
Adafruit PCF8574 I2C GPIO Expander

Created by Kattni Rembor

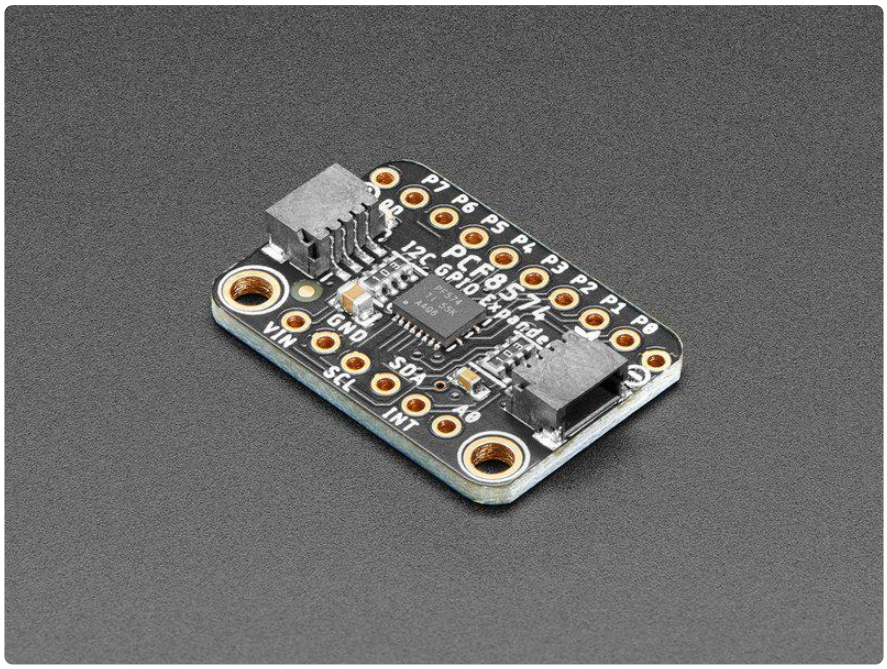


https://learn.adafruit.com/adafruit-pcf8574

Last updated on 2022-08-03 03:16:01 PM EDT

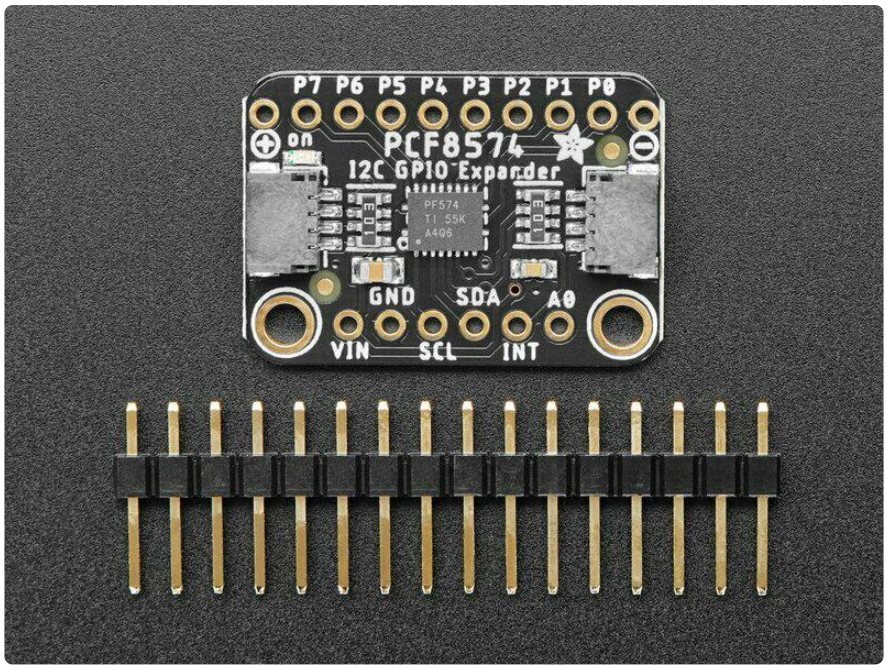
Table of Contents

Overview



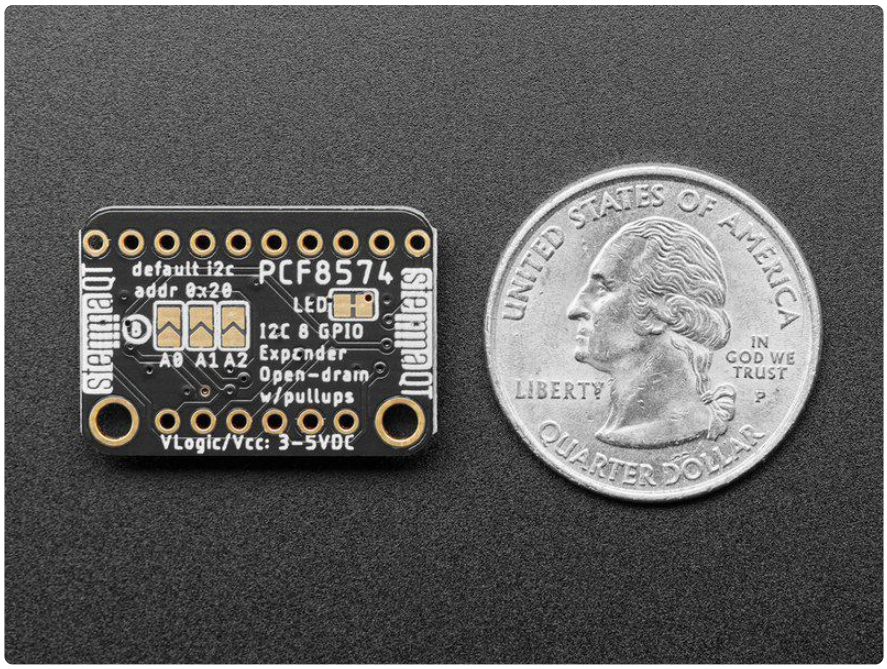
GPIO expanders work like this: you have a board with some number of GPIO but not enough for your project - maybe you need more buttons or LEDs. You could upgrade to a board with massive number of GPIO like the Grand Central (https://adafru.it/QtD), or you could pop on one of these boards.

Connect it over I2C and then you can send/receive I2C commands to control the GPIO pins to write and read them. It's going to be slower than direct GPIO access, but maybe that doesn't matter if it takes a millisecond instead of a microsecond. You only need the two I2C pins, and you can even share the I2C port with other sensors and devices. Heck, you can even add more expanders for massive I/O control!



The PCF8574 is a common, and slightly unusual I2C expander for folks who are used to the MCP230xx series (https://adafru.it/10kC):

- First up, its very affordable - who doesn't love that?
- It has 8 I/O pins
- Three I2C address select jumpers mean up to 8 expanders to one bus for 64 total GPIO added
- Each pin can be an input with light pull-up or an output sink
- IRQ output will automatically alert you when input pins change value
- This chip does not have a pin direction register. You cannot set the pins to be input or output - instead each pin has two possible states. Basically you can think of it as an open-drain output with a 100K resistor pull-up built in.
- Option one: Lightly pulled up 'input' - by default it will read as a high logic level, but connecting the GPIO to ground will cause it to read as a low logic level.
- Option two: Strong 20mA low-driving transistor sink output. This means the output is 'forced' to be low and will always read as a low logic level.



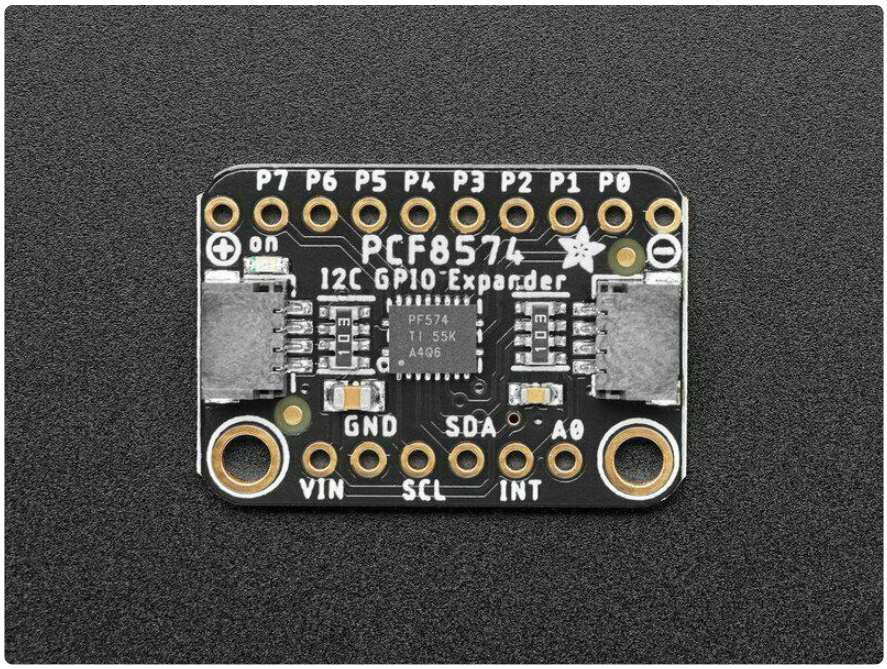
The pin direction / state thing is a little odd but it actually works fine for many purposes as long as you know what to expect.

For example, if you want to read a button or switch, connect one side to the PCF and the other side to ground. Then set the pin to 'light pull-up input' When the button is pressed it will read low, when released it will read high.

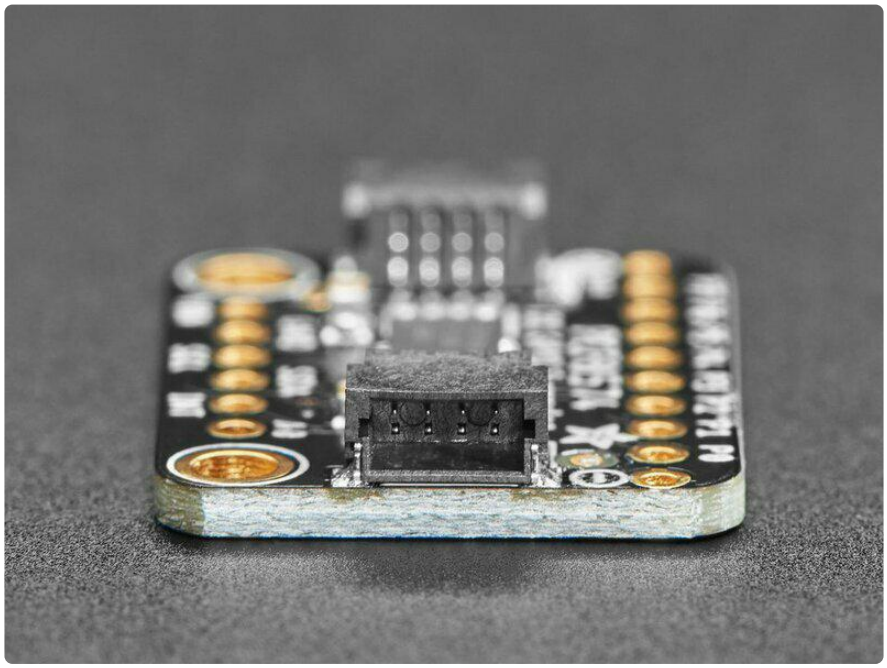
If you want to control an LED, connect the anode to positive voltage through a resistor. When the PCF pin is set to 'light pull-up input' the LED will be off. When the PCF pin is set to 'strong ground output' the LED will connect to ground and turn on.

If you want to send a GPIO output logic level to some other device or peripheral, the light pull-up acts as high logic out, the strong ground output acts as low logic out.

If you want to receive a GPIO input logic level, set the pin to light pull-up and then read the pin to determine if the GPIO input is high or low.



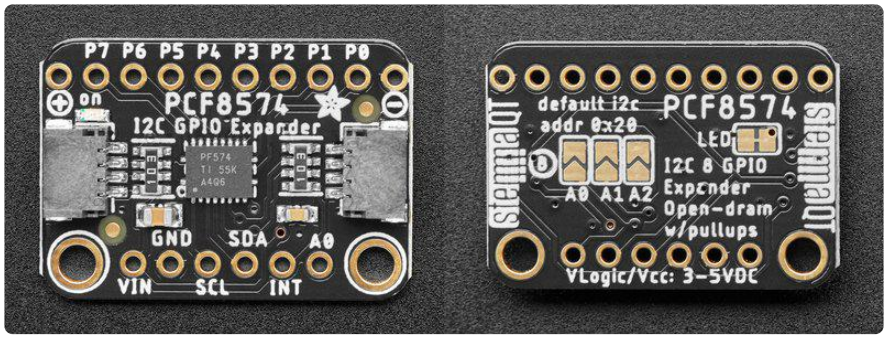
Basically, the only thing to watch for is you cannot drive an LED that is expecting the expander GPIO to go high to turn on the LED, or connect a button input to a positive voltage without adding an additional pull-down resistor. If this is a bit confusing, worry not - all this stuff is taken care of for you in our Arduino PCF8574 library (https://adafru.it/10kD) or CircuitPython/Python PCF8574 library (https://adafru.it/10kE) - you can pretend it has input/output modes and the library will fake out what you are expecting.



To get you going fast, we spun up a custom-made PCB in the STEMMA QT form factor (https://adafru.it/LBQ), making it easy to interface with. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun

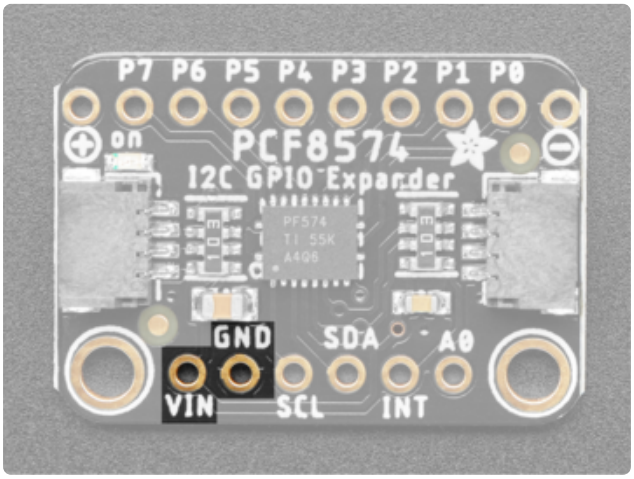
Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless connections between your development board and the PCF8574 or to chain it with a wide range of other sensors and accessories using a compatible cable (https://adafru.it/JnB).

Pinouts



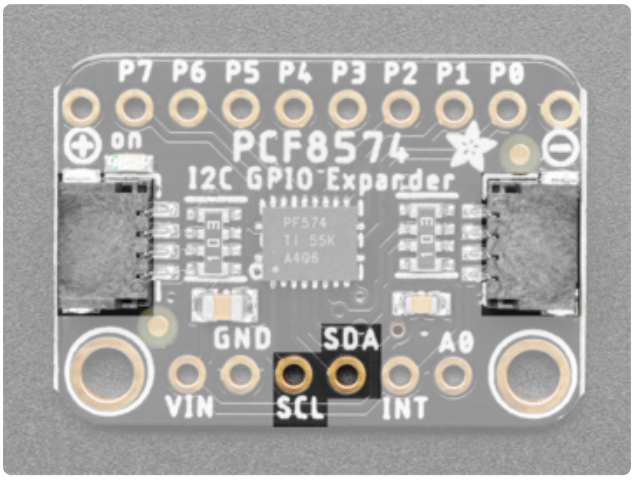
The default I2C address is 0x20.

Power Pins



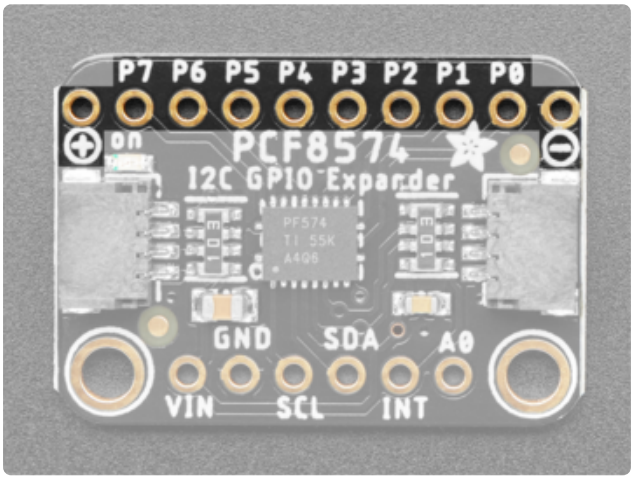
VIN - This is the power pin. To power the board, give it the same power as the logic level of your microcontroller - i.e. for a 5V micro like Arduino, use 5V, or for a 3V micro like a Feather, use 3V.
GND - This is common ground for power and logic.

I2C Logic Pins



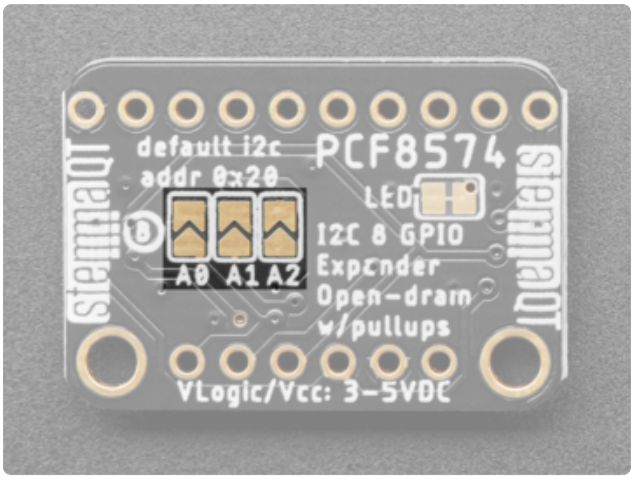
SCL - I2C clock pin, connect to your microcontroller's I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
SDA - I2C data pin, connect to your microcontroller's I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
STEMMA QT (https://adafru.it/Ft4) - These connectors allow you to connect to development boards with STEMMA QT connectors, or to other things, with various associated accessories (https://adafru.it/JRA).

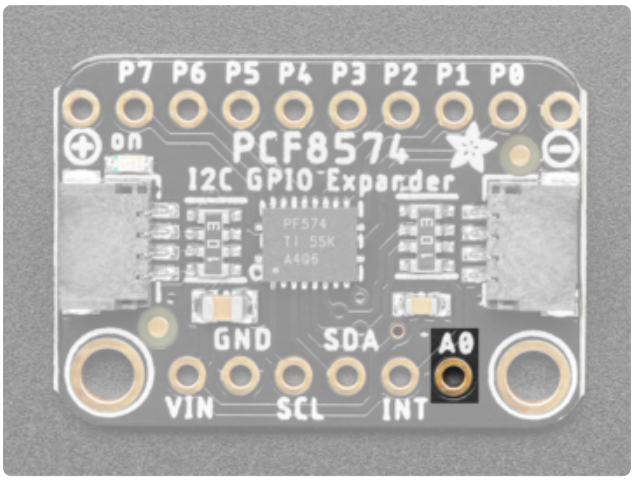
Expander Pins



Along the top are the 8 I/O pins for expanding your project. They are labeled above the pads, left-to-right as P7 through P0.
On the far left is the positive voltage for the expander I/O pins. It is labeled below the pad with a + in a circle
On the far right is the ground for the expander I/O pins. It is labeled below the pad with a - in a circle.

Address Pin and Jumpers





On the back of the board are three address jumpers, labeled A0, A1, and A2. These jumpers allow you to chain up to 8 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumpers "closed" by connecting the two pads.
On the front of the board is one address pin, labeled A0. Just like the jumpers, this pin allows you to change the I2C address to connect multiple boards by connecting it to VIN.

The default I2C address is 0x20. The other address options can be calculated by “adding” the A0/A1/A2 to the base of 0x20.

A0 sets the lowest bit with a value of 1, A1 sets the next bit with a value of 2 and A2 sets the next bit with a value of 4. The final address is 0x20 + A2 + A1 + A0 which would be 0x27.

So for example if A2 is soldered closed and A0 is soldered closed, the address is 0x2 0 + 4 + 1 = 0x25.

If only A0 is soldered closed, the address is 0x20 + 1 = 0x21

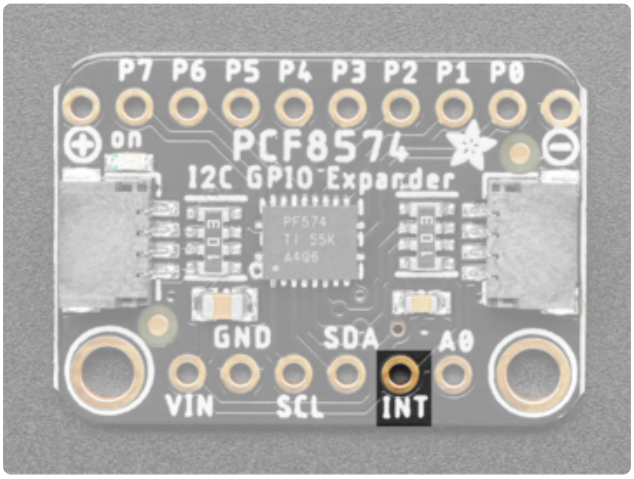
If only A1 is soldered closed, the address is 0x20 + 2 = 0x22

If only A2 is soldered closed, the address is 0x20 + 4 = 0x24

The table below shows all possible addresses, and whether the pin(s) should be high (closed) or low (open).

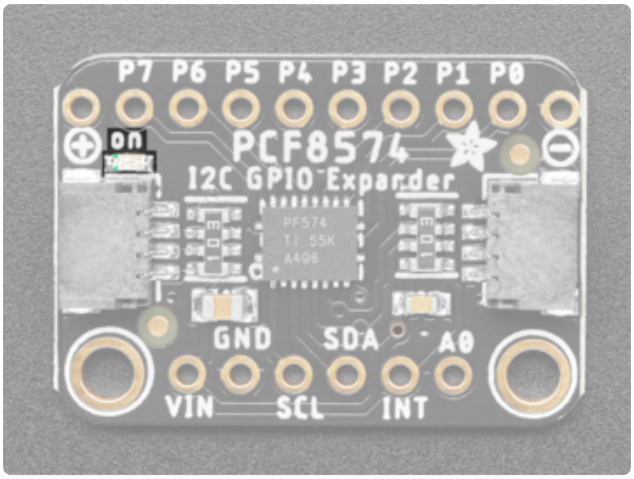
ADDR	A0	A1	A2	ADDR	A0	A1	A2
0x20	L	L	L	0x24	L	L	H
0x21	H	L	L	0x25	H	L	H
0x22	L	H	L	0x26	L	H	H
0x23	H	H	L	0x27	H	H	H

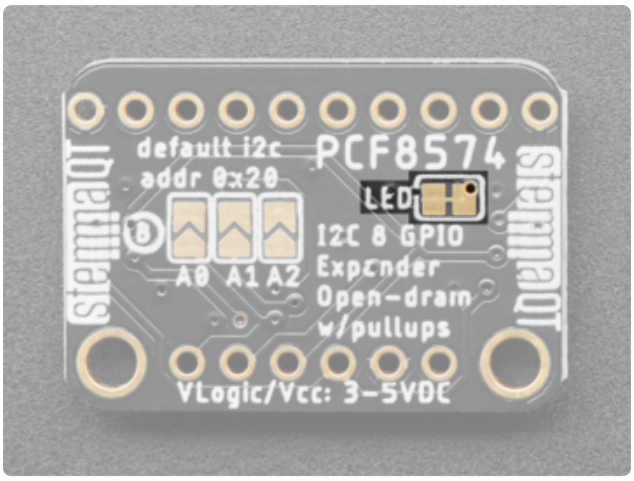
INT Pin



The INT pin is the IRQ output, which will automatically alert you when input pins change value.

On LED and LED Jumper





On the left side of the front of the board is the on LED which lights up when the board has power. It is labeled above the LED as on.
Towards the right side of the back of the board is the LED jumper. It is two pads connected by a trace, labeled as LED to the left of the pads. If you'd rather not have the on LED lit up when the board has power, you can cut the trace between the two pads. To reenable the on LED, you can bridge the pads again with solder.

Python & CircuitPython

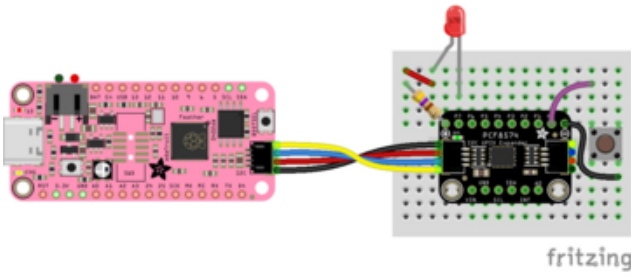
It's easy to use the Adafruit PCF8574 with Python or CircuitPython, and the Adafruit CircuitPython PCF8574 (https://adafru.it/10kE) module. This module allows you to easily write Python code that enables you to utilise the 8 I/O pins on the expander.

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 (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

First wire up a PCF8574 to your board exactly as shown below. These wiring diagrams include a button and an LED, which are necessary for the example below.

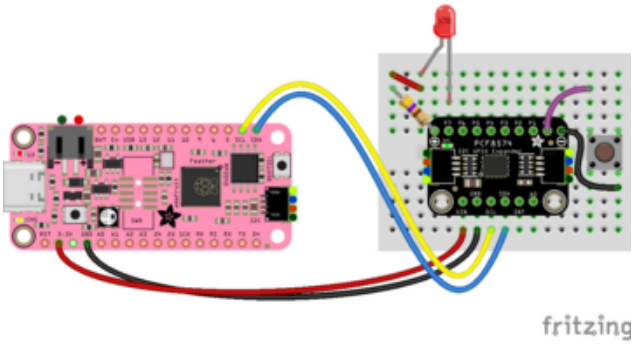
Here's an example of wiring a Feather RP2040 to the expander with I2C using one of the handy STEMMA QT (https://adafru.it/Ft4) connectors:



Feather to expander:

Simply use a STEMMA QT cable (https://adafru.it/Tff) to connect from the STEMMA QT connector on the microcontroller to the STEMMA QT connector on the breakout. Follow the steps below (https://adafru.it/10kF) to connect the LED and button.

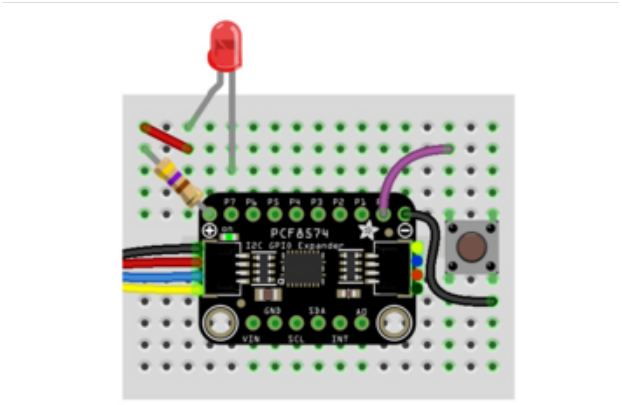
You can also use the standard 0.100" pitch headers to wire it up on a breadboard:



Feather to expander:

Feather 3V to expander VIN (red wire)
Feather GND to expander GND (black wire)
Feather SCL to expander SCL (yellow wire)
Feather SDA to expander SDA (blue wire)
Follow the steps below (https://adafru.it/10kF) to connect the LED and button.

Connect the LED and the button to the expander as follows:



LED to expander:

LED- to expander P7
LED+ to 470Ω resistor
470Ω resistor to + on top edge of expander
Button to expander:

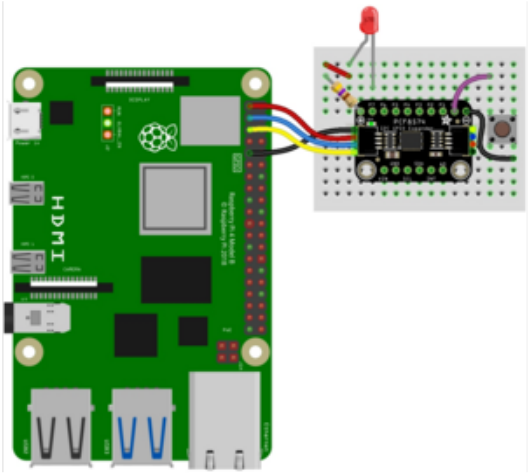
One leg of button to - on top edge of expander
Opposite leg of button to expander P0

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).

These wiring diagrams include a button and an LED, which are necessary for the example below.

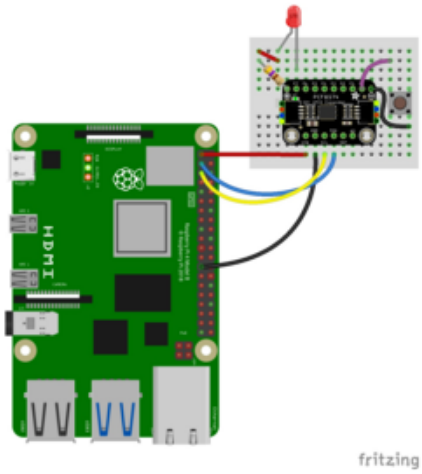
Here's the Raspberry Pi wired to the expander using I2C and a STEMMA QT (https://adafru.it/Ft4) connector.



Pi to expander:

Pi 3V to expander VIN (red wire)
Pi GND to expander GND (black wire)
Pi SCL to expander SCL (yellow wire)
Pi SDA to expander SDA (blue wire)
Follow the steps above (https://adafru.it/10kF) to connect the LED and button.

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



Pi to expander:

Pi 3V to expander VIN (red wire)
Pi GND to expander GND (black wire)
Pi SCL to expander SCL (yellow wire)
Pi SDA to expander SDA (blue wire)
Follow the steps above (https://adafru.it/10kF) to connect the LED and button.

Python Installation of PCF8574 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:

- `pip3 install adafruit-circuitpython-pcf8574`

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!

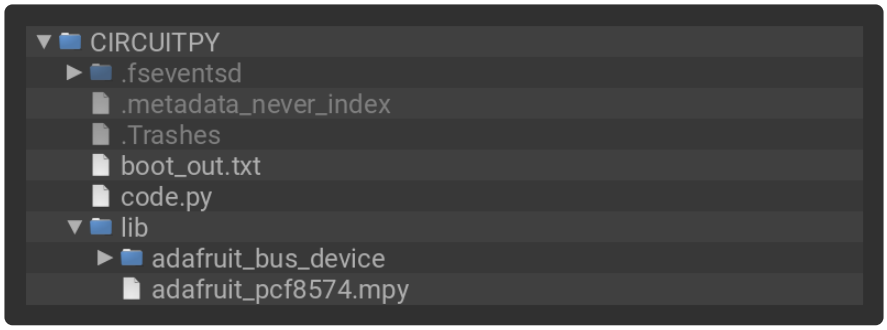
CircuitPython Usage

To use with CircuitPython, you need to first install the ADXL37x 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 folder and file:

- adafruit_bus_device/
- adafruit_pcf8574.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:

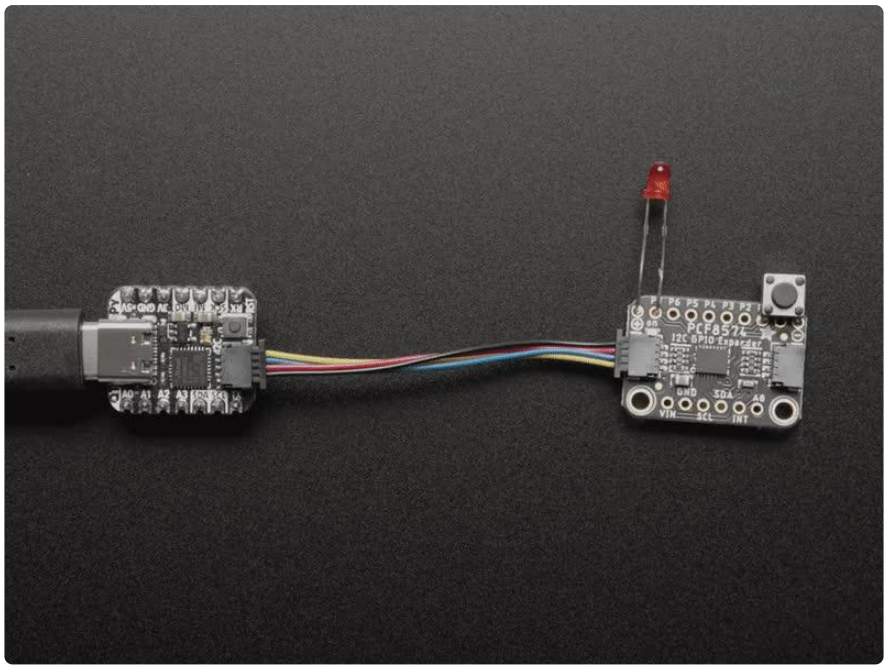
```
python3 code.py
```

Example Code

```
# SPDX-FileCopyrightText: 2017 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2022 ladyada for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

import time
import board
import digitalio
import adafruit_pcf8574

print("PCF8574 digitalio LED + button test")

i2c = board.I2C()
pcf = adafruit_pcf8574.PCF8574(i2c)

# get a 'digitalio' like pin from the pcf
led = pcf.get_pin(7)
button = pcf.get_pin(0)

# Setup pin7 as an output that's at a high logic level default
led.switch_to_output(value=True)
# Setup pin0 as an output that's got a pullup
button.switch_to_input(pull=digitalio.Pull.UP)


while True:
    led.value = button.value
    time.sleep(0.01)  # debounce
```

Now, press the button to see the LED light up.

The GIF below shows a "lazy" way to connect the button and LED to the expander. You should always include a resistor when wiring an LED!



That's all there is to using the PCF8574 with CircuitPython!

Python Docs

Python Docs (https://adafru.it/10hE)

Arduino

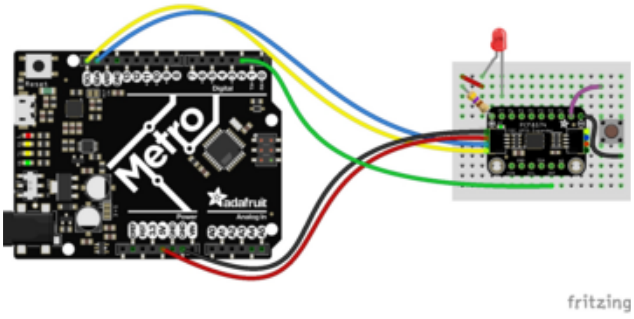
Using the PCF8574 with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit PCF8574 (https://adafru.it/10kD) library and running the provided example code.

Wiring

Wire as shown for a 5V board like an Uno. If you are using a 3V board, like an Adafruit Feather, wire the board's 3V pin to the PCF8574 VIN.

These wiring diagrams include a button and an LED, which are necessary for the example below.

Here is an Adafruit Metro wired up to the PCF8574 using the STEMMA QT connector:

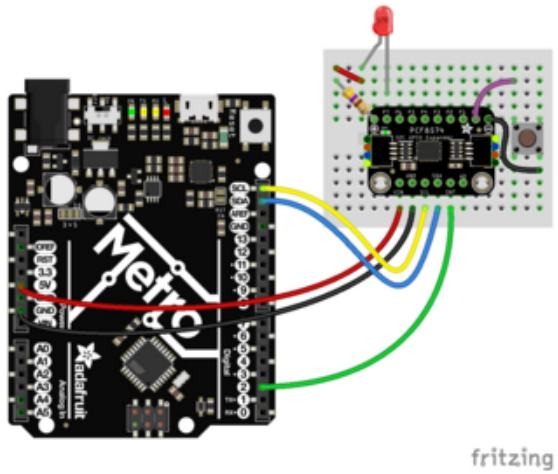


Metro to expander:

Use a STEMMA QT to male header pin cable (https://adafru.it/FA-).

Metro 5V to expander VIN (red wire)
Metro GND to expander GND (black wire)
Metro SCL to expander SCL (yellow wire)
Metro SDA to expander SDA (blue wire)
Metro 2 to expander INT (green wire)
Please follow the steps below (https://adafru.it/10ld) for LED and button wiring.

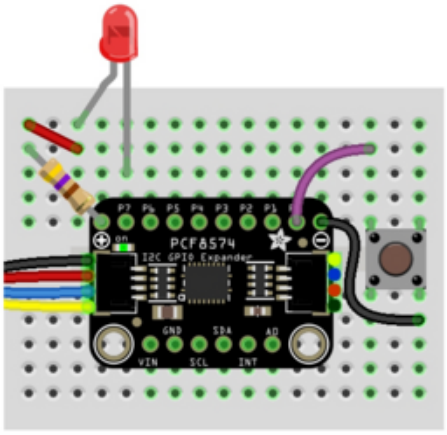
Here is an Adafruit Metro wired up using a solderless breadboard:



Metro to expander:

Metro 5V to expander VIN (red wire)
Metro GND to expander GND (black wire)
Metro SCL to expander SCL (yellow wire)
Metro SDA to expander SDA (blue wire)
Metro 2 to expander INT (green wire
Please follow the steps below (https://adafru.it/10ld) for LED and button wiring.

Connect the LED and the button to the expander as follows:



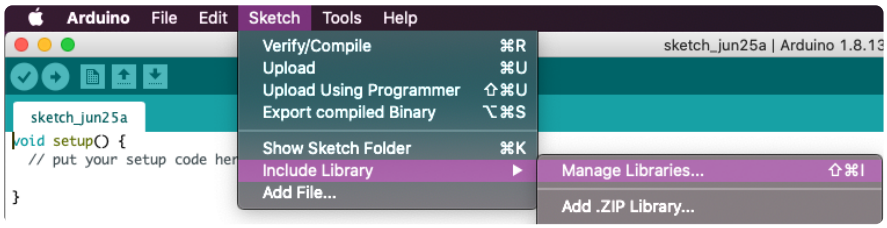
LED to expander:

LED- to expander P7
LED+ to 470Ω resistor
470Ω resistor to + on top edge of expander
Button to expander:

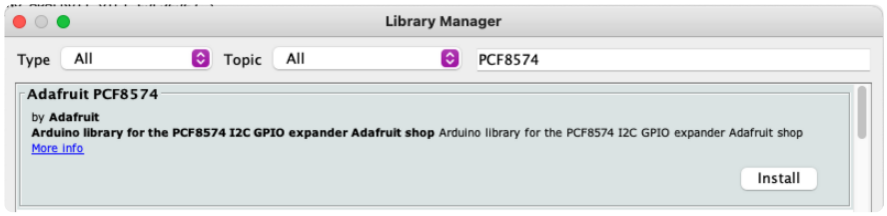
One leg of button to - on top edge of expander
Opposite leg of button to expander P0

Library Installation

You can install the PCF8574 library for Arduino using the Library Manager in the Arduino IDE.



Click the Manage Libraries ... menu item, search for PCF8574 , and select the Adafruit PCF8574 library:



When asked about dependencies, click "Install all".

Load Example

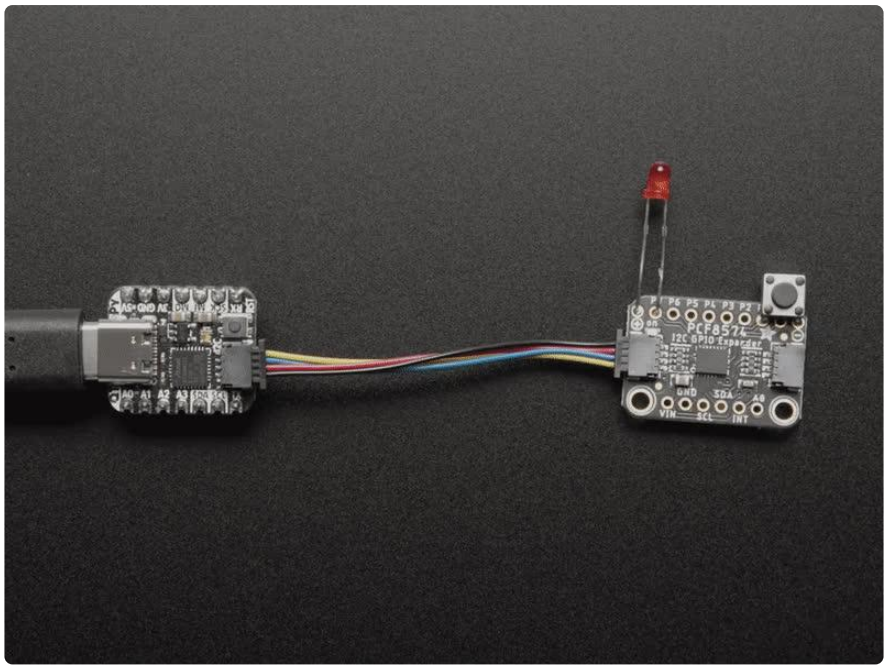
Open up File -> Examples -> Adafruit PCF8574 -> buttonledirq and upload to your Arduino wired to the sensor.

```
#include <Adafruit_PCF8574.h>

/* Example for 1 button that is connected from PCF GPIO #0 to ground,
 * and one LED connected from power to PCF GPIO #7
 * We also have the IRQ output connected to an Interrupt input pin on the
 * Arduino so we are not constantly polling from the PCF8574 expander
 */
```

```
Adafruit_PCF8574 pcf;

#define PCF_BUTTON  0  // on the GPIO expander!
#define PCF_LED     7  // on the GPIO expander!

#define ARDUINO_IRQ 2  // make sure this pin is possible to make IRQ input

void setup() {
  while (!Serial) { delay(10); }
  Serial.begin(115200);
  Serial.println("Adafruit PCF8574 button/led IRQ test");

  if (!pcf.begin(0x20, &Wire)) {
    Serial.println("Couldn't find PCF8574");
    while (1);
  }

  pcf.pinMode(PCF_BUTTON, INPUT_PULLUP);
  pcf.pinMode(PCF_LED, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  // set up the interrupt pin on IRQ signal toggle
  pinMode(ARDUINO_IRQ, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ARDUINO_IRQ), button_detect, CHANGE);
}

// We use a flag to make sure we don't enter the interrupt more than once
volatile bool in_irq = false;

// called when the button is pressed!
void button_detect(void) {
  if (in_irq) return; // we are already handling an irq so don't collide!

  in_irq = true;
  interrupts(); // Arduino UNO seems to require that we turn on interrupts for I2C
to work!
  bool val = pcf.digitalRead(PCF_BUTTON);
  pcf.digitalWrite(PCF_LED, val);
  in_irq = false;
}

void loop() {
  delay(100); // we do nothing here!
}
```

Once loaded, press the button to see the LED light up when the button is pressed.

The GIF below shows a "lazy" way to connect the button and LED to the expander. You should always include a resistor when wiring an LED!



Arduino Docs

Arduino Docs (https://adafru.it/10hF)

Downloads

Files

- PCF8574 Datasheet (https://adafru.it/10la)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/10lb)
- EagleCAD PCB files on GitHub (https://adafru.it/10lc)

Schematic and Fab Print