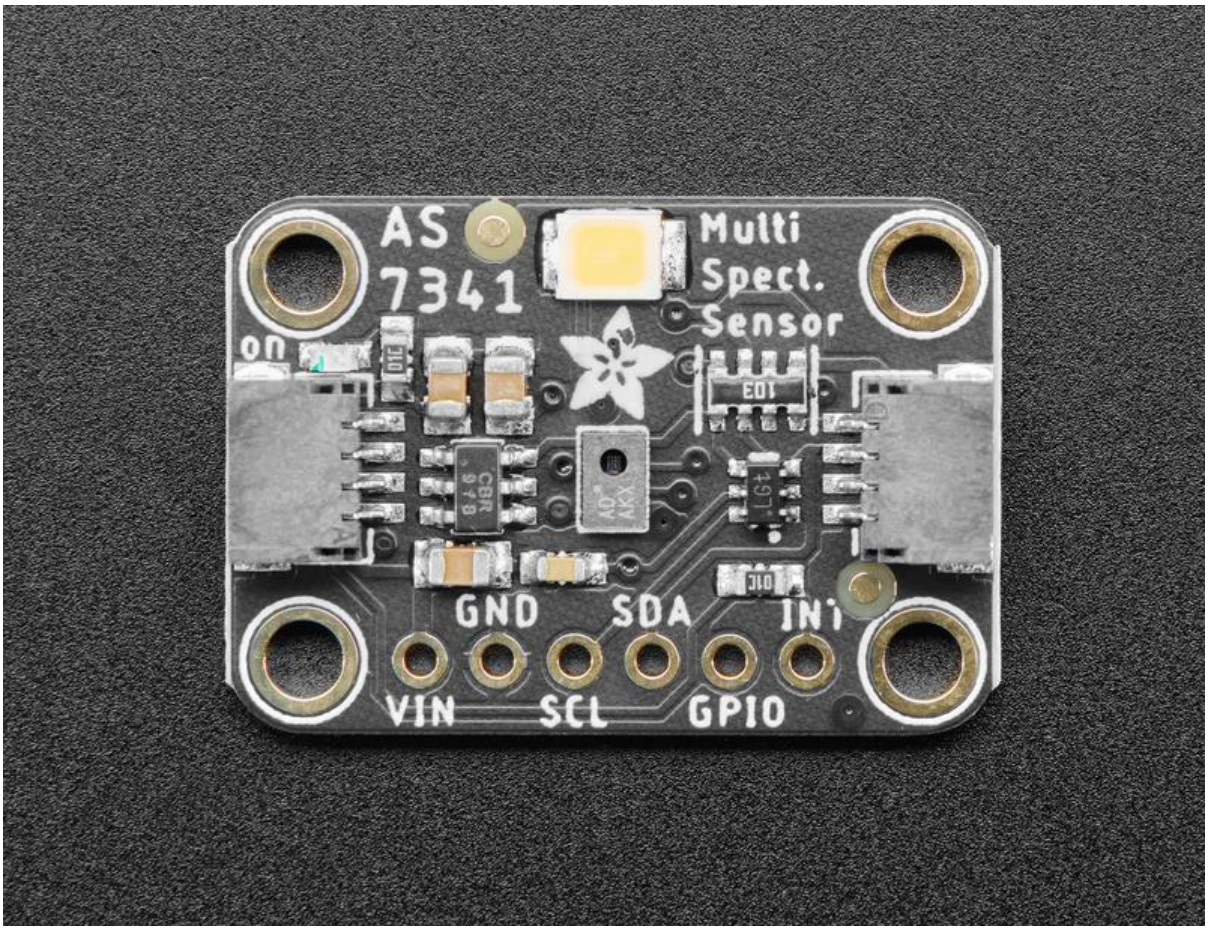
Adafruit AS7341 10-Channel Light / Color Sensor Breakout

Created by Bryan Siepert

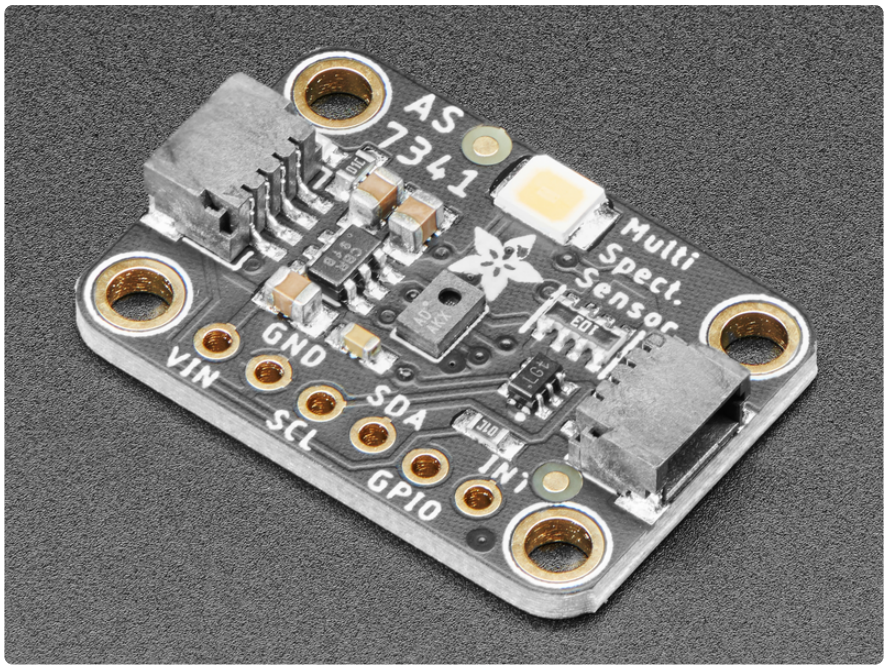


https://learn.adafruit.com/adafruit-as7341-10-channel-light-color-sensor-breakout

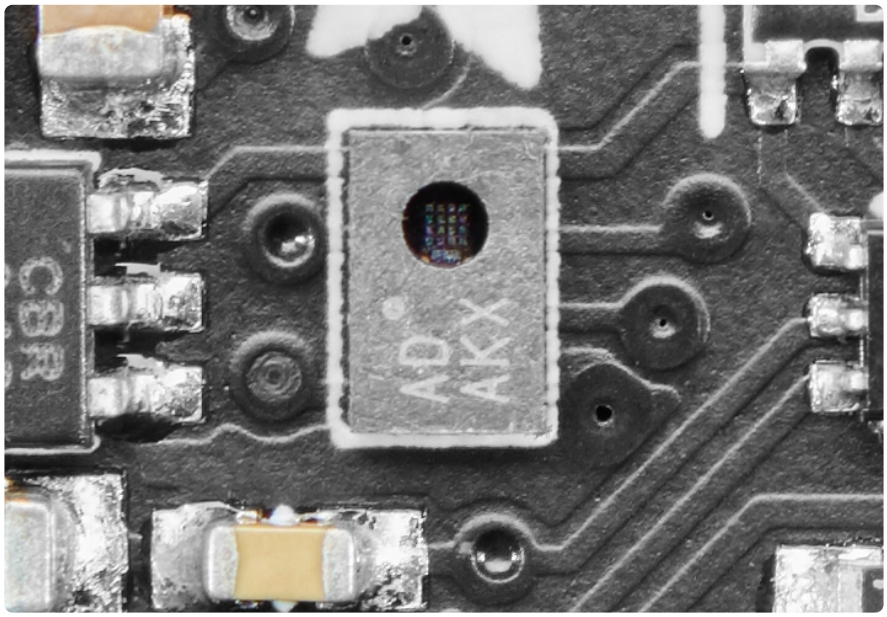
Last updated on 2023-08-29 04:29:56 PM EDT

Table of Contents

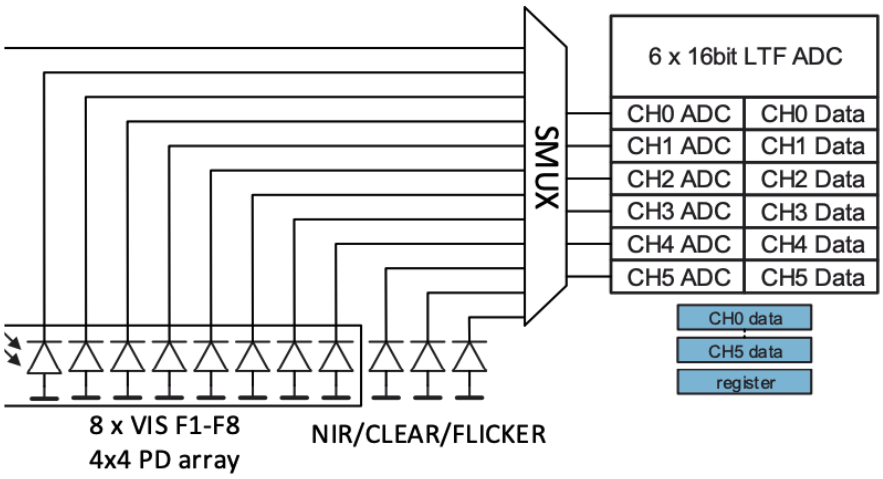
Overview



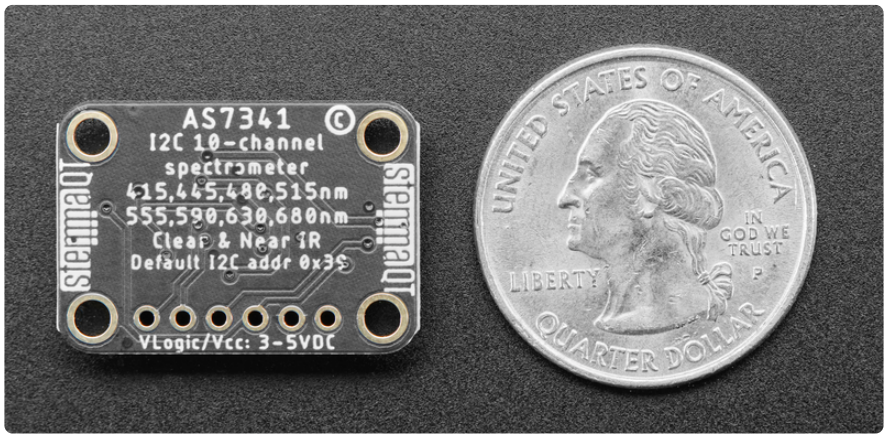
The AS7341 by AMS () is a multi-channel spectrometer which is a special type of light sensor that is able to detect not only the amount of light present, but also the amounts of light within different wavelengths. This means that you can use it to detect, much better than the human eye is capable of, what color or colors of light is present. The AS7341 packs within its 3x2mm footprint 16 different sensors that can detect 8 separate, overlapping bands of colored light. As if that weren't enough, it also includes sensors for white light as well as Near Infra-red light, and even sensors made specifically for detecting light flicker at specific frequencies from things like indoor lighting.



The super-human color measuring capabilities of the AS7341 can be used to quantify the specific makeup of whatever interesting colors you can point it at, and can do so with more accuracy and specificity than a well-trained artist. Now you don't need to go to art school to tell Smalt () from Ultramarine (), instead, you can have the AS7341 can give you a clue how blue your blue is. This is possible thanks to the impressive collection of sensors in the AS7341 being routed through a 16-bit 6-channel ADC that takes the raw measurements and converts them to digital values that can be read out over I2C.

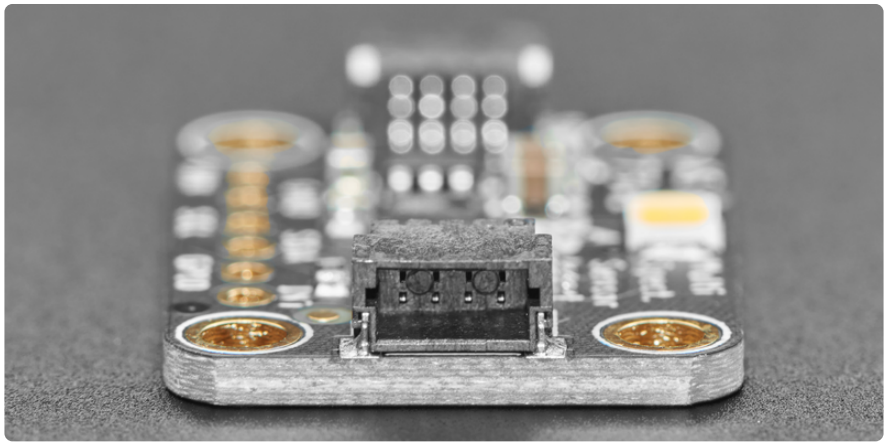


11 readable individual sensor elements (10 light channels plus flicker detection) don't exactly fit through a 6-channel ADC all at once, so the chip includes a so-called Super MUX (SMUX) that allows you to route the signal from any sensor to any ADC channel. Now that's some super multiplexing! The sensor also has GPIO and interrupt pins that can allow it to communicate directly with other sensors, or the microcontroller itself.

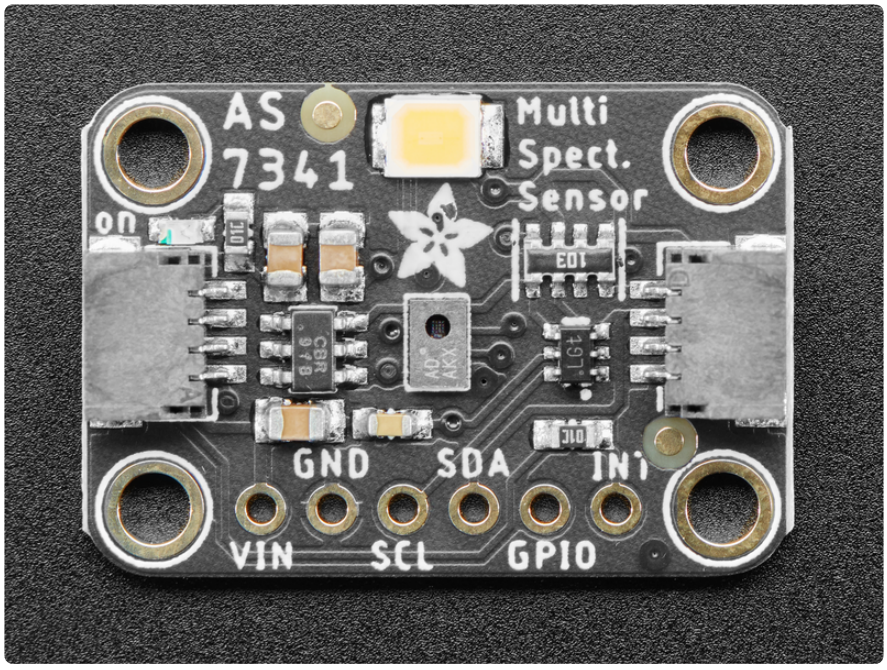


All of this capability is made accessible by mounting the sensor on a Stemma QT form factor breakout board, complete with level shifting circuitry and SparkFun Qwiic () co mpatible Stemma QT () connectors. This means that you can, without needing to solder, connect our AS7341 breakout into your 3.3V or 5V microcontroller of choice be it an Arduino Uno, Raspberry Pi, or one of the many CircuitPython compatible boards (

). While it certainly takes a bit of work to make all those different light sensors share their measurements, our Arduino and CircuitPython libraries take care of all of that hard work for you and even include example code to help get you started. Read on and you'll find library installation instructions, as well as wiring diagrams that make using the AS7341 super easy.



Pinouts



Power Pins

- VIN - this is the power pin. Since the sensor chip uses 1.8 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller 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. 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 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 () - These connectors allow you to connectors to dev boards with S TEMMA QT connectors or to other things with various associated accessories ()

Other Pins

- INT-This is the interrupt pin. You can setup the AS7341 to pull this low when certain conditions are met such as new measurement data being available. Consult the datasheet () for usage
- GPIO - This is a General Purpose Input Output pin that can be controlled via the AS7341 and can be used to trigger measurements. Consult the datasheet () for usage

---

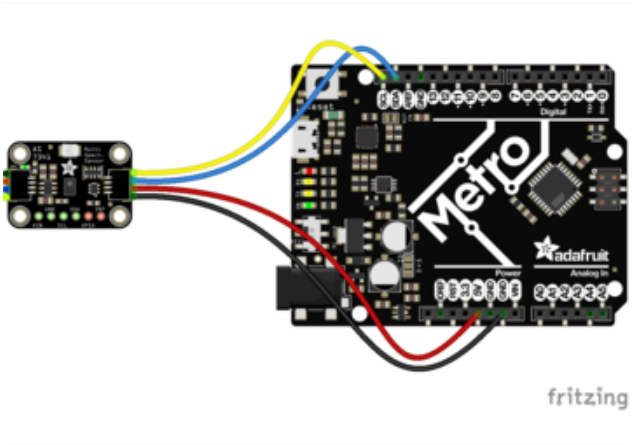
Arduino

Using the AS7341 with Arduino is a simple matter of wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit AS7341 () library we've written, and running the provided example code.

I2C Wiring

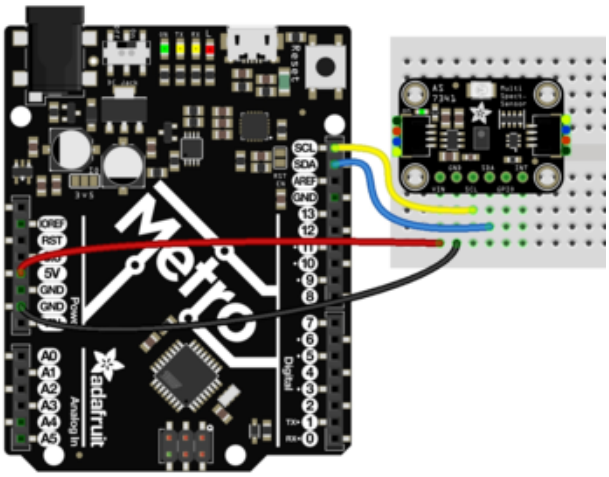
Use this wiring if you want to connect via I2C interface. The I2C address for the AS7341 is 0x39

Here is how to wire up the sensor 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 board VIN (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 board GND (black wire) to Arduino GND
Connect board SCL (yellow wire) to Arduino SCL
Connect board SDA (blue wire) to Arduino SDA

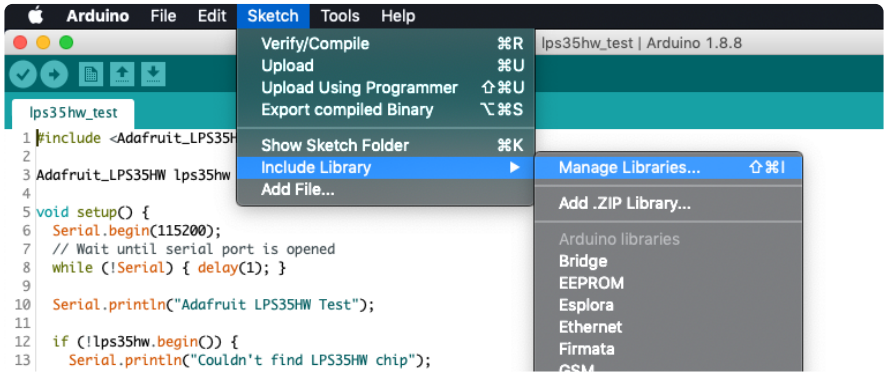
Here is how to wire the sensor to a board using a solderless breadboard:



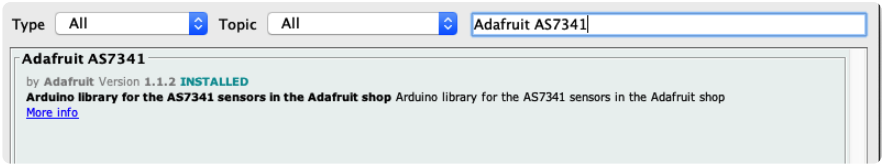
Connect board VIN (red wire) to Arduino 5V if you are running a 5V board Arduino (Uno, etc.). If your board is 3V, connect to that instead.
Connect board GND (black wire) to Arduino GND
Connect board SCL (yellow wire) to Arduino SCL
Connect board SDA (blue wire) to Arduino SDA

Library Installation

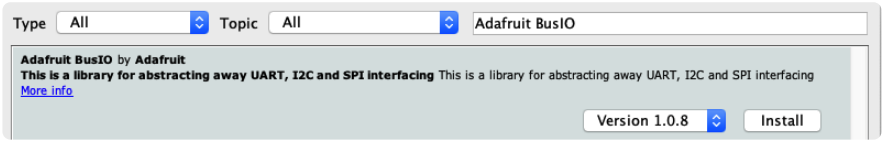
You can install the Adafruit AS7341 library for Arduino using the Library Manager in the Arduino IDE.



Click the Manage Libraries ... menu item, search for Adafruit AS7341, and select the A dafruit AS7341 library:



Follow the same process for the Adafruit BusIO library.



Load Example

Open up File -> Examples -> Adafruit AS7341 -> get_channel

After opening the demo file, upload to your Arduino wired up to the sensor. Once you upload the code, you will see the raw readings for each light frequency band being printed when you open the Serial Monitor (Tools->Serial Monitor) at 115200 baud, similar to this:

```
F1 415nm : 1022
F2 445nm : 1420
F3 480nm : 3179
F4 515nm : 2911
F5 555nm : 7546
F6 590nm : 4810
F7 630nm : 5728
F8 680nm : 4036
Clear    : 8113
Near IR  : 5151

F1 415nm : 1021
F2 445nm : 1419
F3 480nm : 3179
F4 515nm : 2909
F5 555nm : 7551
F6 590nm : 4818
F7 630nm : 5721
F8 680nm : 4030
Clear    : 8102
Near IR  : 5145
```

Example Code

```
/* This example will read all channels from the AS7341 and print out reported
values */

#include <Adafruit_AS7341.h>

Adafruit_AS7341 as7341;


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

  // Wait for communication with the host computer serial monitor
  while (!Serial) {
    delay(1);
  }
```

```
  if (!as7341.begin()){
    Serial.println("Could not find AS7341");
    while (1) { delay(10); }
  }

  as7341.setATIME(100);
  as7341.setASTEP(999);
  as7341.setGain(AS7341_GAIN_256X);
}

void loop() {
  // Read all channels at the same time and store in as7341 object
  if (!as7341.readAllChannels()){
    Serial.println("Error reading all channels!");
    return;
  }

  // Print out the stored values for each channel
  Serial.print("F1 415nm : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_415nm_F1));
  Serial.print("F2 445nm : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_445nm_F2));
  Serial.print("F3 480nm : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_480nm_F3));
  Serial.print("F4 515nm : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_515nm_F4));
  Serial.print("F5 555nm : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_555nm_F5));
  Serial.print("F6 590nm : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_590nm_F6));
  Serial.print("F7 630nm : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_630nm_F7));
  Serial.print("F8 680nm : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_680nm_F8));

  Serial.print("Clear    : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_CLEAR));

  Serial.print("Near IR  : ");
  Serial.println(as7341.getChannel(AS7341_CHANNEL_NIR));

  Serial.println("");
}
```

Arduino Docs

Arduino Docs ()

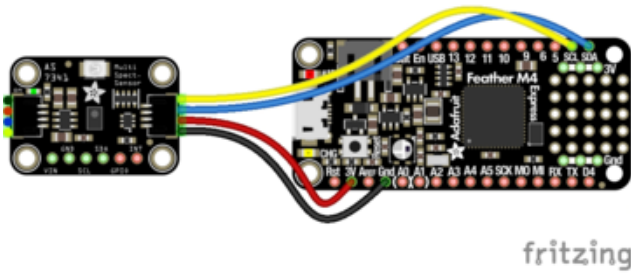
Python & CircuitPython

It's easy to use the AS7341 with Python or CircuitPython, and the Adafruit CircuitPython AS7341 () module. This module allows you to easily write Python code that reads multi-spectral color intensity from the AS7341 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 ().

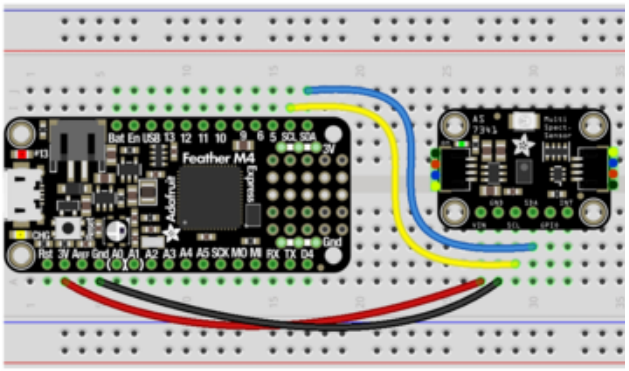
CircuitPython Microcontroller Wiring

First wire up a AS7341 to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using one of the handy STEMMA QT () connectors:



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)

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

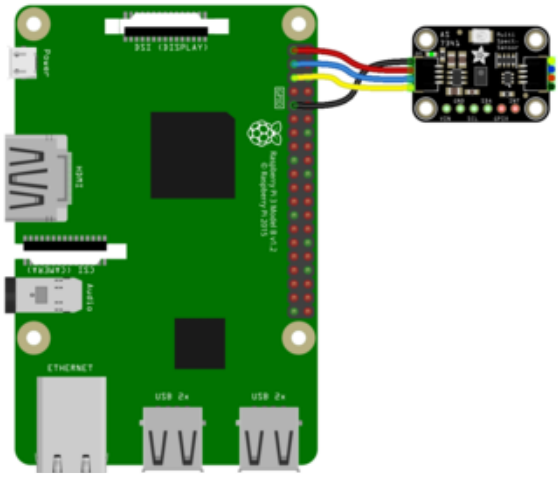


Board 3V to sensor VIN (red wire)
Board GND to sensor GND (black wire)
Board SCL to sensor SCL (yellow wire)
Board SDA to sensor SDA (blue wire)

Python Computer Wiring

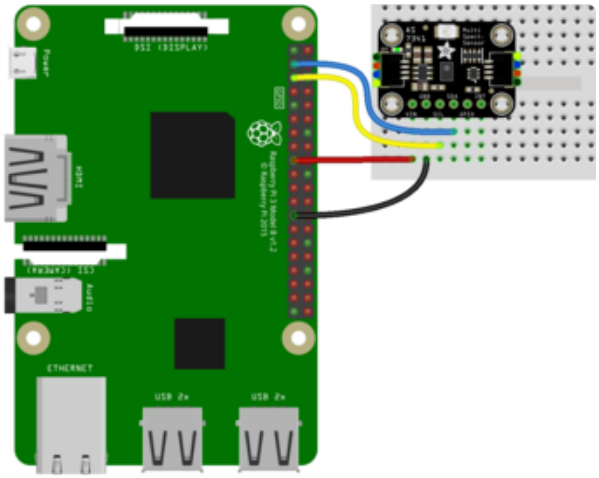
Since there's dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

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



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

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



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

CircuitPython Installation of AS7341 Library

You'll need to install the Adafruit CircuitPython AS7341 () 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 the library bundle ().

Before continuing make sure your board's lib folder or root filesystem has the adafruit_AS7341.mpy file as well as the adafruit_bus_device and adafruit_register folders.

Next connect to the board's serial REPL ()so you are at the CircuitPython >>> prompt.

Python Installation of AS7341 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:

- `sudo pip3 install adafruit-circuitpython-as7341`

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 the sensor we'll initialize it and read the measurements for each channel from the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
from time import sleep
import board
from adafruit_as7341 import AS7341

i2c = board.I2C()
sensor = AS7341(i2c)
```

```
>>> from time import sleep
>>> import board
>>> from adafruit_as7341 import AS7341
>>> i2c = board.I2C()
>>> sensor = AS7341(i2c)
```

Now you're ready to read values from the sensor using these properties using the code below

- channel_415nm - Data for F1, the 415nm channel
- channel_445nm - Data for F2, the 445nm channel
- channel_480nm - Data for F3, the 480nm channel
- channel_515nm - Data for F4, the 515nm channel

- channel_555nm - Data for F5, the 555nm channel
- channel_590nm - Data for F6, the 590nm channel
- channel_630nm - Data for F7, the 630nm channel
- channel_680nm - Data for F8, the 680nm channel

```
print("F1 - 415nm/Violet  %s" % sensor.channel_415nm)
print("F2 - 445nm//Indigo %s" % sensor.channel_445nm)
print("F3 - 480nm//Blue   %s" % sensor.channel_480nm)
print("F4 - 515nm//Cyan   %s" % sensor.channel_515nm)
print("F5 - 555nm/Green   %s" % sensor.channel_555nm)
print("F6 - 590nm/Yellow  %s" % sensor.channel_590nm)
print("F7 - 630nm/Orange  %s" % sensor.channel_630nm)
print("F8 - 680nm/Red     %s" % sensor.channel_680nm)
```

```
>>> print("F1 - 415nm/Violet  %s" % sensor.channel_415nm)
F1 - 415nm/Violet  2076
>>> print("F2 - 445nm//Indigo %s" % sensor.channel_445nm)
F2 - 445nm//Indigo 9248
>>> print("F3 - 480nm//Blue   %s" % sensor.channel_480nm)
F3 - 480nm//Blue   10337
>>> print("F4 - 515nm//Cyan   %s" % sensor.channel_515nm)
F4 - 515nm//Cyan   16123
>>> print("F5 - 555nm/Green   %s" % sensor.channel_555nm)
F5 - 555nm/Green   23106
>>> print("F6 - 590nm/Yellow  %s" % sensor.channel_590nm)
F6 - 590nm/Yellow  28788
>>> print("F7 - 630nm/Orange  %s" % sensor.channel_630nm)
F7 - 630nm/Orange  33143
>>> print("F8 - 680nm/Red     %s" % sensor.channel_680nm)
F8 - 680nm/Red     7400
```

Example Code

```
# SPDX-FileCopyrightText: 2020 Bryan Siepert, written for Adafruit Industries
# SPDX-License-Identifier: MIT
from time import sleep
import board
from adafruit_as7341 import AS7341

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
sensor = AS7341(i2c)


def bar_graph(read_value):
    scaled = int(read_value / 1000)
    return "[%5d] " % read_value + (scaled * "*")


while True:
    print("F1 - 415nm/Violet  %s" % bar_graph(sensor.channel_415nm))
    print("F2 - 445nm//Indigo %s" % bar_graph(sensor.channel_445nm))
    print("F3 - 480nm//Blue   %s" % bar_graph(sensor.channel_480nm))
    print("F4 - 515nm//Cyan   %s" % bar_graph(sensor.channel_515nm))
    print("F5 - 555nm/Green   %s" % bar_graph(sensor.channel_555nm))
    print("F6 - 590nm/Yellow  %s" % bar_graph(sensor.channel_590nm))
    print("F7 - 630nm/Orange  %s" % bar_graph(sensor.channel_630nm))
    print("F8 - 680nm/Red     %s" % bar_graph(sensor.channel_680nm))
    print("Clear              %s" % bar_graph(sensor.channel_clear))
    print("Near-IR (NIR)      %s" % bar_graph(sensor.channel_nir))
    print("\n------------------------------------------------")
    sleep(1)
```

Python Docs

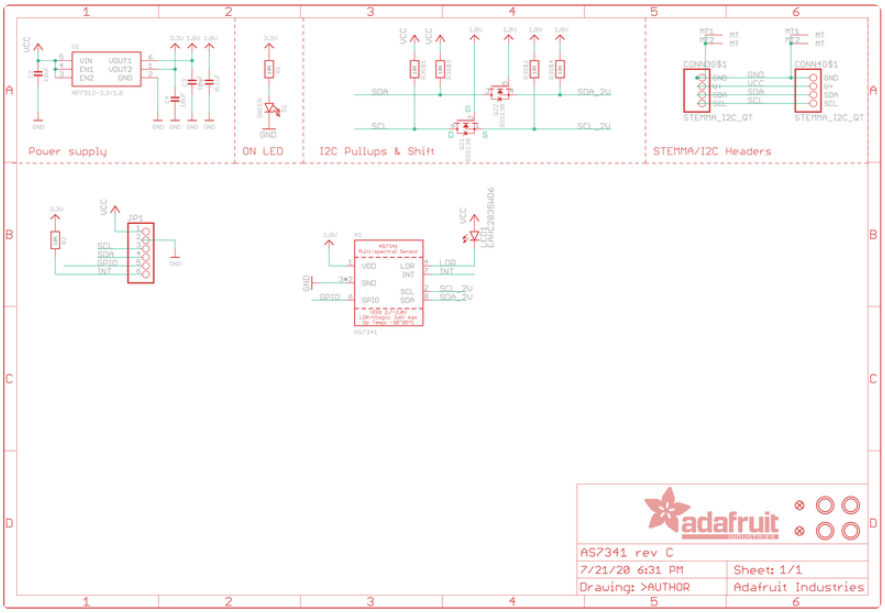
Python Docs ()

Downloads

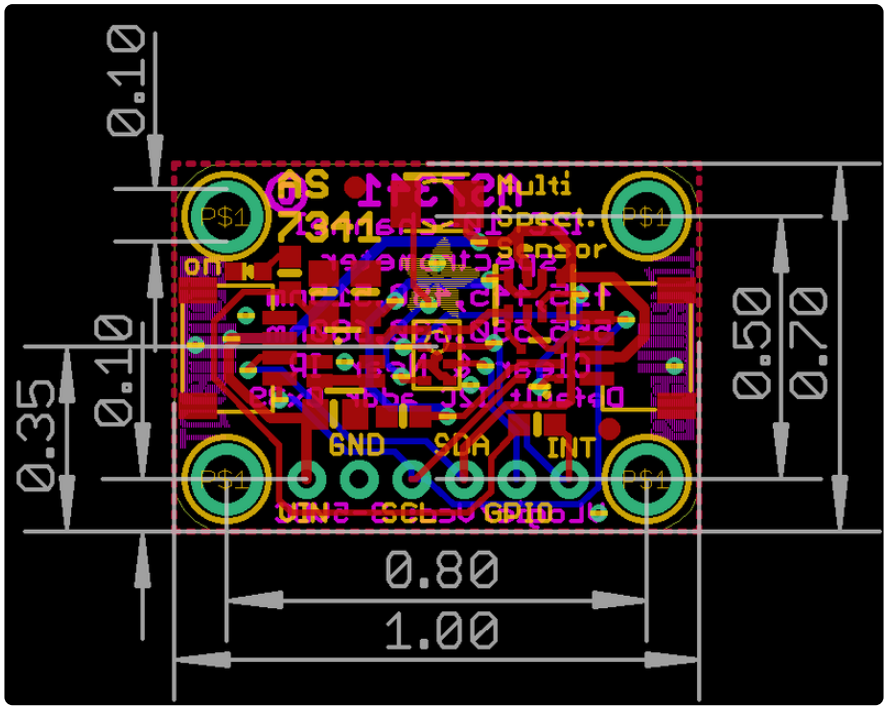
Files

- AS7341 Datasheet ()
- EagleCAD files on GitHub ()
- 3D Models on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()

Schematic



Fab Print



3D Model