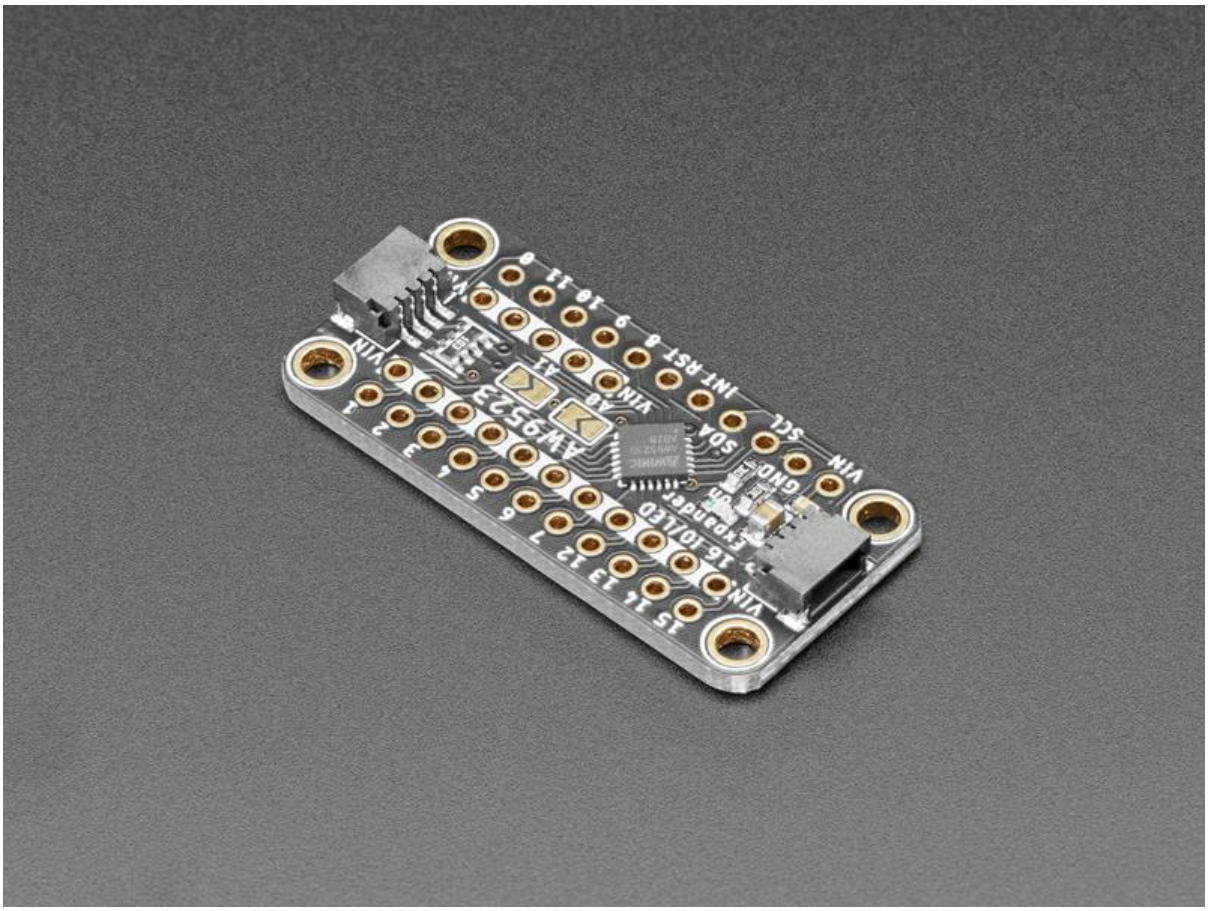
# Adafruit AW9523 GPIO Expander and LED Driver

Created by Kattni Rembor

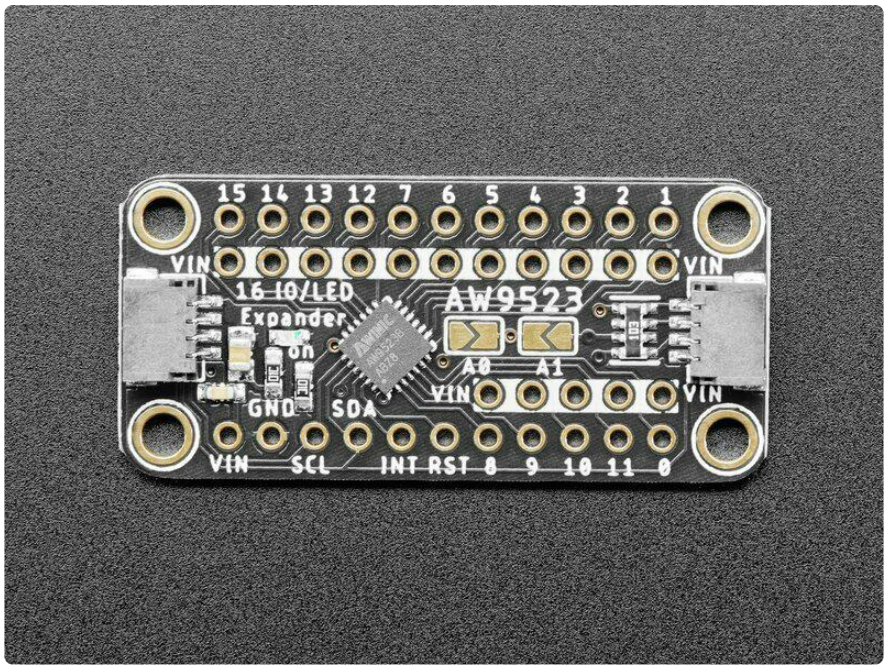



https://learn.adafruit.com/adafruit-aw9523-gpio-expander-and-led-driver

Last updated on 2022-12-01 04:02:26 PM EST

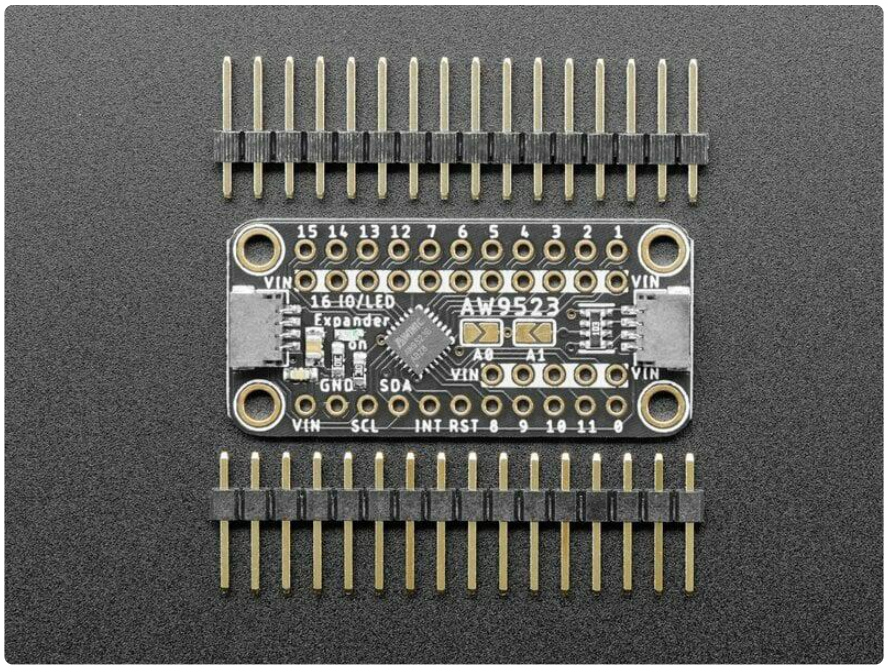
# Table of Contents

# Overview



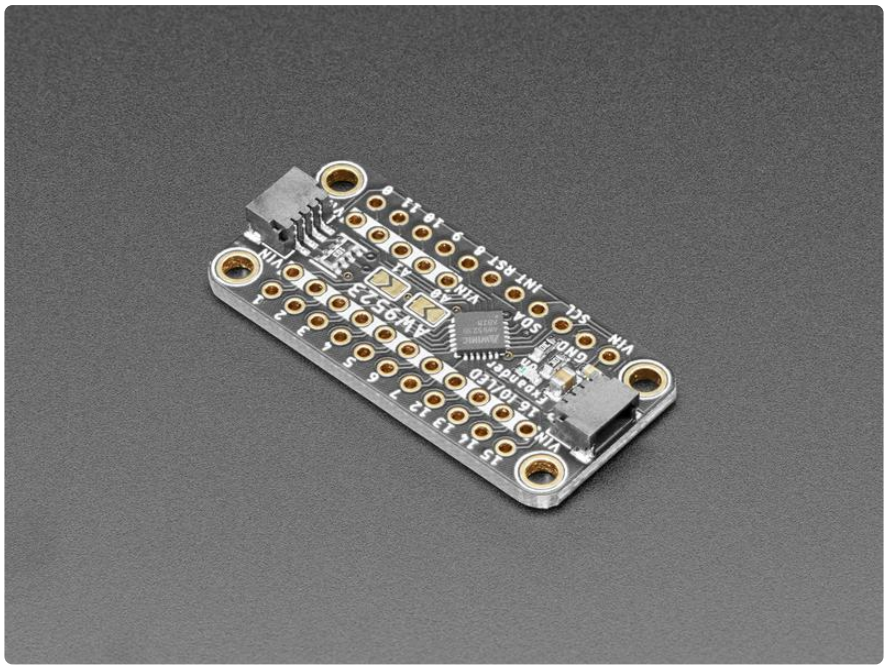

Expand your project possibilities, with the Adafruit AW9523 GPIO Expander and LED Driver Breakout - a cute and powerful I2C expander with a lot of tricks up it's sleeve.

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 (), 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!



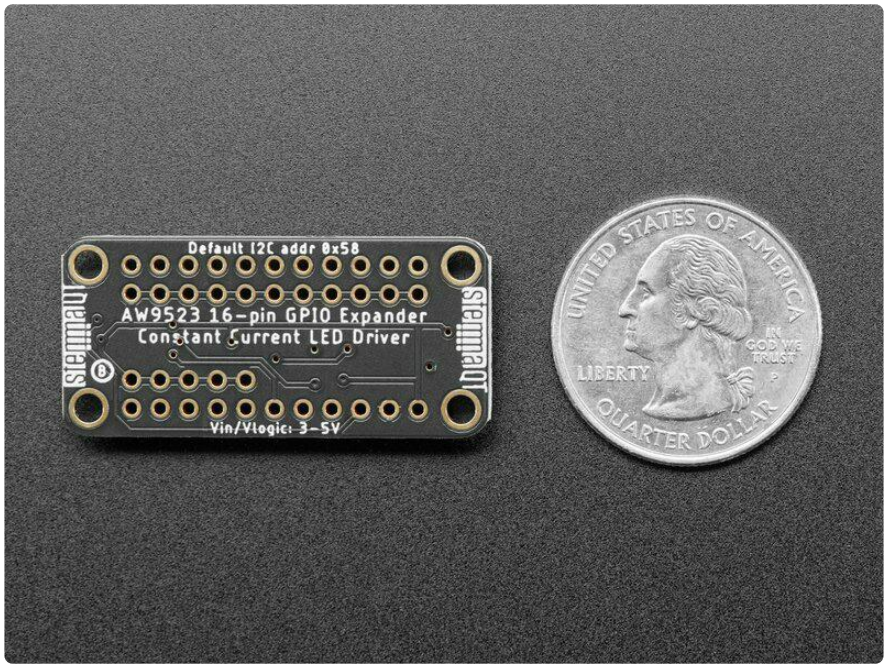

The AW9523 is a twist on the common I2C expander:

- First up, its very affordable - who doesn't love that?
- It has 16 I/O pins, that'll double most boards' pin count
- Four I2C address options, so you can connect 4 expanders to one bus
- Each pin can be an input or an output
- IRQ output can alert you when input pins change value
- This chip does not support internal pull-ups or pull-downs, you will need to add an external resistor if you need one
- However, it does have 8-bit linear constant-current LED dimming support so you can connect LEDs without resistors and have great looking dimming without PWM
- The first 8 pins can be configured as open drain (as a group)



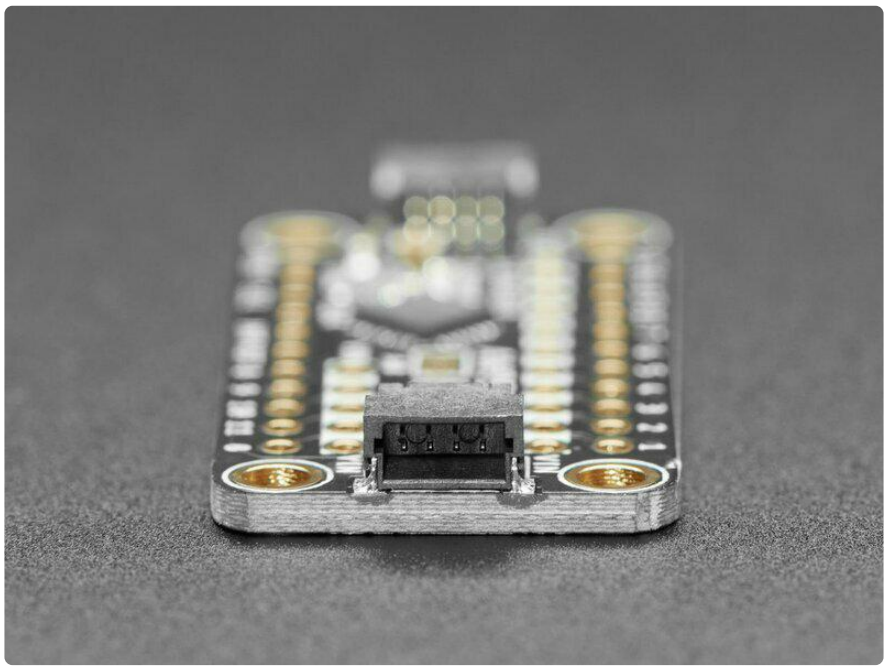
The lack of internally-configurable pull's is a bit of a bummer, but we think the expander more than makes up for it with the constant-current LED drive. If you're using an expander to add lots of controllable LEDs, this board will make it very easy. Since its constant-current, you don't need resistors in line with each LED (although it won't hurt if you do): simply connect the LED anode to one of the many VIN pads, then connect the cathode to the GPIO pin.

Of course, you can control any buttons or other I/O with the pins - we just think this board is particularly suited to LED driving. There's also an interrupt output, you can enable the pin-change IRQ for any pins so you can be notified when its time to read the I/O states.



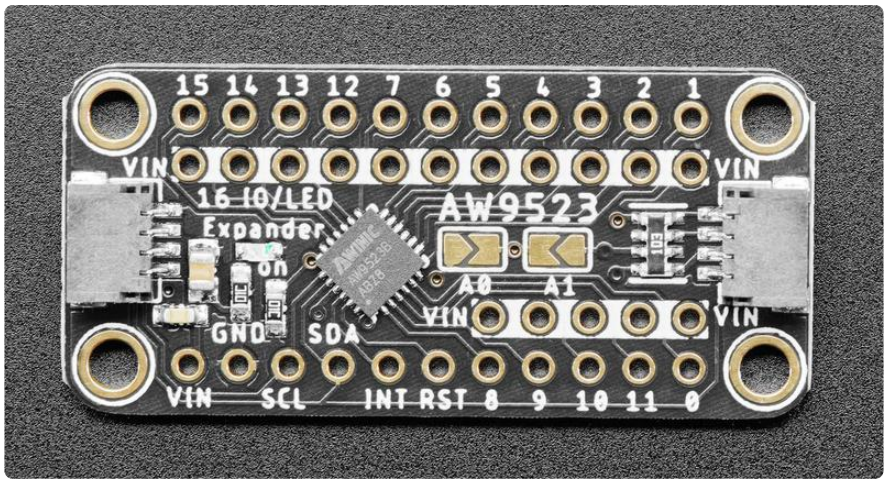
One oddity about this chip is the default I2C address determines the initial boot-state of the pins. Our libraries immediately soft-reset and configure all the pins to inputs and push-pull so you can expect the same behavior no matter what the I2C address is. However, we recommend you check the datasheet Table 1 to make sure this doesn't affect your hardware.

We've written both Arduino () and CircuitPython/Python libraries () for the AW9523, so you can get started whether you have an Arduino-compatible UNO or a Raspberry Pi 4 - or anything in between.



To get you going fast, we spun up a custom made PCB in the STEMMA QT form factor (), making it easy to interface with. The STEMMA QT connectors () on either side are compatible with the SparkFun Qwiic () I2C connectors. This allows you to make solderless connections between your development board and the AW9523 or to chain it with a wide range of other sensors and accessories using a compatible cable ( ). QT Cable is not included, but we have a variety in the shop ().

# Pinouts




## Power Pins:

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

## I2C Logic pins:

- SCL - I2C clock pin, connect to your microcontroller I2C clock line.
- SDA - I2C data pin, connect to your microcontroller I2C data line.
- STEMMA QT () - These connectors allow you to connectors to dev boards with S TEMMA QT connectors or to other things with various associated accessories ()

## GPIO Pins

- 0 - 15 - GPIO expander pins for LED drive (current-source dimming) or GPIO mode. Can be used as input or output. Supports 8-bit linear constant-current LED dimming support so you can connect LEDs without resistors. Note that this chip does not support internal pull-up or pull-downs, so you will need to add an external resistor if you need one.

## Other Pins

- INT - Interrupt output pin, can alert you when input value changes.

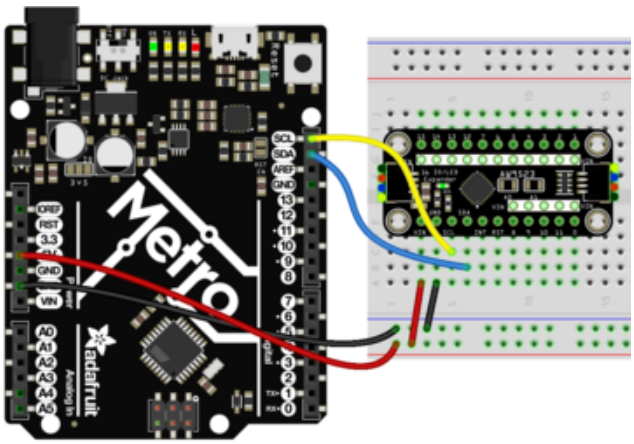
- RST - Hardware reset pin, set low to perform a hardware reset. Has 10kΩ pull-up resistor.
- A0 and A1 jumpers - These can be used to change the default I2C address from 0x58 to 0x59 (A0 shorted), 0x5A (A1 shorted) or 0x5B (both shorted!)

---

# Arduino

## Wiring

Connecting the AW9523 breakout to your Feather or Arduino is easy:

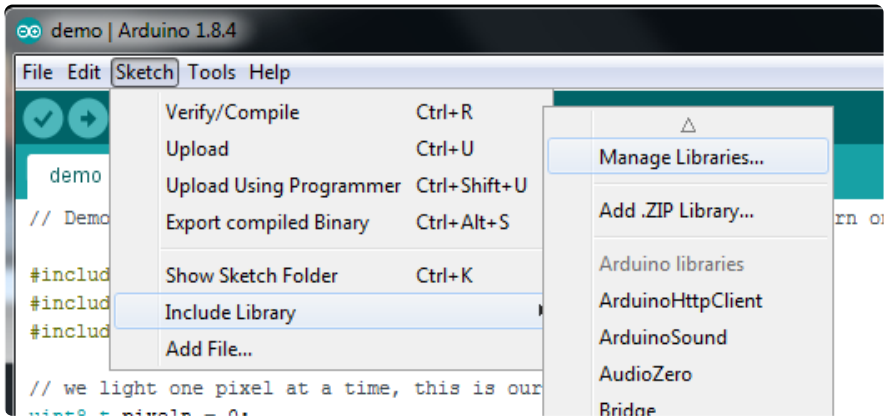


If you are running a Feather (3.3V), connect Feather 3V to board VIN
If you are running a 5V Arduino (Uno, etc.), connect Arduino 5V to board VIN
Connect Feather or Arduino GND to board GND
Connect Feather or Arduino SCL to board SCL
Connect Feather or Arduino SDA to board SDA

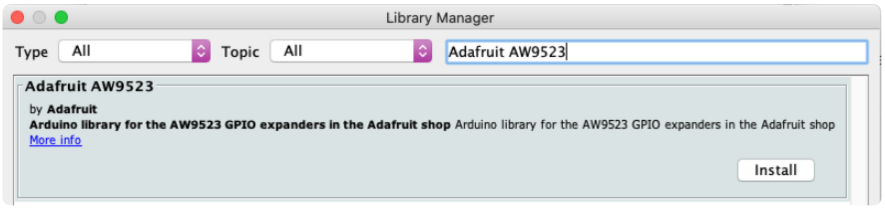
The final results should resemble the illustration above, showing an Adafruit Metro development board.

## Installation

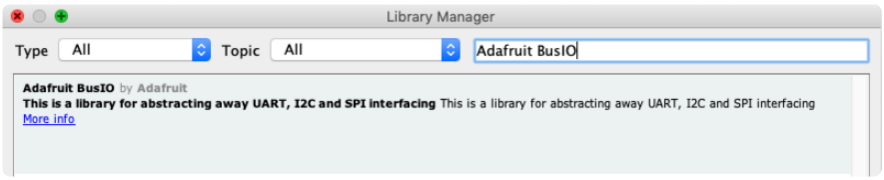
You can install the Adafruit AS9523 Library for Arduino using the Library Manager in the Arduino IDE:



Click the Manage Libraries ... menu item, search for Adafruit AW9523, and select the Adafruit AW9523 library:




Then follow the same process for the Adafruit BusIO library.

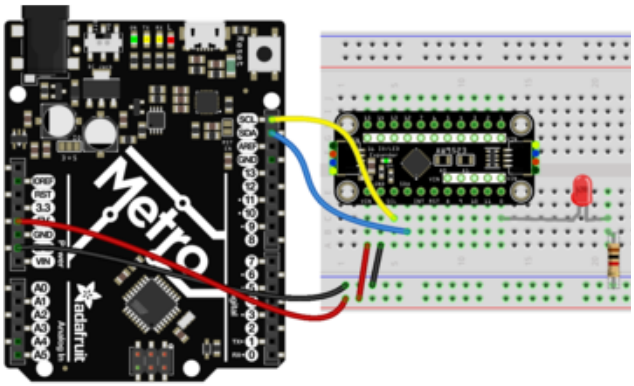



# Load Example

There are three different examples for the Adafruit AW9523 in Arduino.

## Blink

This example, unsurprisingly, blinks a single GPIO output pin. It's a good demo for setting a pin to be an output and changing the state



In this example, connect an LED + resistor to the expander pin labeled 0

Connect the anode (positive) leg of the LED to the 0 pad, then the cathode (negative) leg of the LED to a ~ 1K ohm resistor and the other end of the resistor to ground.

```
#include <Adafruit_AW9523.h>

Adafruit_AW9523 aw;

uint8_t LedPin = 0;  // 0 thru 15

void setup() {
  Serial.begin(115200);
```

```
  while (!Serial) delay(1);  // wait for serial port to open

  Serial.println("Adafruit AW9523 GPIO Expander test!");

  if (! aw.begin(0x58)) {
    Serial.println("AW9523 not found? Check wiring!");
    while (1) delay(10);  // halt forever
  }

  Serial.println("AW9523 found!");
  aw.pinMode(LedPin, OUTPUT);
}


void loop() {
  aw.digitalWrite(LedPin, HIGH);
  delay(100);
  aw.digitalWrite(LedPin, LOW);
  delay(100);
}
```

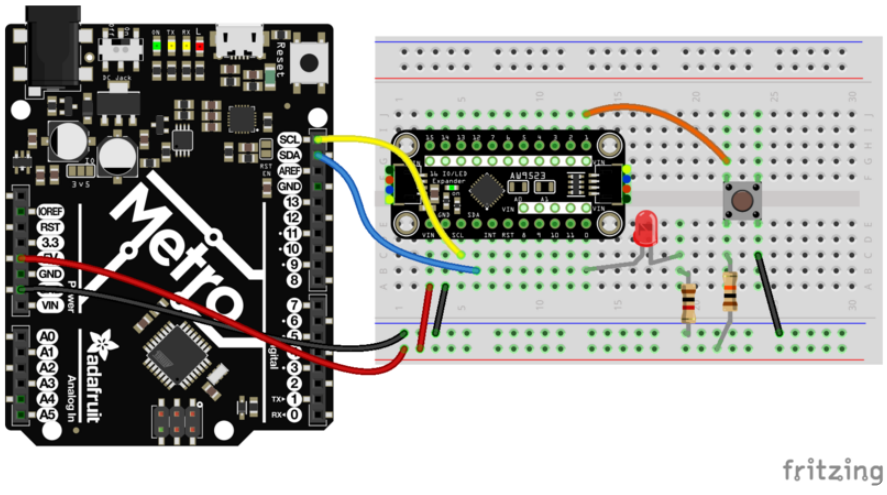
The LED connected to pin 0 will blink every 100 milliseconds!

## Button Input and LED Output

This example shows how you can create an input on the expander. Note that this expander does not support internal pull resistors

Keep the LED wired up on pin 0, with a resistor to ground, this is the output

Connect a tactile switch with one side connected to pin 1 of the expander. Add a ~10K pull-up resistor on the expander pin 1 to Vin. Connect the other side of the button to ground. This makes it so the button pin is HIGH by default, and when you press the button, the pin goes LOW




Open up File -> Examples -> Adafruit AW9523 -> ledbutton_demo and upload to your Arduino wired up to the breakout.

```
#include <Adafruit_AW9523.h>

Adafruit_AW9523 aw;

uint8_t LedPin = 0;
uint8_t ButtonPin = 1;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(1);  // wait for serial port to open

  Serial.println("Adafruit AW9523 Button + LED test!");

  if (! aw.begin(0x58)) {
    Serial.println("AW9523 not found? Check wiring!");
    while (1) delay(10);  // halt forever
  }

  Serial.println("AW9523 found!");
  aw.pinMode(LedPin, OUTPUT);
  aw.pinMode(ButtonPin, INPUT);
  aw.enableInterrupt(ButtonPin, true);
}


void loop() {
  aw.digitalWrite(LedPin, aw.digitalRead(ButtonPin));

  delay(10);
}
```

This line in the main loop does all the work:

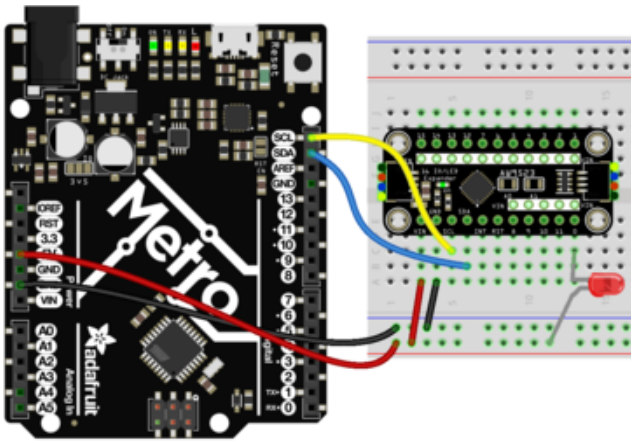
`aw.digitalWrite(LedPin, aw.digitalRead(ButtonPin));`

We read the button pin, and whatever the logic value is, sets that to the LED output. So when the button is not pressed, the LED is on (HIGH). When the button is pressed, the LED is off (LOW)

You can also see we enable the interrupt pin to listen to the button, you can use the INT pin to then monitor for 'changes' without performing an I2C read of the expander

## Constant current LED fade demo

This example shows how to set up a pin to use it as a constant-current driver for an LED. This is great for flicker-free LED driving, and consistent brightness no matter what the voltage is



In this example, connect an LED to the expander pin labeled 0

Connect the cathode (negative) leg of the LED to the 0 pad, then the anode (positive) to power

Note that you do not need a resistor with the LED because we are limited the current internally. However, there's no harm if the LED has a resistor.

Constant current mode is open drain sink ONLY. That means you must connect LED negative sides to the pin and the positive sides to power - this is opposite from how we normally drive LEDs from a GPIO pin

```
#include <Adafruit_AW9523.h>

Adafruit_AW9523 aw;

uint8_t LedPin = 0;  // 0 thru 15

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(1);  // wait for serial port to open

  Serial.println("Adafruit AW9523 Constant Current LED test!");

  if (! aw.begin(0x58)) {
    Serial.println("AW9523 not found? Check wiring!");
    while (1) delay(10);  // halt forever
  }

  Serial.println("AW9523 found!");
  aw.pinMode(LedPin, AW9523_LED_MODE); // set to constant current drive!
}


uint8_t x = 0;

void loop() {
  // Loop from 0 to 255 and then wrap around to 0 again
  aw.analogWrite(LedPin, x++);
  delay(10);
}
```

Open up File -> Examples -> Adafruit AW9523 -> constcurrent_demo and upload to your Arduino wired up to the breakout.

The `aw.analogWrite(LedPin, x++);` line will cause the LED on pin 0 to fade up from 0 (no current) to 255 (max current) Then once at maximum 255, the x variable will wrap around and start again from 0 so you'll see it 'pulse'

# Arduino Docs

Arduino Docs ()

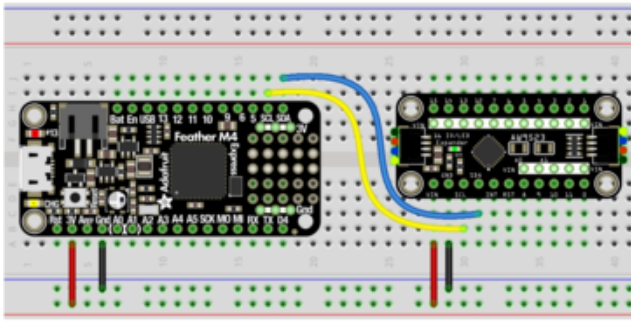
# Python & CircuitPython

It's easy to use the Adafruit AW9523 GPIO Expander and LED Driver Breakout with CircuitPython and the Adafruit CircuitPython AW9523 () module.  This module allows you to easily write Python code that reads temperature and humidity data from the sensor.

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 an AW9523 to your board exactly as follows. Here is an example of the AW9523 wired to a Feather using I2C:

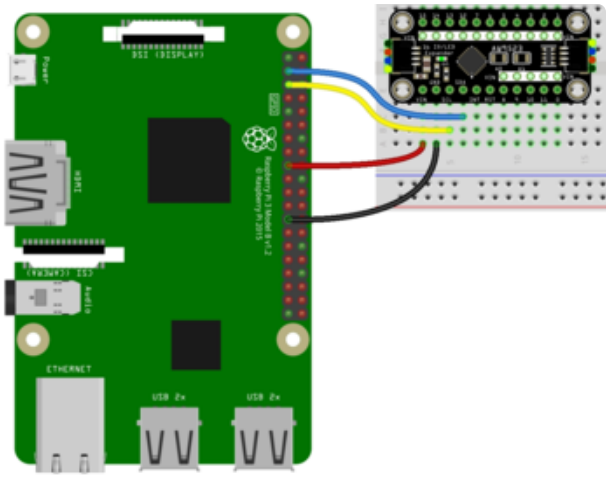


Board 3V to breakout VIN (red wire)
Board GND to breakout GND (black wire)
Board SCL to breakout SCL (yellow wire)
Board SDA to breakout SDA (blue wire)

## 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 with I2C:



Pi 3V to breakout VIN (red wire)
Pi GND to breakout GND (black wire)
Pi SCL to breakout SCL (yellow wire)
Pi SDA to breakout SDA (blue wire)

## CircuitPython Installation of AW9523 Library

You'll need to install the Adafruit CircuitPython AW9523 () library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython () 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 (). Our CircuitPython starter guide has a great page on how to install libraries from the library bundle ().

Copy the following file from the bundle to the lib folder on your CIRCUITPY drive:

- adafruit_aw9523.mpy

Also copy the following folder (not its individual contents, but the whole folder itself) into lib:

- adafruit_register

Before continuing make sure your board's lib folder or root filesystem has both the ad afruit_aw9523.mpy file and adafruit_register folder copied over.

## Python Installation of AW9523 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-aw9523`

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

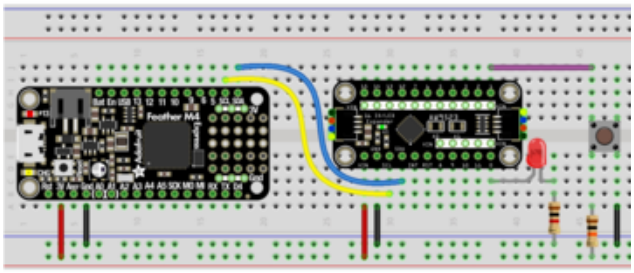
To demonstrate the usage of this breakout, you have two examples to choose from: I/O using a button and an LED, and constant current fade.

### Button and LED

This example shows how you can create an output and an input on the expander. Note that this expander does not support internal pull resistors.

Connect an LED to pin 0, with a resistor to ground, this is the output.

Connect a tactile switch with one side connected to pin 1 of the expander. Add a ~10K pull-up resistor on the expander pin 1 to Vin. Connect the other side of the button to ground. This makes it so the button pin is HIGH by default, and when you press the button, the pin goes LOW.



LED+ to breakout GPIO 0
LED- to 1K resistor
1K resistor to breadboard ground rail
One leg of button to breadboard ground rail
Other leg of button to breakout GPIO 1 +
10K pull-up resistor
10K pull-up resistor to breadboard VIN rail

Save the following as code.py on your CIRCUITPY drive.

```
# SPDX-FileCopyrightText: Copyright (c) 2020 ladyada for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import board
import digitalio
import adafruit_aw9523

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
aw = adafruit_aw9523.AW9523(i2c)

led_pin = aw.get_pin(0)  # LED on AW9523 io 0
button_pin = aw.get_pin(1)  # Button on AW io 1

# LED is an output, initialize to high
led_pin.switch_to_output(value=True)
# Button is an input, note pull-ups are not supported!
button_pin.direction = digitalio.Direction.INPUT

while True:
    # LED mirrors button pin
    led_pin.value = button_pin.value
```

Now, press the button. The LED lights up! The LED mirrors the button, so when the button is pressed, the LED lights up.

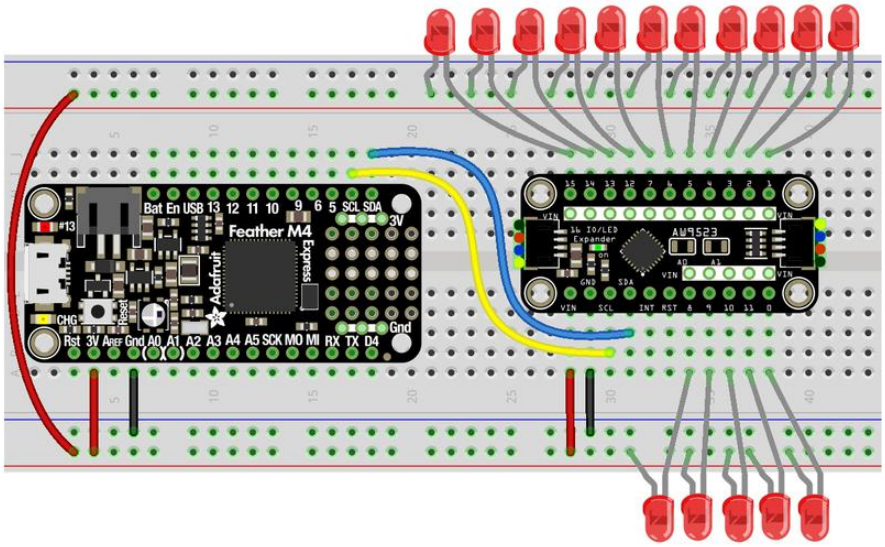
## Constant Current Fade

This example shows how to set up all the pins to use them as a constant-current driver for LEDs. This is great for flicker-free LED driving, and consistent brightness no matter what the voltage is.

In this example, connect LEDs to each expander pin.

Connect the cathode (negative) leg of the LED to the numbered GPIO pad, then the a node (positive) to power

Note that you do not need resistors with the LEDs because we are limiting the current internally. However, there's no harm if the LED has a resistor.




Constant current mode is open drain sink ONLY. That means you must connect LED negative sides to the pin and the positive sides to power - this is opposite from how we normally drive LEDs from a GPIO pin

Save the following as code.py on your CIRCUITPY drive.

```
# SPDX-FileCopyrightText: Copyright (c) 2020 ladyada for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import busio
import board
import adafruit_aw9523

i2c = busio.I2C(board.SCL, board.SDA)
aw = adafruit_aw9523.AW9523(i2c)
print("Found AW9523")

# Set all pins to outputs and LED (const current) mode
aw.LED_modes = 0xFFFF
aw.directions = 0xFFFF

n = 0
while True:
    for pin in range(16):
        # every LED is 'offset' by 16 counts so they dont all pulse together
        aw.set_constant_current(pin, (pin * 16 + n) % 256)
    # n increments to increase the current from 0 to 255, then wraps around
    n = (n + 1) % 256
```

All of the LEDs light up in a pattern.

The `aw.set_constant_current(pin, (pin * 16 + n) % 256)` plus `n = (n + 1) % 256` will cause the LEDs on each pin to fade up from 0 (no current) to 255 (max current), then fade back down. The fade on each pin is offset by 16 counts so they don't all pulse together.

That's all there is to using the AW9523 with CircuitPython and Python!

---

# Python Docs

Python Docs ()

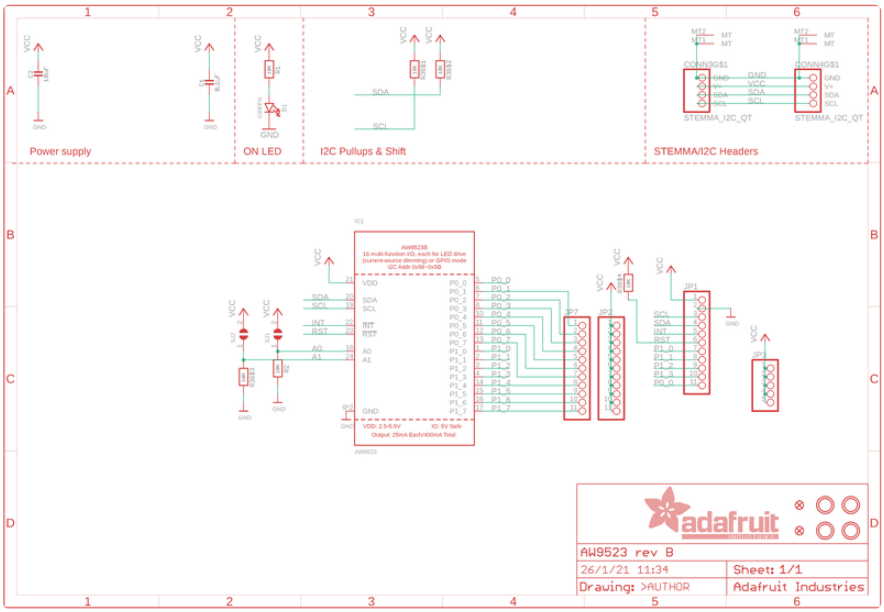
---

# Downloads

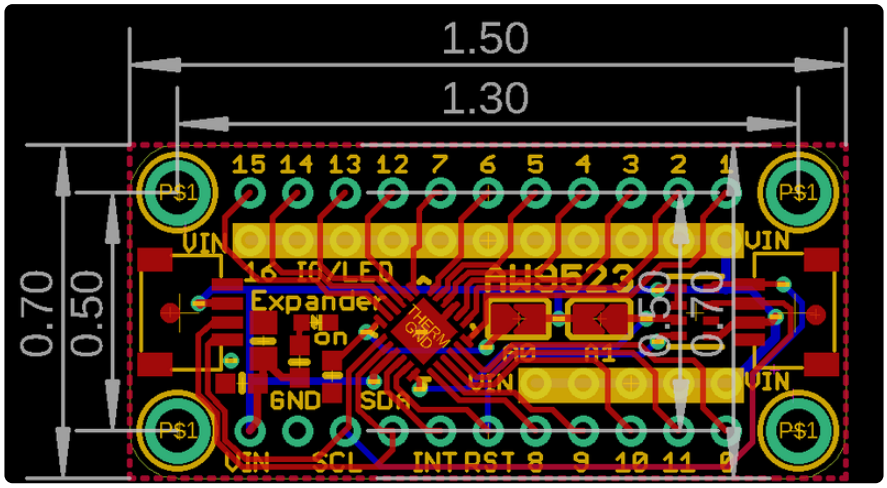
## Files:

- AW9523 Datasheet ()
- Fritzing object in the Adafruit Fritzing Library ()
- EagleCAD PCB files on GitHub ()

# Schematic



# Fab Print


1.50
1.30
0.70
0.50
15 14 13 12 7 6 5 4 3 2 1
P$1
VIN
16 IO/LED
Expander
on
GND
SDA
SCL
INT RST 8 9 10 11 0
AU9523
A0 A1
THERM GND
0.50
0.70