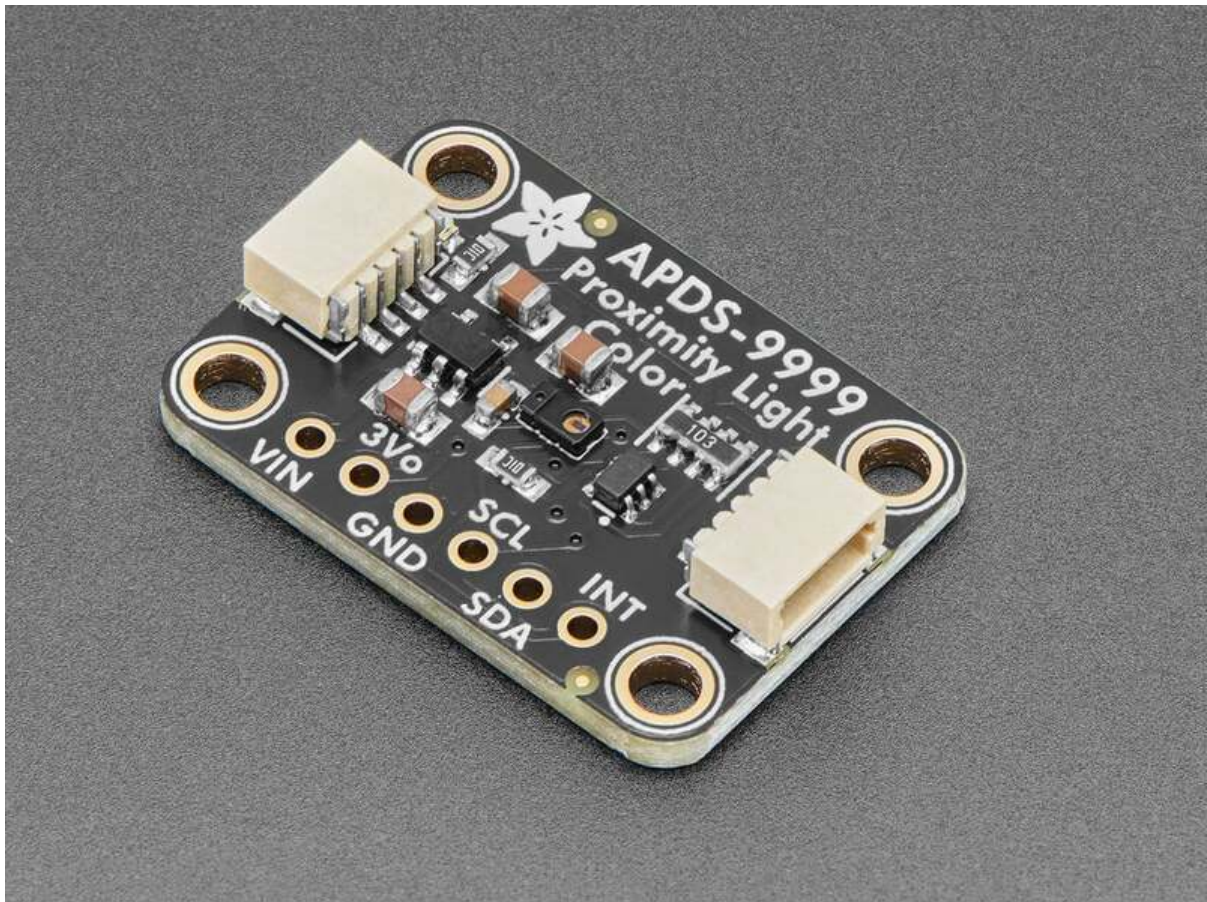




Adafruit APDS9999 Proximity, Lux Light & Color Sensor

Created by Tim C



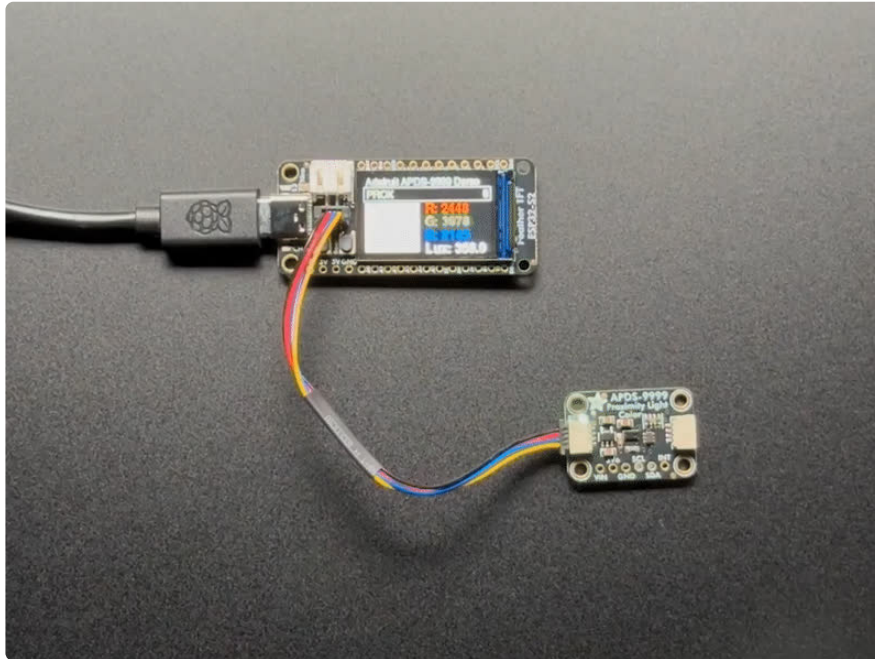
<https://learn.adafruit.com/adafruit-apds9999-proximity-lux-light-color-sensor>

Last updated on 2026-04-28 07:37:04 PM UTC

Table of Contents

Overview	3
Pinouts	5
<ul style="list-style-type: none">• Power Pins• I2C Data Pins• Interrupt Pin• LED Jumper	
CircuitPython & Python	8
<ul style="list-style-type: none">• CircuitPython Microcontroller Wiring• Python Computer Wiring• Python Installation of APDS9999 Library• CircuitPython Usage• Python Usage• Example Code	
Python Docs	13
Arduino	13
<ul style="list-style-type: none">• Wiring• Library Installation• Example Code	
Arduino Docs	17
WipperSnapper	17
<ul style="list-style-type: none">• What is WipperSnapper• Wiring• Usage	
Downloads	25
<ul style="list-style-type: none">• Files• Schematic and Fab Print	

Overview

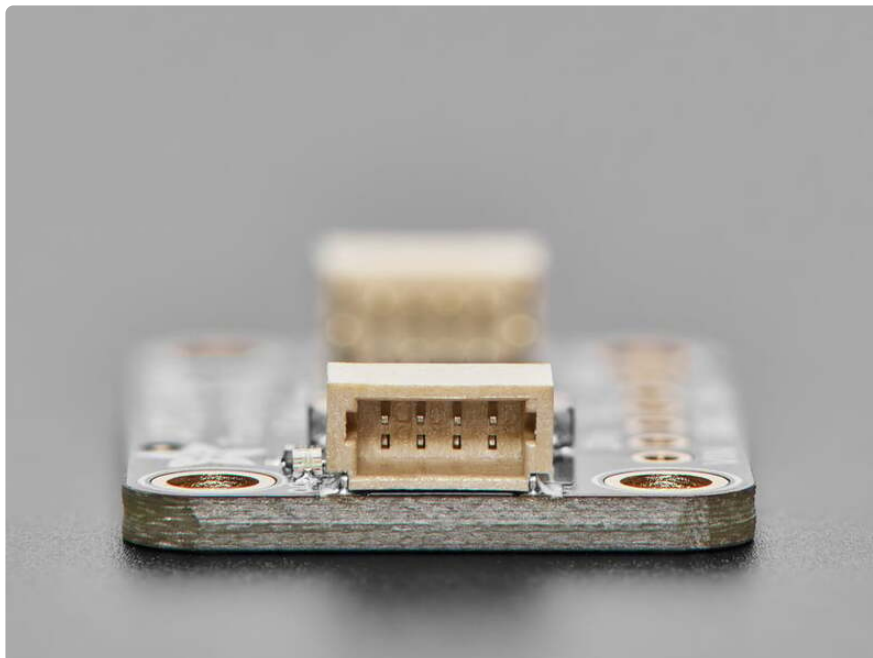


If you want a sensor that's a 3-in-1, the **Adafruit APDS-9999 Proximity, Lux Light, and Color Sensor** has RGB color, lux light level as well as infrared proximity all built into one device. When connected to your microcontroller (running our library code), it can detect the amount of red, blue, green, and clear light, as well as how close an object is to the front of the sensor. This device uses an I2C interface, so it's easy to wire up and use.

Note that this chip is considered by Broadcom to be the 'successor' [to the APDS-9960](http://adafru.it/3595) (<http://adafru.it/3595>) which has technically been discontinued, but it does not have gesture sensing! It does have true Lux sensing though, not just light counts.

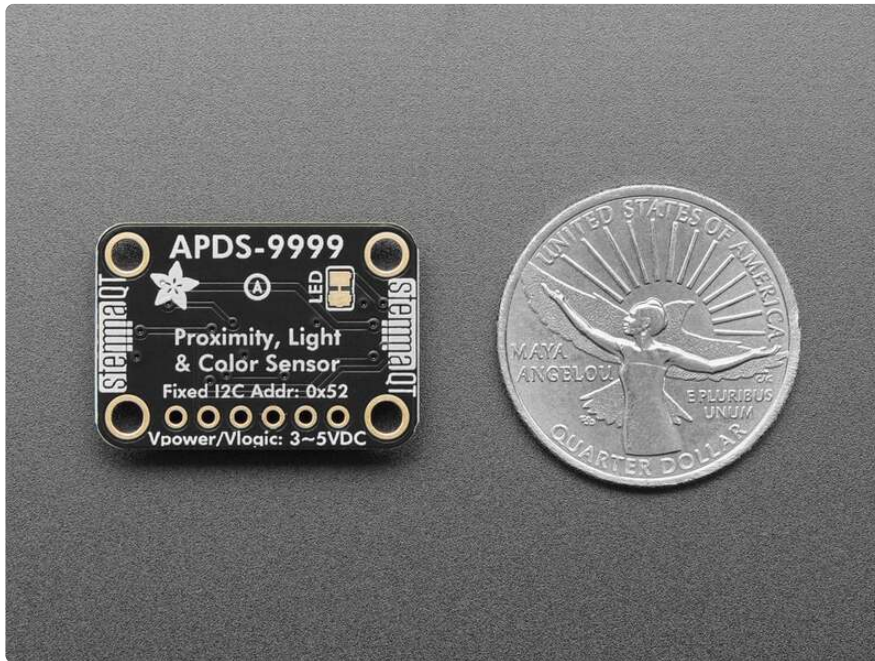


The proximity sensing bounces IR light, it isn't a time-of-flight sensor, so it doesn't give you exact distances, just how much light has reflected. It's best for short distances, 2 inches / 50mm or less, but can detect objects up to 150mm away. The APDS9999 also has a configurable interrupt pin that can trigger when a specified proximity threshold is crossed or when a color sensor threshold is exceeded.

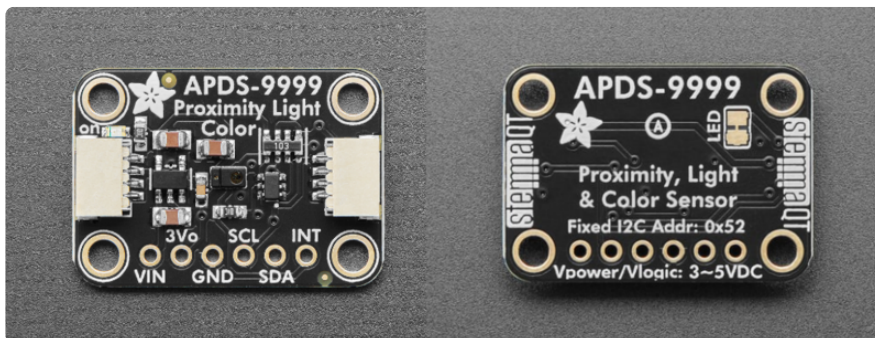


For your convenience, we've pick-and-placed the sensor on a PCB with a 3.3V regulator and some level shifting so it can be easily used with your favorite 3.3V or 5V microcontroller. We've included [SparkFun qwiic \(https://adafru.it/Fpw\)](https://adafru.it/Fpw)

compatible [STEMMA QT \(https://adafru.it/Ft4\)](https://adafru.it/Ft4) connectors for the I2C bus, so **you** don't even need to solder! [Just wire up to your favorite microcontroller or computer with a plug-and-play QT cable \(https://adafru.it/JRA\)](https://adafru.it/JRA) to light/color/proximity data ASAP. **QT Cable is not included**, but we have a variety in the shop (<https://adafru.it/17VE>).

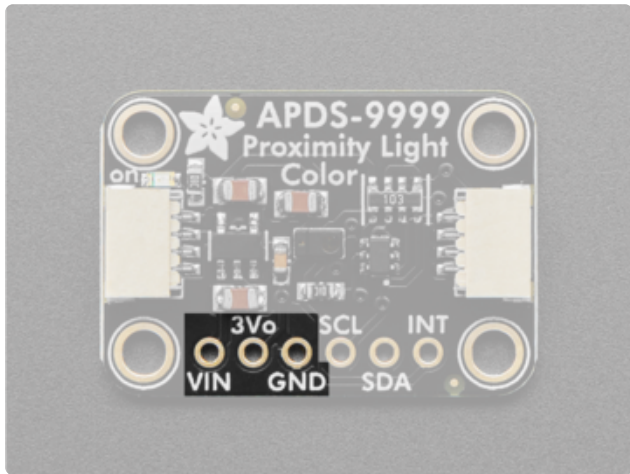


Pinouts



The fixed I2C address is **0x52**.

Power Pins

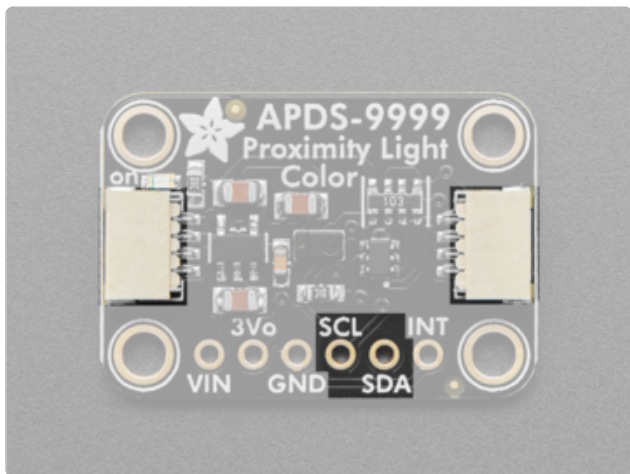


Vin - this is the power pin. Since the sensor chip uses 3VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it to 3VDC. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V.

3Vo - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like

GND - common ground for power and logic

I2C Data Pins

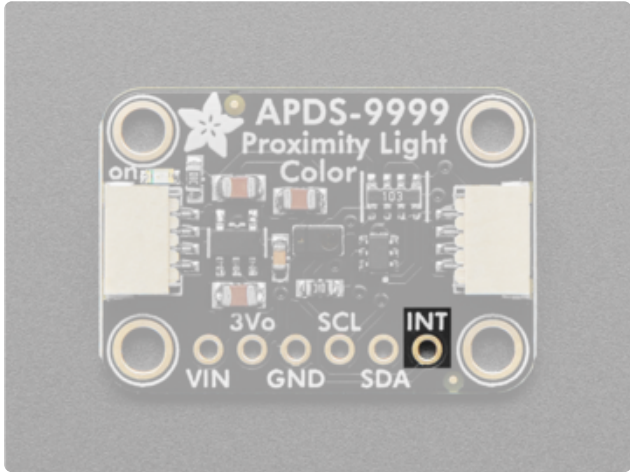


SCL - the I2C clock pin, connect to your microcontroller's I2C clock line.

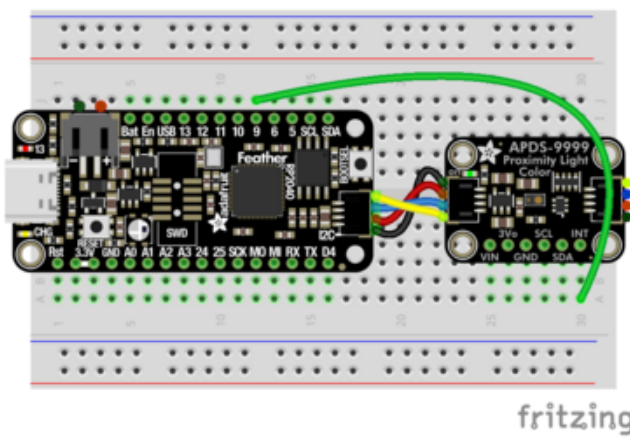
SDA - the I2C data pin, connect to your microcontroller's I2C data line.

STEMMA QT (<https://adafru.it/Ft4>) - These connectors allow you to connect to dev boards with **STEMMA QT** connectors or to other things with [various associated accessories](#) (<https://adafru.it/Ft6>)

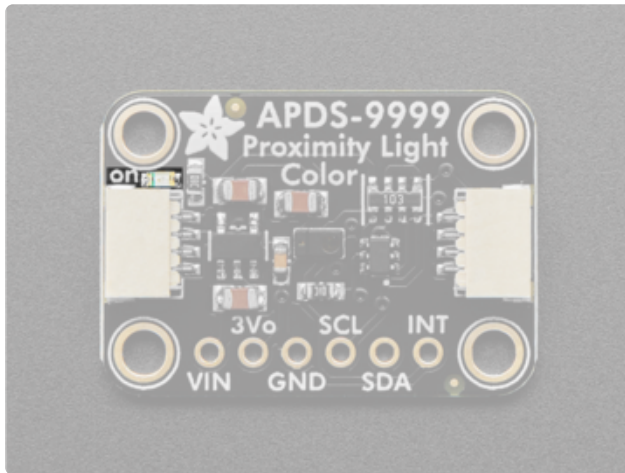
Interrupt Pin



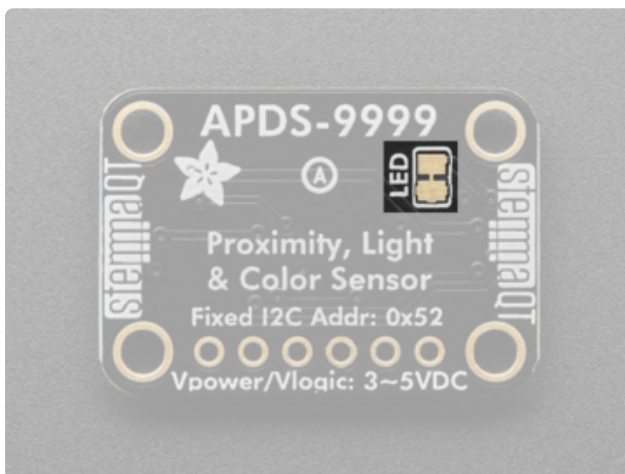
INT - the interrupt pin. It is an open-drain interrupt. To use, connect the pin to a GPIO pin on your microcontroller.



LED Jumper



Above the left STEMMA QT connector there is a small green **ON** power indicator LED. When the breakout is powered this LED will turn on.



On the back of the APDS9999 breakout there is a jumper with a small trace between two pads that can be cut to disable the **ON** power indicator LED on the board.

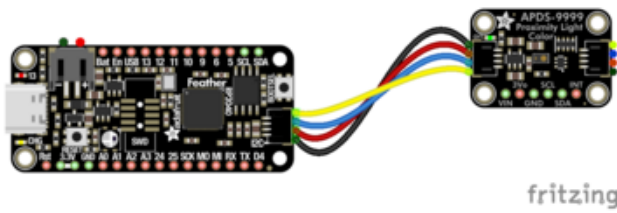
CircuitPython & Python

It's easy to use the **APDS9999** with Python or CircuitPython, and the [Adafruit_CircuitPython_APDS9999](https://adafru.it/1aAP) (<https://adafru.it/1aAP>) module. This module allows you to easily write Python code to read data from the sensor.

You can use this driver 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) (<https://adafru.it/BSN>).

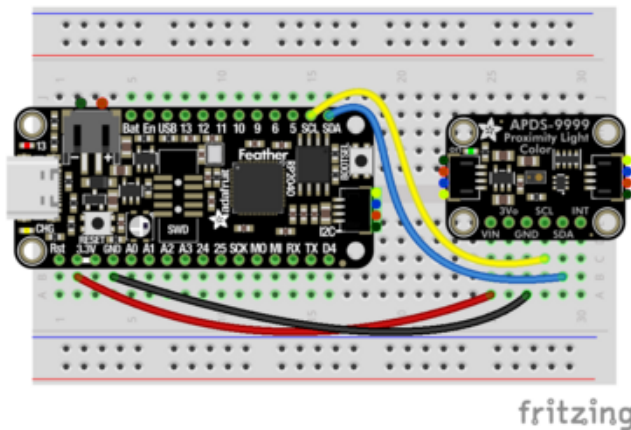
CircuitPython Microcontroller Wiring

First wire up a APDS9999 to your board exactly as shown below. Here's an example of wiring a Feather RP2040 to the sensor with I2C using one of the handy [STEMMA QT](https://adafru.it/Ft4) (<https://adafru.it/Ft4>) connectors:



- Board STEMMA 3V to sensor VIN (red wire)
- Board STEMMA GND to sensor GND (black wire)
- Board STEMMA SCL to sensor SCL (yellow wire)
- Board STEMMA SDA to sensor SDA (blue wire)

The following is the APDS9999 wired to a Feather RP2040 using a solderless breadboard:

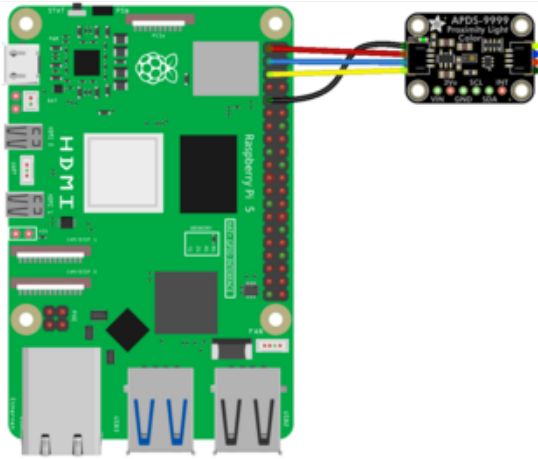


- Board 3V to sensor VIN (red wire)
- Board GND to sensor GND (black wire)
- Board SCL to sensor SCL (yellow wire)
- Board SDA to sensor SDA (blue wire)

Python Computer Wiring

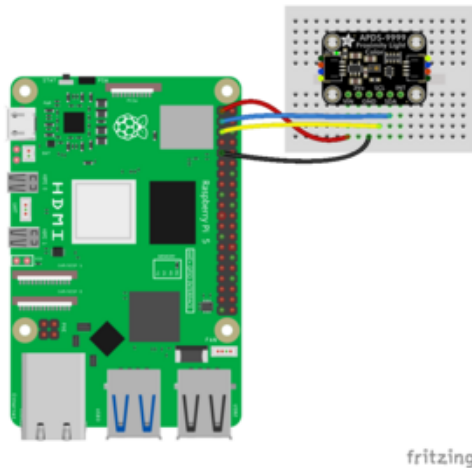
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported \(https://adafru.it/BSN\)](https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C using the STEMMA connector:



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

Here's the Raspberry Pi wired with I2C using a solderless breadboard:



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

Python Installation of APDS9999 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 \(https://adafru.it/BSN\)](https://adafru.it/BSN)!

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

- `pip3 install adafruit-circuitpython-apds9999`

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!

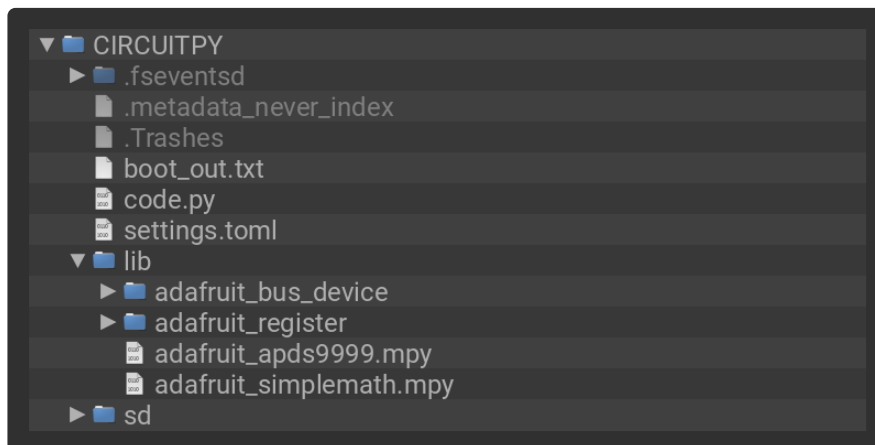
CircuitPython Usage

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_APDS9999** 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 **code.py** file to your **CIRCUITPY** drive.

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

- **adafruit_bus_device/**
- **adafruit_register/**
- **adafruit_apds9999.mpy**
- **adafruit_simplemath.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

If running **CircuitPython**: Once everything is saved to the **CIRCUITPY** drive, [connect to the serial console \(https://adafru.it/Bec\)](#) to see the data printed out!

If running **Python**: The console output will appear wherever you are running Python.

```
1 # SPDX-FileCopyrightText: Copyright (c) 2026 Tim Cocks for Adafruit Industries
2 #
3 # SPDX-License-Identifier: MIT
4 import time
5
6 import board
7
8 from adafruit_apds9999 import APDS9999
9
10 """
11 Demonstrate the setup and basic RGB/IR and proximity
12 sensing functionality of the APDS9999
13 """
14
15 apds_sensor = APDS9999(board.I2C())
16
17 apds_sensor.light_sensor_enabled = True
18 apds_sensor.proximity_sensor_enabled = True
19 apds_sensor.rgb_mode = True
20
21 while True:
22     time.sleep(1)
23     r, g, b, ir = apds_sensor.rgb_ir
24
25     print(
26         f"r: {r}, g: {g}, b: {b}, ir: {ir} "
27         f"lux: {apds_sensor.calculate_lux(g)} proximity: {apds_sensor.proximity}"
28     )
```

https://github.com/adafruit/Adafruit_CircuitPython_APDS9999/blob/main/examples/apds9999_simpletest.py

First, the sensor gets initialized over I2C, then configured to enable both light and proximity sensing. In the main loop the RGB/IR and proximity data are read from the sensor and printed to the serial console once per second.

```
code.py output:
r: 42, g: 109, b: 44, ir: 13 lux: 6.431 proximity: 23
r: 42, g: 109, b: 44, ir: 13 lux: 6.431 proximity: 22
r: 42, g: 108, b: 43, ir: 13 lux: 6.372 proximity: 22
r: 60, g: 137, b: 59, ir: 17 lux: 8.083 proximity: 22
r: 58, g: 134, b: 58, ir: 17 lux: 7.906 proximity: 22
r: 58, g: 134, b: 58, ir: 17 lux: 7.906 proximity: 22
r: 58, g: 134, b: 58, ir: 17 lux: 7.906 proximity: 22
r: 58, g: 134, b: 58, ir: 17 lux: 7.906 proximity: 22
r: 58, g: 134, b: 58, ir: 17 lux: 7.906 proximity: 22
r: 58, g: 134, b: 58, ir: 17 lux: 7.906 proximity: 22
```

Python Docs

[Python Docs \(https://adafru.it/1aAH\)](https://adafru.it/1aAH)

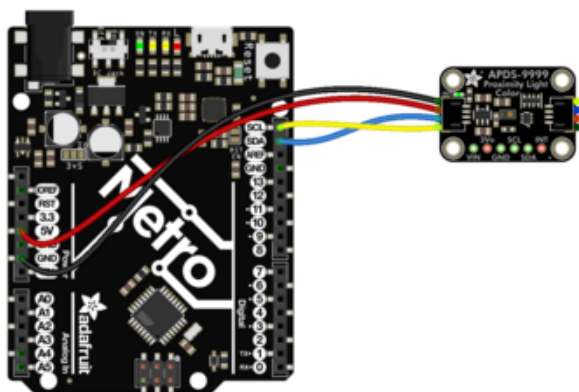
Arduino

Using the APDS9999 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller, installing the [Adafruit_APDS9999 \(https://adafru.it/1aAQ\)](https://adafru.it/1aAQ) 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 sensor **VIN**.

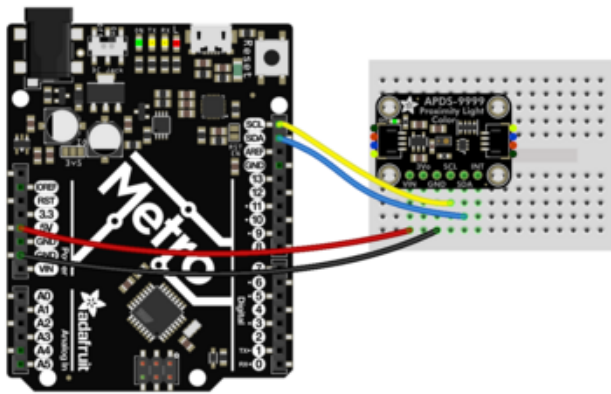
Here is an Adafruit Metro wired up to the sensor using the STEMMA QT connector:



fritzing

Board 5V 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)

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

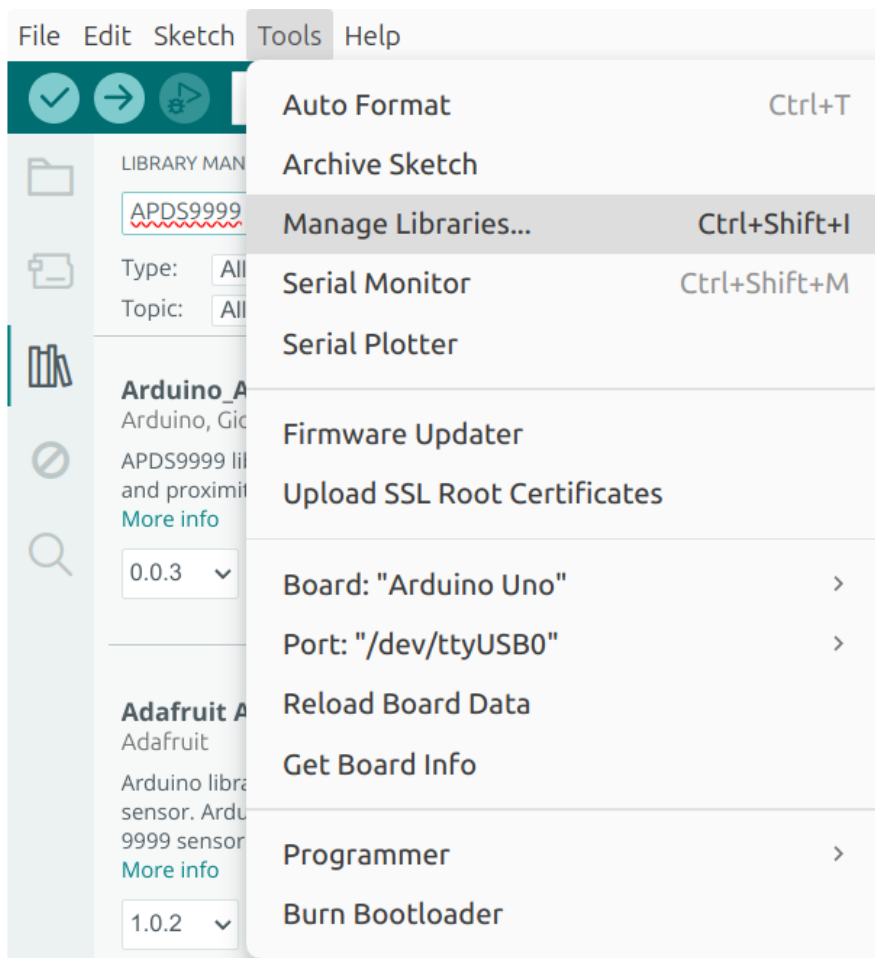


- Board 5V to breakout VIN (red wire)
- Board GND to breakout GND (black wire)
- Board SCL to breakout SCL (yellow wire)
- Board SDA to breakout SDA (blue wire)

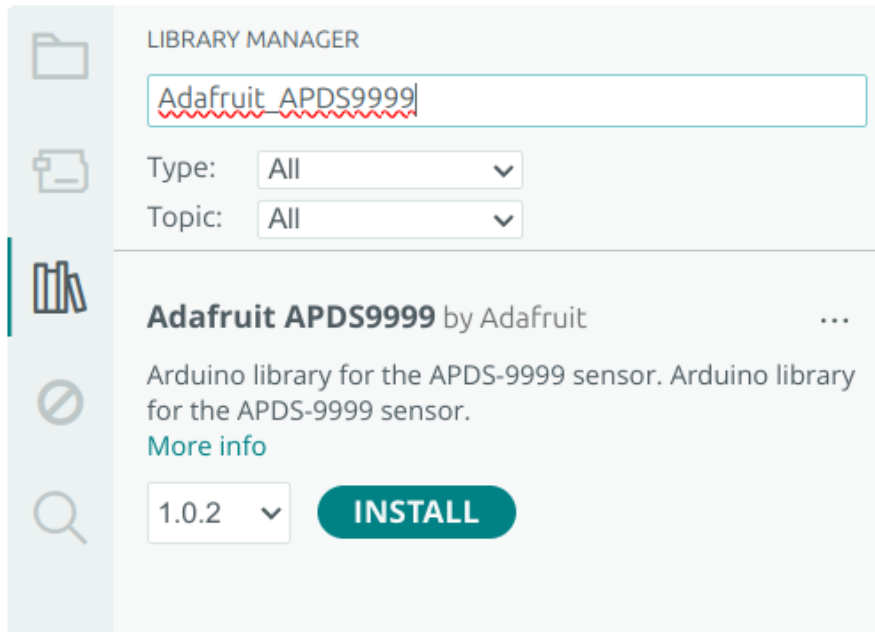
fritzing

Library Installation

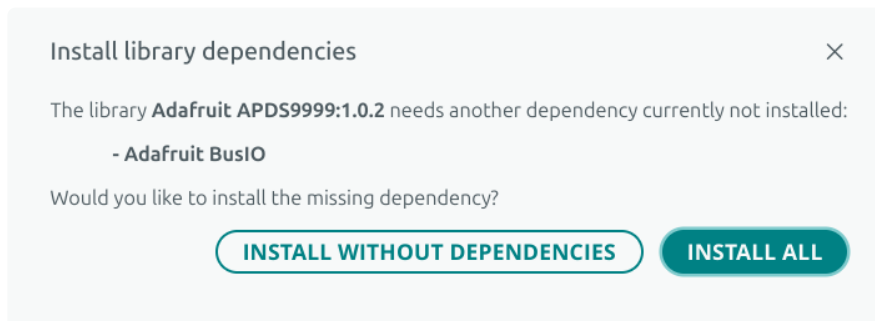
You can install the **Adafruit_APDS9999** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **Adafruit_APDS9999**, and select the **Adafruit_APDS9999** library:



If asked about dependencies, click "Install all".



If the "Dependencies" window does not come up, then you already have the dependencies installed.



If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Example Code

```
1  /*
2  * Simple color sensor example for APDS-9999
3  */
4
5  #include "Adafruit_APDS9999.h"
6
7  Adafruit_APDS9999 apds;
8
9  void setup() {
10   Serial.begin(115200);
11   while (!Serial)
12     delay(10);
13
14   if (!apds.begin()) {
15     Serial.println("Failed to find APDS-9999");
16     while (1)
17       delay(10);
18   }
19
20   apds.enableLightSensor(true);
21   apds.setRGBMode(true);
22   Serial.println("APDS-9999 Color Sensor");
23 }
24
25 void loop() {
26   uint32_t r, g, b, ir;
27
28   if (apds.getRGBIRData(&r, &g, &b, &ir)) {
29     Serial.print("R:");
30     Serial.print(r);
31     Serial.print(" G:");
32     Serial.print(g);
33     Serial.print(" B:");
34     Serial.print(b);
35     Serial.print(" IR:");
36     Serial.print(ir);
37     Serial.print(" Lux:");
38     Serial.println(apds.calculateLux(g));
39   }
40
41   delay(100);
42 }
```

https://github.com/adafruit/Adafruit_APDS9999/blob/main/examples/color_sensor/color_sensor.ino

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the APDS9999 recognized over I2C. Then, the RGB, IR, and calculated lux values will be printed out to the Serial Monitor.

```
Output Serial Monitor x
Message (Enter to send message to 'Arduino Uno' on '/dev/ttyUSB0')

APDS-9999 Color Sensor
R:0 G:0 B:0 IR:0 Lux:0.00
R:105 G:181 B:103 IR:26 Lux:32.58
R:104 G:179 B:102 IR:25 Lux:32.22
R:103 G:178 B:102 IR:25 Lux:32.04
R:104 G:179 B:102 IR:25 Lux:32.22
R:104 G:179 B:102 IR:26 Lux:32.22
R:104 G:179 B:102 IR:26 Lux:32.22
R:103 G:178 B:102 IR:27 Lux:32.04
R:104 G:179 B:102 IR:27 Lux:32.22
R:106 G:182 B:105 IR:25 Lux:32.76
R:106 G:176 B:100 IR:18 Lux:31.68
R:123 G:199 B:110 IR:18 Lux:35.82
R:154 G:255 B:149 IR:23 Lux:45.90
R:177 G:296 B:179 IR:27 Lux:53.28
R:185 G:311 B:191 IR:28 Lux:55.98
R:191 G:320 B:198 IR:29 Lux:57.60
R:191 G:322 B:201 IR:29 Lux:57.96
R:192 G:322 B:202 IR:30 Lux:57.96
```

Arduino Docs

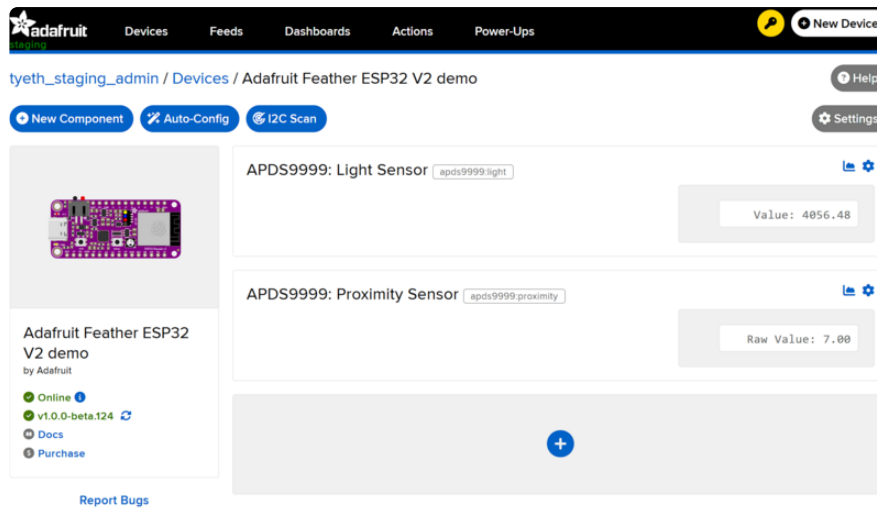
[Arduino Docs \(https://adafru.it/1aAI\)](https://adafru.it/1aAI)

WipperSnapper



WipperSnapper version 1 doesn't allow the reporting of RGB (Red/Green/Blue) colour information. This means **no RGB data will be recorded from this sensor in v1**, but it will still report light levels and proximity data.

Version 2 is in progress and we hope to have R/G/B data supported later this year.



What is WipperSnapper

WipperSnapper is a firmware designed to turn any WiFi-capable board into an Internet-of-Things device without programming a single line of code. WipperSnapper connects to [Adafruit IO \(https://adafru.it/fsU\)](https://adafru.it/fsU), a web platform designed ([by Adafruit! \(https://adafru.it/Bo5\)](https://adafru.it/Bo5)) to display, respond, and interact with your project's data.

Simply load the WipperSnapper firmware onto your board, add credentials, and plug it into power. Your board will automatically register itself with your Adafruit IO account.

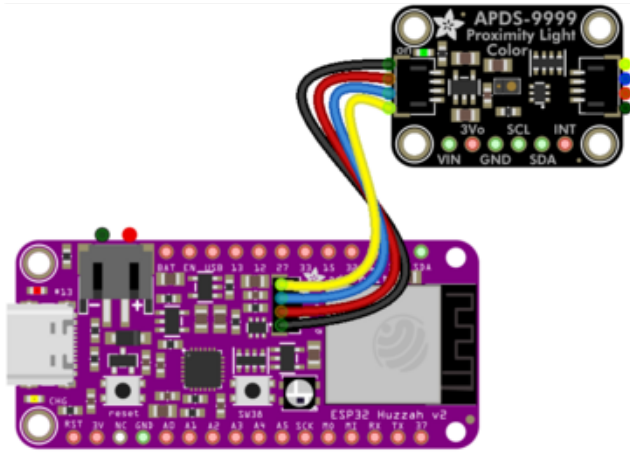
From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

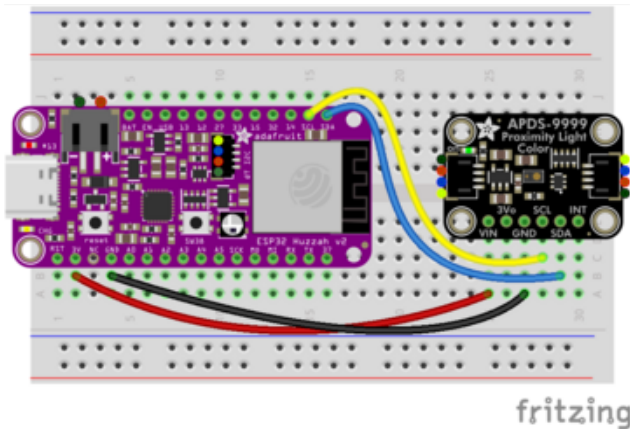
<https://adafru.it/Vfd>

Wiring

Wire up the APDS-9999 exactly as follows. Here it's shown connected via our convenient StemmaQT cable, or alternatively wired up on a prototyping breadboard:



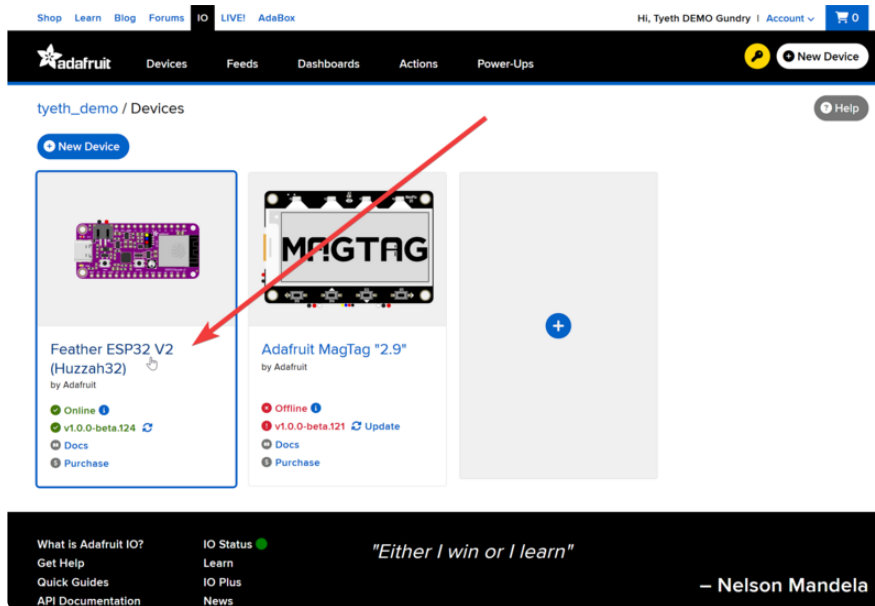
- Board 3V to sensor VIN (red wire)
- Board GND to sensor GND (black wire)
- Board SCL to sensor SCL (yellow wire)
- Board SDA to sensor SDA (blue wire)



Usage

Connect your board to Adafruit IO Wippersnapper and [navigate to the WipperSnapper board list \(https://adafru.it/TAu\)](https://adafru.it/TAu).

On this page, select the WipperSnapper board you're using to be brought to the board's interface page.



If you do not see your board listed here - you need [to connect your board to Adafruit IO](https://adafru.it/Vfd) (<https://adafru.it/Vfd>) first.

Feather ESP32 V2 (Huzzah32)

by Adafruit

- ✓ Online ⓘ
- ✓ v1.0.0-beta.124 ↻
- 📖 Docs
- 💰 Purchase

Adafruit MagTag "2.9"

by Adafruit

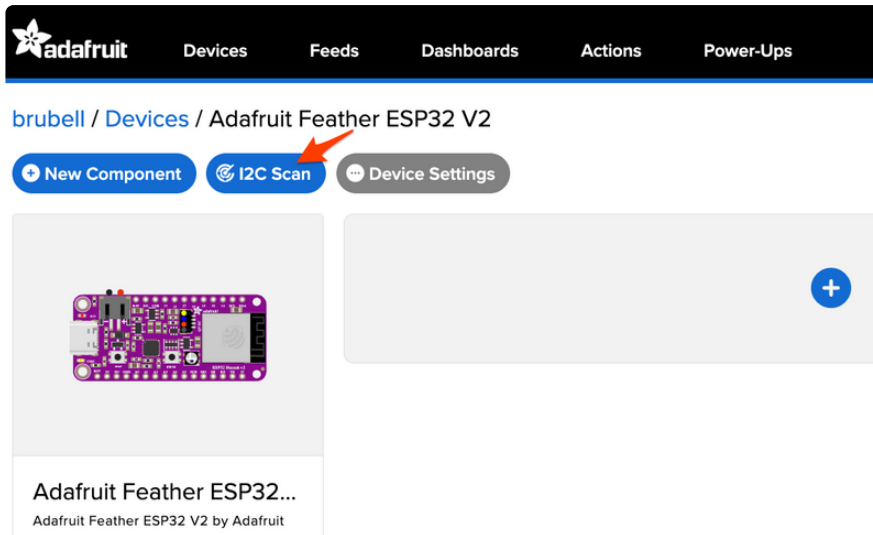
- ✓ Online ⓘ
- ! v1.0.0-beta.122 ↻ Update
- 📖 Docs
- 💰 Purchase

On the device page, quickly **check** that you're running the latest version of the WipperSnapper firmware.

The device tile on the left indicates the version number of the firmware running on the connected board.

If the firmware version is green with a checkmark - continue with this guide. If the firmware version is red with an exclamation mark "!" - [update to the latest WipperSnapper firmware](https://adafru.it/Vfd) (<https://adafru.it/Vfd>) on your board before continuing.

Next, make sure the sensor is plugged into your board and click the **I2C Scan** button.

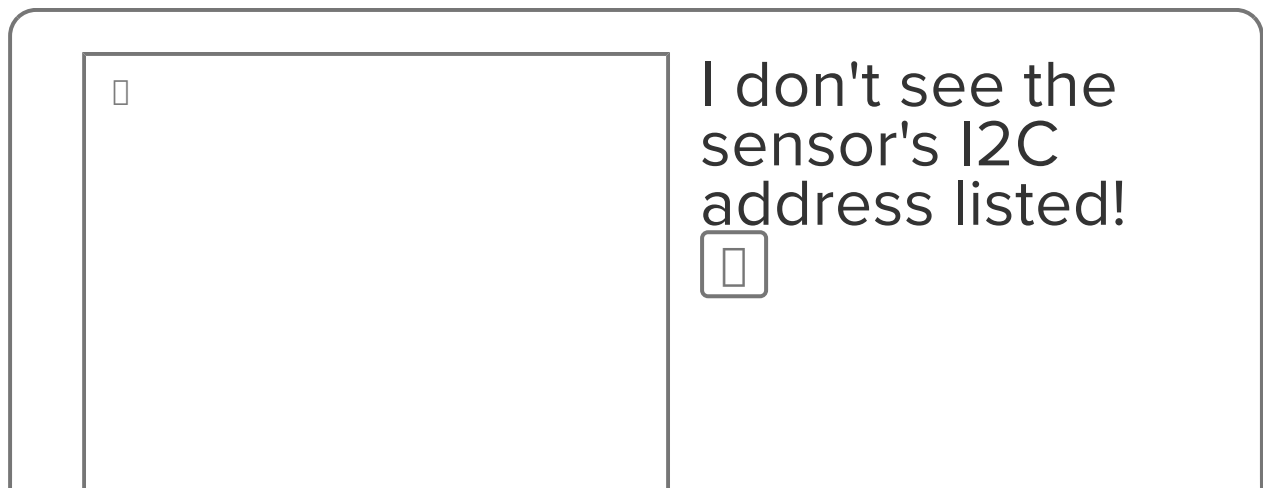


You should see the APDS-9999's default I2C address of `0x52` pop-up in the I2C scan list.

I2C Scan Complete ✕

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00								--	--	--	--	--	--	--	--	--
10	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50	--	--	52	--	--	--	--	--	--	--	--	--	--	--	--	--
60	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Close Scan Again



First, double-check the connection and/or wiring between the sensor and the board.

Then, reset the board and let it re-connect to Adafruit IO WipperSnapper.

With the sensor detected in an I2C scan, you're ready to add the sensor to your board.

Click the **New Component** button or the **+** button to bring up the component picker.



Adafruit IO supports a large amount of components. To quickly find your sensor, type **APDS-9999** into the search bar, then select the **APDS-9999** component.

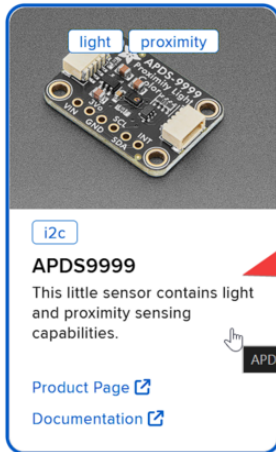
New Component



Which component would you like to set up?

APDS-9999

Displaying 1 matching Components.



The component card for the APDS-9999 sensor. At the top is an image of the sensor with labels 'light' and 'proximity'. Below the image is a blue 'i2c' tag. The title is 'APDS9999' followed by the description: 'This little sensor contains light and proximity sensing capabilities.' At the bottom are two links: 'Product Page' and 'Documentation', both with external link icons. A red arrow points from the search bar to the component card, and another red arrow points from the top right to the component card.

Cancel

On the component configuration page, the APDS-9999's sensor address should be listed along with the sensor's settings.

The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the APDS-9999 sensor and send the data to Adafruit IO. Measurements can range from every second to every 24 hours.

For this example, set the **Send Every** interval to every 30 seconds.

Create APDS9999 Component



Select I2C Address

0x52

Enable APDS9999: Light Sensor?

Name:

APDS9999: Light Sensor

Send Data:

Every 30 seconds

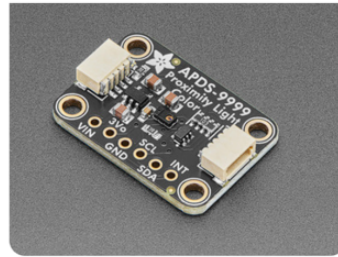
Enable APDS9999: Proximity Sensor?

Name:

APDS9999: Proximity Sensor

Send Data:

Every 30 seconds



[← Back to Component Type](#)

Create Component

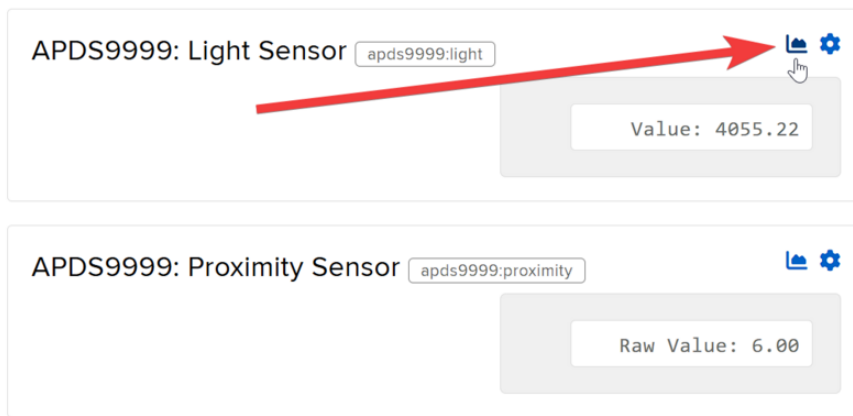
Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.

The screenshot shows the Adafruit IO dashboard for a device named 'Adafruit Feather ESP32 V2 demo'. The dashboard displays two configured sensors:

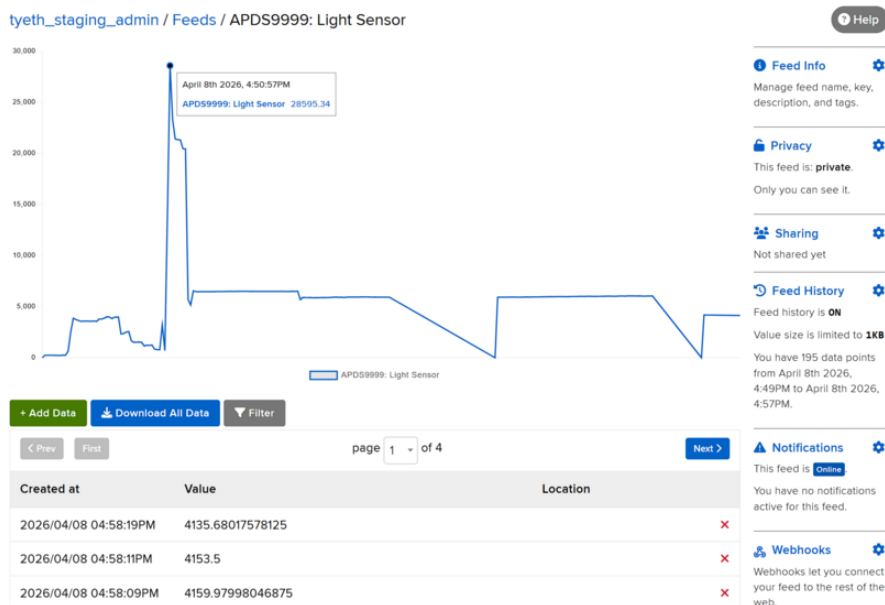
- APDS9999: Light Sensor** (ID: apds9999.light) with a current value of 4856.48.
- APDS9999: Proximity Sensor** (ID: apds9999.proximity) with a raw value of 7.00.

The dashboard also includes a 'New Component' button, 'Auto-Config', 'I2C Scan', and 'Settings' options. A footer section contains navigation links, IO status, and a quote by H. G. Wells: "Human history becomes more and more a race between education and catastrophe".

To view the data that has been logged from the sensor, click on the graph next to the sensor name.



Here you can see the feed history and edit things about the feed such as the name, privacy, webhooks associated with the feed and more. If you want to learn more about how feeds work, [check out this page \(https://adafru.it/10aZ\)](https://adafru.it/10aZ).



Downloads

Files

- [APDS9999 Datasheet \(https://adafru.it/1aAR\)](https://adafru.it/1aAR)
- [EagleCAD PCB files on GitHub \(https://adafru.it/1aAS\)](https://adafru.it/1aAS)
- [Fritzing object in the Adafruit Fritzing Library \(https://adafru.it/1aAT\)](https://adafru.it/1aAT)

Schematic and Fab Print

