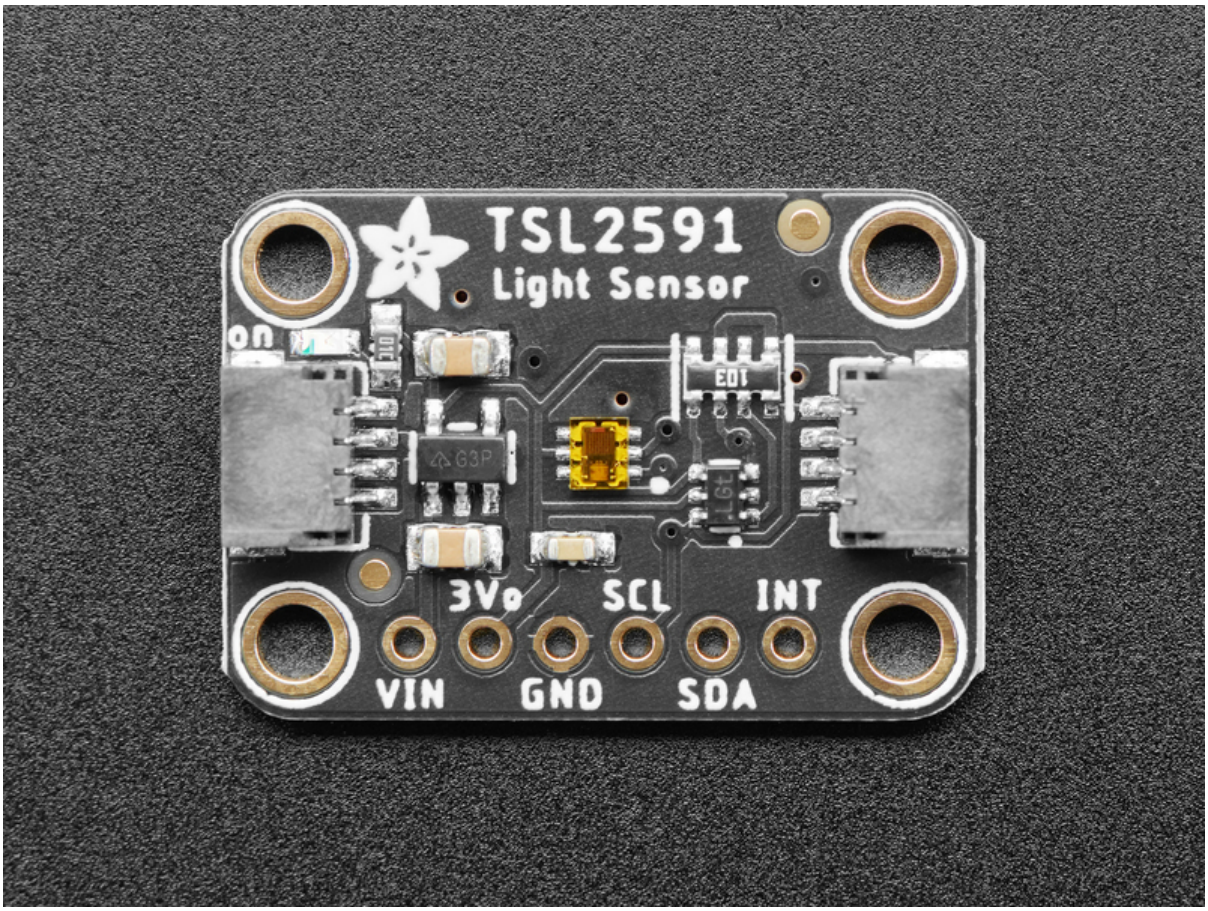
Adafruit TSL2591 High Dynamic Range Digital Light Sensor

Created by lady ada



https://learn.adafruit.com/adafruit-tsl2591

Last updated on 2024-06-03 01:32:44 PM EDT

Downloaded from Arrow.com.

Table of Contents

Overview 3

Pinouts 5

- Power Pins:
- I2C Logic pins:
- Other Pins:

Assembly 7

- Prepare the header strip:
- Add the breakout board:
- And Solder!

Wiring & Test 10

- Install Adafruit_TSL2591 library
- Install Adafruit_Sensor
- Load Demo
- Library Reference
- Constructor
- Gain and Timing
- Unified Sensor API
- void getEvent(sensors_event_t*)
- void getSensor(sensor_t*)
- Raw Data Access API

Arduino Library Docs 18

Python & CircuitPython 18

- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- CircuitPython Installation of TSL2591
- CircuitPython and Python Usage
- Full Example Code

Python Docs 24

WipperSnapper 24

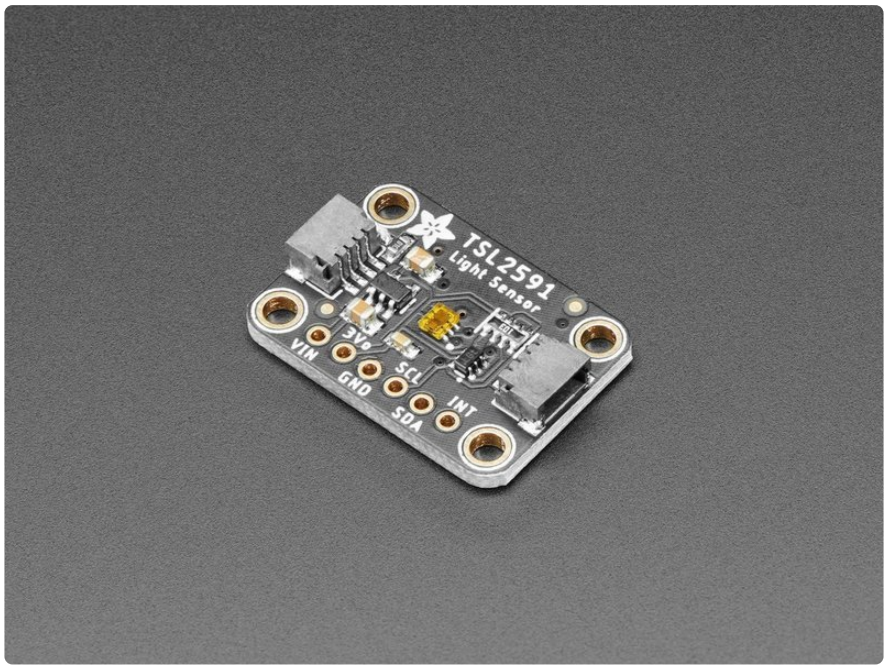
- What is WipperSnapper
- Wiring
- Usage

Downloads 30

- Datasheets & Files
- Schematic and Fab Print for STEMMA QT Version
- Schematic and Fab Print for Original Version

Downloaded from Arrow.com.

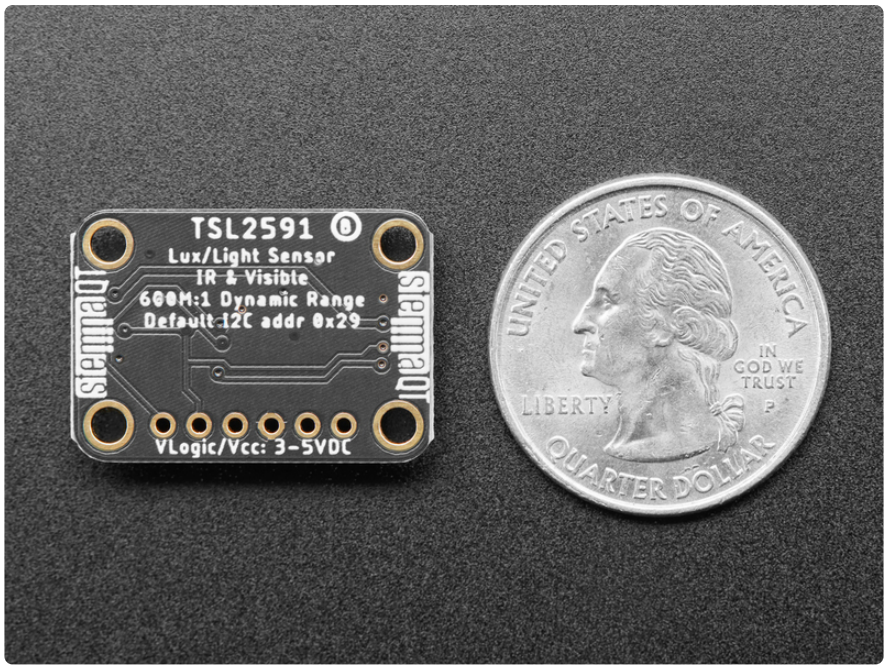
Overview



When the future is dazzlingly-bright, this ultra-high-range luminosity sensor will help you measure it. The TSL2591 luminosity sensor is an advanced digital light sensor, ideal for use in a wide range of light situations. Compared to low cost CdS cells, this sensor is more precise, allowing for exact lux calculations and can be configured for different gain/timing ranges to detect light ranges from up to 188uLux up to 88,000 Lux on the fly.

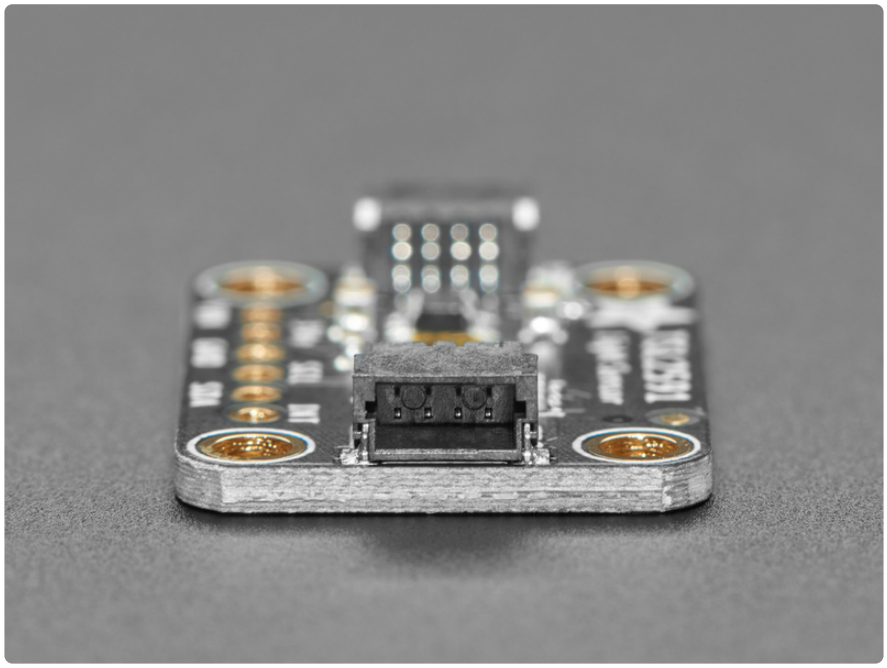
The best part of this sensor is that it **contains both infrared and full spectrum diodes**! That means you can separately measure infrared, full-spectrum or human-visible light. Most sensors can only detect one or the other, which does not accurately represent what human eyes see (since we cannot perceive the IR light that is detected by most photo diodes)

Downloaded from Arrow.com.



This sensor is much like the TSL2561 but with a wider range (and the interface code is different). This sensor has a massive 600,000,000:1 dynamic range! Unlike the TSL2561 you cannot change the I2C address either, so keep that in mind.

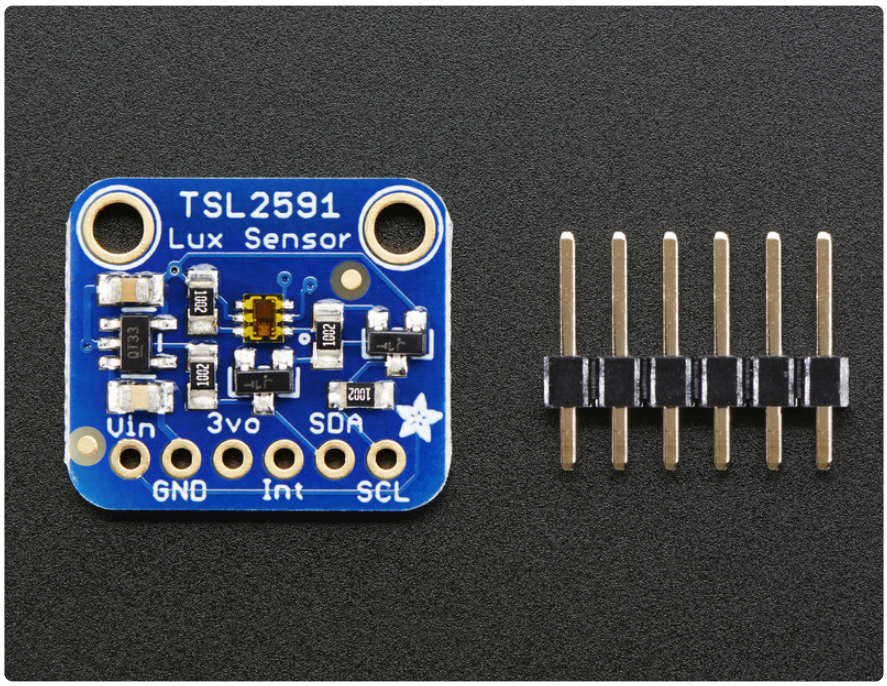
As if that weren't enough, we've also added SparkFun qwiic (https://adafru.it/Fpw) compatible STEMMA QT **(https://adafru.it/Ft4)** connectors for the I2C bus **so you don't even need to solder.** Just wire up to your favorite micro with a plug-and-play cable to get 6-DoF data ASAP. For a no-solder experience, just wire up to your favorite micro using a STEMMA QT adapter cable. (https://adafru.it/JnB) The Stemma QT connectors also mean the TSL2591 can be used with our various associated accessories. (https://adafru.it/Ft6)



Downloaded from Arrow.com.

The built in ADC means you can use this with any microcontroller, even if it doesn't have analog inputs. The current draw is extremely low, so its great for low power data-logging systems. about 0.4mA when actively sensing, and less than 5 uA when in power-down mode.

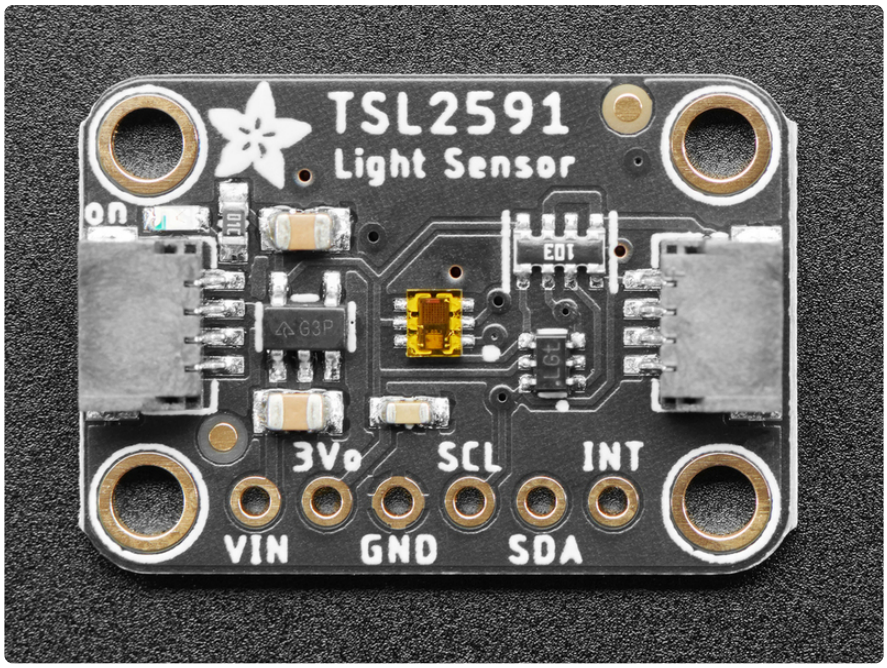
There are two versions of this board - the STEMMA QT version shown above, and the original header-only version shown below. Code works the same on both!

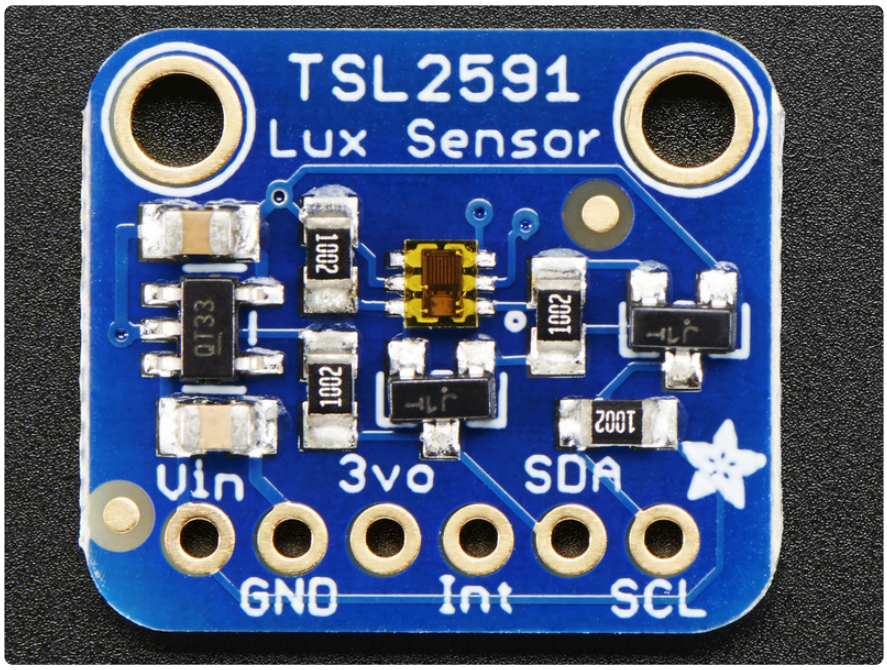


Pinouts

The TSL2591 is a I2C sensor. That means it uses the two I2C data/clock wires available on most microcontrollers, and can share those pins with other sensors as long as they don't have an address collision. For future reference, the I2C address is **0x29** and you can't change it!

Downloaded from Arrow.com.





Power Pins:

- **Vin** - this is the power pin. Since the chip uses 3 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 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

Downloaded from Arrow.com.

(https://adafru.it/dGy)I2C Logic pins:

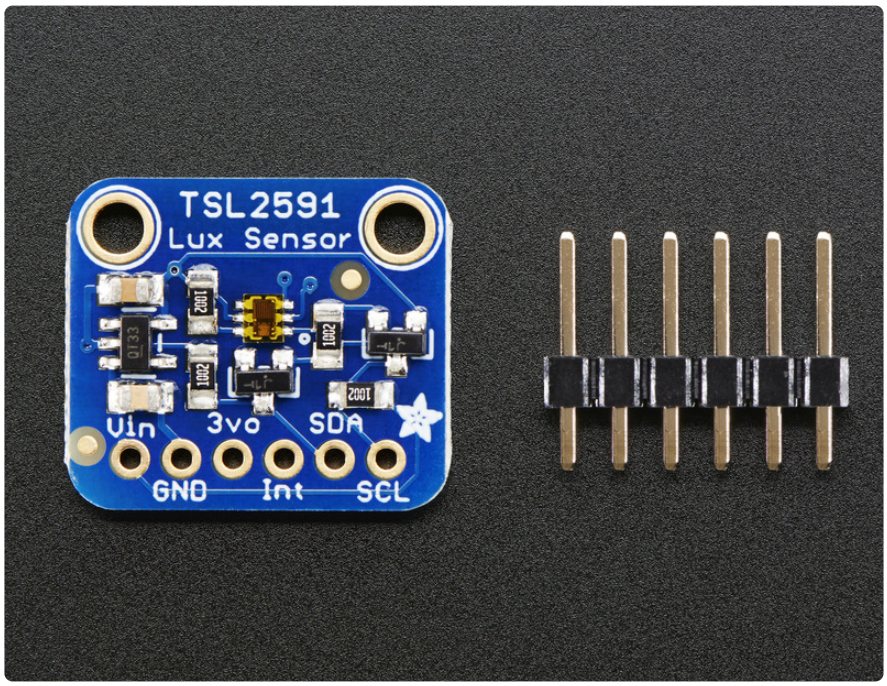
- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line.
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line.

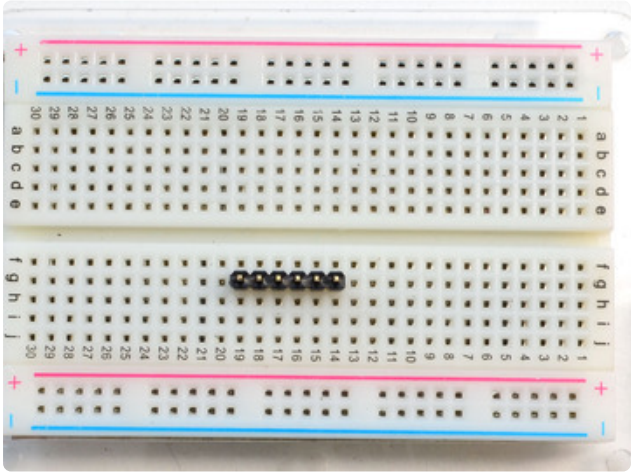
Other Pins:

- **INT** - this is the INTerrupt pin from the sensor. It can be programmed to do a couple different things by noodling with the i2c registers. For example trigger when a conversion is done, or when the light level has changed a lot, etc. We don't have library support for this pin

---

Assembly

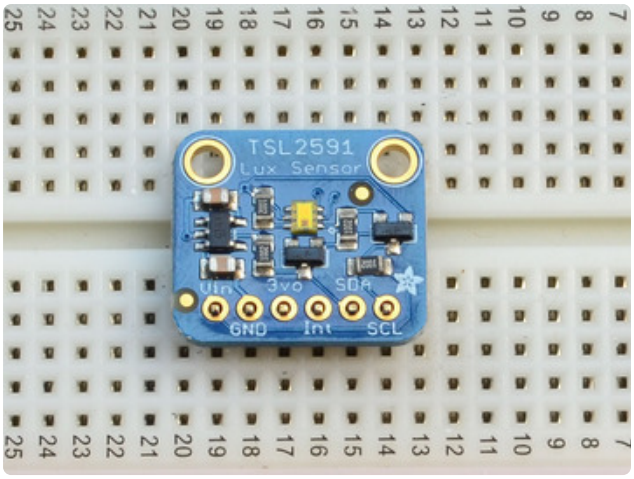




Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

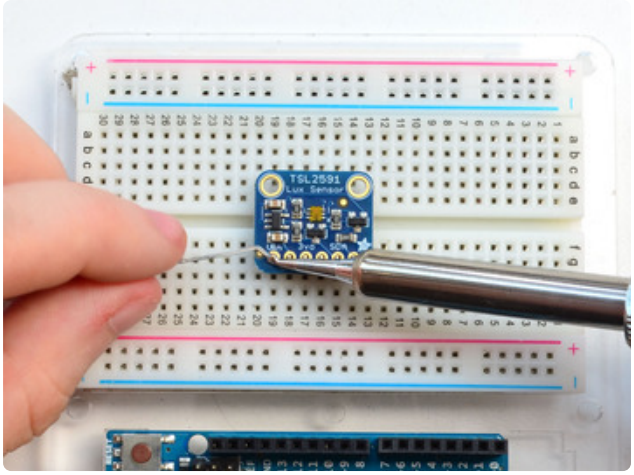
Downloaded from Arrow.com.

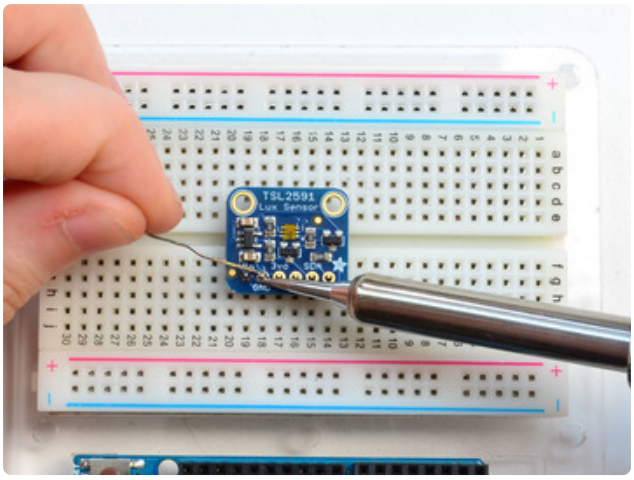


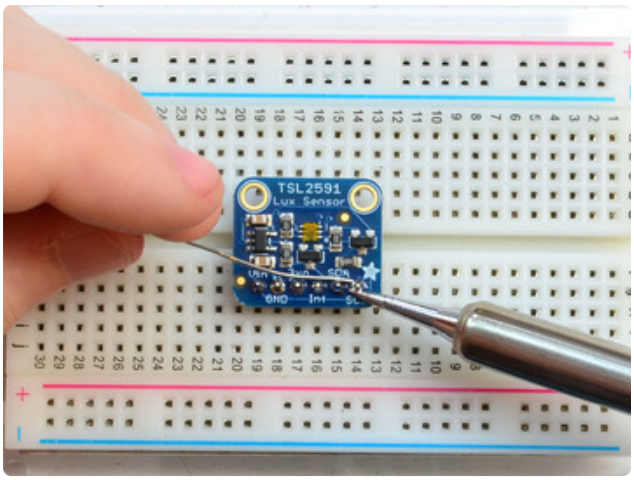
Add the breakout board:

Place the breakout board over the pins so that the short pins poke through the breakout pads

Downloaded from Arrow.com.





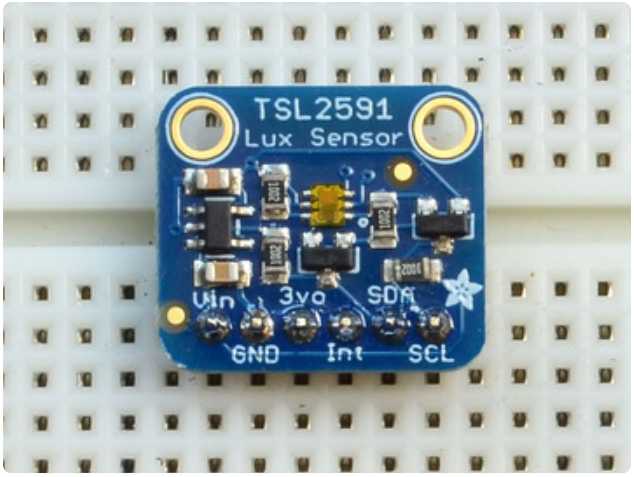


And Solder!

Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk)).

Downloaded from Arrow.com.

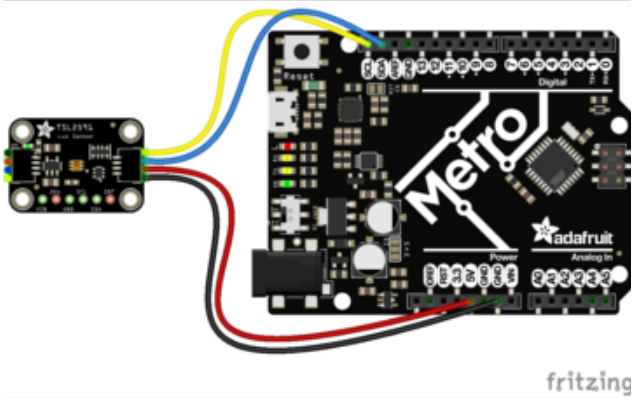


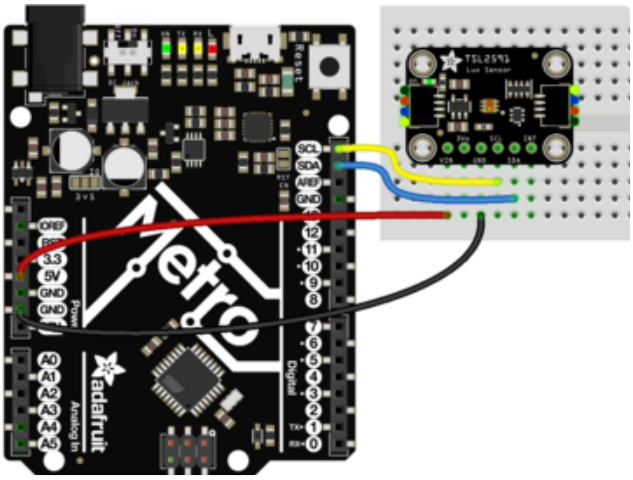
You're done! Check your solder joints visually and continue onto the next steps

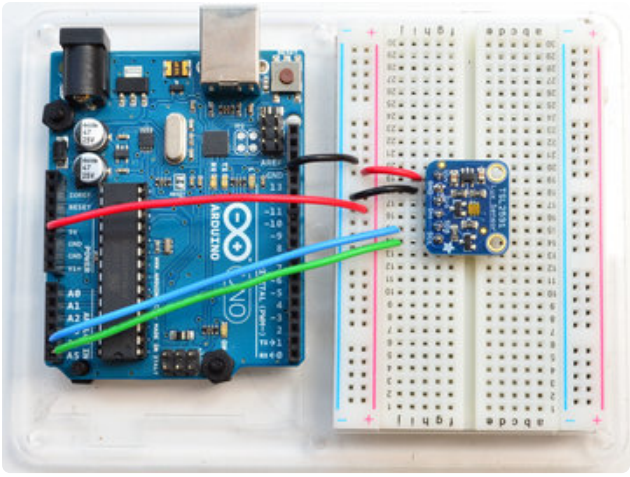
Wiring & Test

You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, just make sure it has I2C, then port the code - its pretty simple stuff!

Downloaded from Arrow.com.







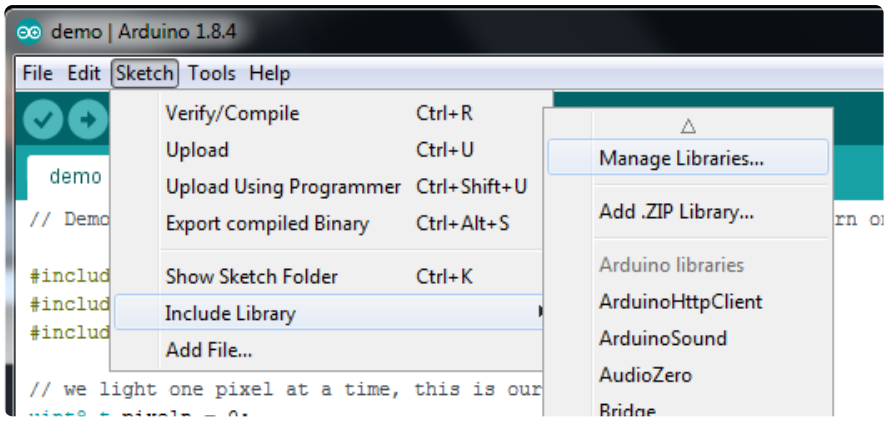
Connect **Vin** to the power supply, 3-5V is fine. **(red wire on STEMMA QT version)** Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
Connect **GND** to common power/data ground **(black wire on STEMMA QT version)**
Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino. **(yellow wire on STEMMA QT version)** On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**
Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino. **(blue wire on STEMMA QT version)** On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**
The TSL2591 has a default I2C address of BOTH **0x29** and **0x28** (which we don't use in code but it definitely is taken) and these cannot be changed!

Install Adafruit_TSL2591 library

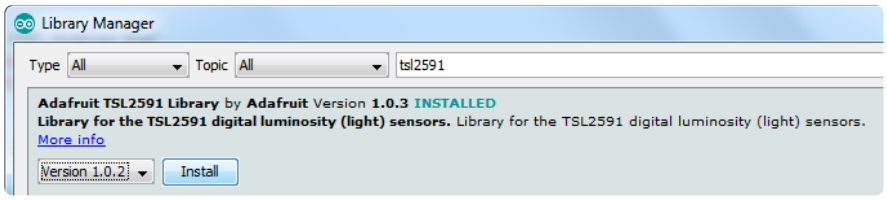
To begin reading sensor data, you will need to install the Adafruit_TSL2591 library (code on our github repository) (https://adafru.it/dGz). It is available from the Arduino library manager so we recommend using that.

From the IDE open up the library manager...

Downloaded from Arrow.com.



And type in **adafruit tsl2591** to locate the library. Click **Install**

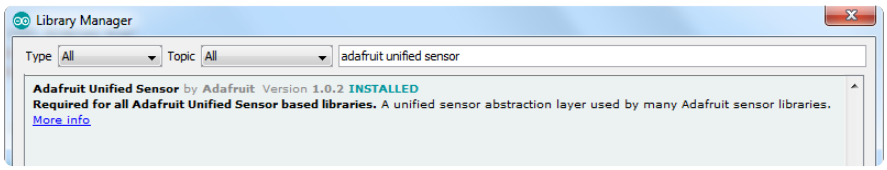


We also have a great tutorial on Arduino library installation at: http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (https://adafru.it/aYM)

Install Adafruit_Sensor

The TSL2591 library uses the Adafruit_Sensor support backend (https://adafru.it/aZm) so that readings can be normalized between sensors.

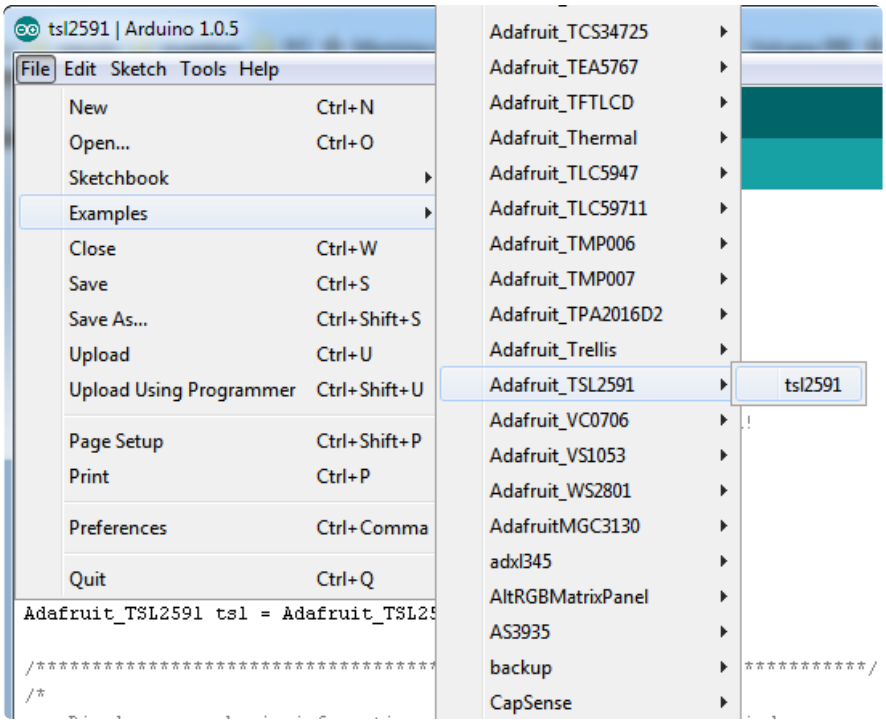
Search the library manager for **Adafruit Unified Sensor** and install that too (you may have to scroll a bit)



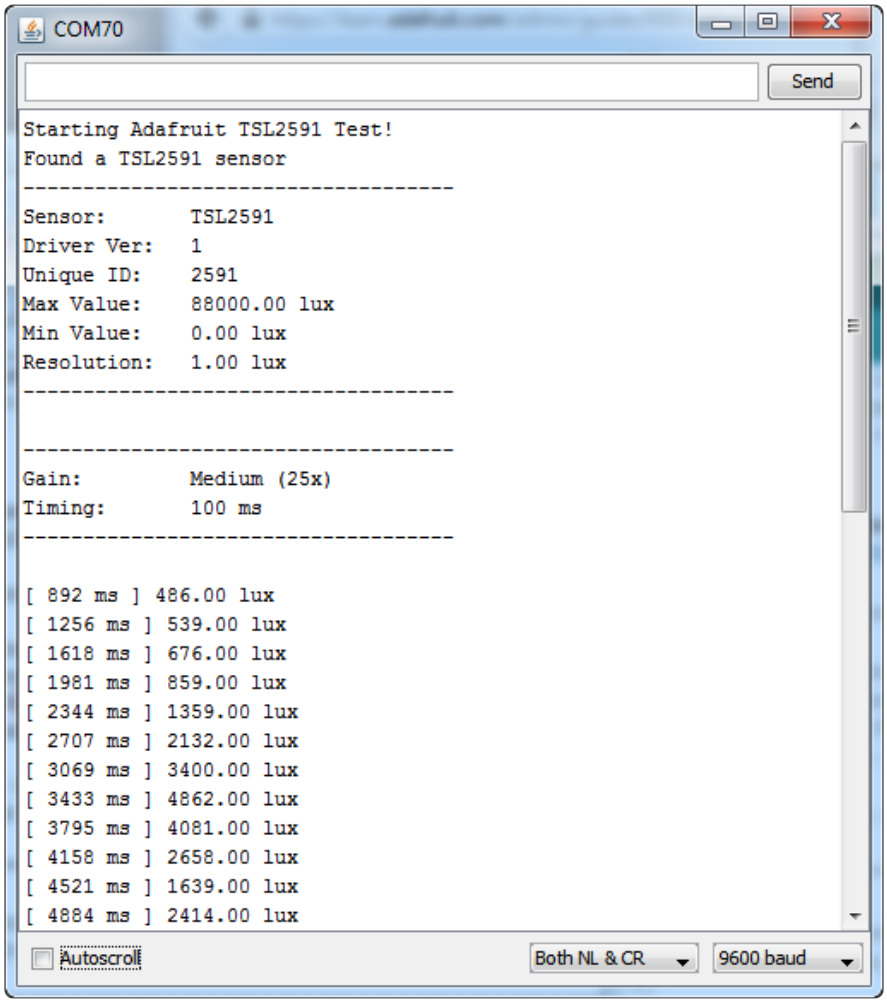
Load Demo

Open up **File->Examples->Adafruit_TSL2591->tsl2591** and upload to your Arduino wired up to the sensor

Downloaded from Arrow.com.



Thats it! Now open up the serial terminal window at 9600 speed to begin the test.



Downloaded from Arrow.com.

Try covering with your hand or shining a lamp onto the sensor to experiment with the light levels!

Library Reference

The **Adafruit_TSL2591** library contains a number of public functions to help you get started with this sensor.

Constructor

To create an instance of the Adafruit_TSL2591 driver, simple declare an appropriate object, along with a 32-bit numeric value to identify this sensor (in case you have several TSL2591s and want to track them separately in a logging system).

```
Adafruit_TSL2591 tsl = Adafruit_TSL2591(2591);
```

Gain and Timing

You can adjust the gain settings and integration time of the sensor to make it more or less sensitive to light, depending on the environment where the sensor is being used.

The gain can be set to one of the following values (though the last value, MAX, has limited use in the real world given the extreme amount of gain applied):

- **TSL2591_GAIN_LOW**: Sets the gain to 1x (bright light)
- **TSL2591_GAIN_MEDIUM**: Sets the gain to 25x (general purpose)
- **TSL2591_GAIN_HIGH**: Sets the gain to 428x (low light)
- **TSL2591_GAIN_MAX**: Sets the gain to 9876x (extremely low light)

Gain can be read or set via the following functions:

- **void setGain(tsl2591Gain_t gain);**
- **tsl2591Gain_t getGain();**

The integration time can be set between 100 and 600ms, and the longer the integration time the more light the sensor is able to integrate, making it more sensitive in low light the longer the integration time. The following values can be used:

- **TSL2591_INTEGRATIONTIME_100MS**
- **TSL2591_INTEGRATIONTIME_200MS**
- **TSL2591_INTEGRATIONTIME_300MS**
- **TSL2591_INTEGRATIONTIME_400MS**

Downloaded from Arrow.com.

- **TSL2591_INTEGRATIONTIME_500MS**
- **TSL2591_INTEGRATIONTIME_600MS**

The integration time can be read or set via the following functions:

- **void setTiming (tsl2591IntegrationTime_t integration);**
- **tsl2591IntegrationTime_t getTiming();**

An example showing how these functions are used can be seen in the code below:

```
/**************************************************************************/
/*
    Configures the gain and integration time for the TSL2561
*/
/**************************************************************************/
void configureSensor(void)
{
  // You can change the gain on the fly, to adapt to brighter/dimmer light
situations
  //tsl.setGain(TSL2591_GAIN_LOW);    // 1x gain (bright light)
  tsl.setGain(TSL2591_GAIN_MED);      // 25x gain
  //tsl.setGain(TSL2591_GAIN_HIGH);   // 428x gain

  // Changing the integration time gives you a longer time over which to sense light
  // longer timelines are slower, but are good in very low light situtations!
  tsl.setTiming(TSL2591_INTEGRATIONTIME_100MS);  // shortest integration time
(bright light)
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_200MS);
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_300MS);
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_400MS);
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_500MS);
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_600MS);  // longest integration time (dim
light)

  /* Display the gain and integration time for reference sake */
  Serial.println("------------------------------------");
  Serial.print  ("Gain:         ");
  tsl2591Gain_t gain = tsl.getGain();
  switch(gain)
  {
    case TSL2591_GAIN_LOW:
      Serial.println("1x (Low)");
      break;
    case TSL2591_GAIN_MED:
      Serial.println("25x (Medium)");
      break;
    case TSL2591_GAIN_HIGH:
      Serial.println("428x (High)");
      break;
    case TSL2591_GAIN_MAX:
      Serial.println("9876x (Max)");
      break;
  }
  Serial.print  ("Timing:       ");
  Serial.print((tsl.getTiming() + 1) * 100, DEC);
  Serial.println(" ms");
  Serial.println("------------------------------------");
  Serial.println("");
}
```


Downloaded from Arrow.com.

Unified Sensor API

The Adafruit_TSL2591 library makes use of the Adafruit unified sensor framework (https://adafru.it/dGB) to provide sensor data in a standardized format and scale. If you wish to make use of this framweork, the two key functions that you need to work with are **getEvent** and **getSensor**, as described below:

void getEvent(sensors_event_t*)

This function will read a single sample from the sensor and return it in a generic sensors_event_t object. To use this function, you simply pass in a sensors_event_t reference, which will be populated by the function, and then read the results, as shown in the following code:

```
/**************************************************************************/
/*
    Performs a read using the Adafruit Unified Sensor API.
*/
/**************************************************************************/
void unifiedSensorAPIRead(void)
{
  /* Get a new sensor event */
  sensors_event_t event;
  tsl.getEvent(&amp;event);

  /* Display the results (light is measured in lux) */
  Serial.print("[ "); Serial.print(event.timestamp); Serial.print(" ms ] ");
  if ((event.light == 0) |
      (event.light &gt; 4294966000.0) |
      (event.light &lt;-4294966000.0))
  {
    /* If event.light = 0 lux the sensor is probably saturated */
    /* and no reliable data could be generated! */
    /* if event.light is +/- 4294967040 there was a float over/underflow */
    Serial.println("Invalid data (adjust gain or timing)");
  }
  else
  {
    Serial.print(event.light); Serial.println(" lux");
  }
}
```

Note that some checks need to be performed on the sensor data in case the sensor saturated. If saturation happens, please adjust the gain and integration time up or down to change the sensor's sensitivity and output range.

void getSensor(sensor_t*)

This function returns some basic information about the sensor, and operates in a similar fashion to getEvent. You pass in an empty sensor_t reference, which will be populated by this function, and we can then read the results and retrieve some key details about the sensor and driver, as shown in the code below:

Downloaded from Arrow.com.

```
/**************************************************************************/
/*
    Displays some basic information on this sensor from the unified
    sensor API sensor_t type (see Adafruit_Sensor for more information)
*/
/**************************************************************************/
void displaySensorDetails(void)
{
  sensor_t sensor;
  tsl.getSensor(&amp;sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value);
Serial.println(" lux");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value);
Serial.println(" lux");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution);
Serial.println(" lux");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}
```

Raw Data Access API

If you don't wish to use the Unified Sensor API, you can access the raw data for this sensor via the following three functions:

- **uint16_t getLuminosity (uint8_t channel);**
- **uint32_t getFullLuminosity ( );**
- **uint32_t calculateLux (uint16_t ch0, uint16_t ch1);**

getLuminosity can be used to read either the visible spectrum light sensor, or the infrared light sensor. It will return the raw 16-bit sensor value for the specified channel, as shown in the code below:

```
/**************************************************************************/
/*
    Shows how to perform a basic read on visible, full spectrum or
    infrared light (returns raw 16-bit ADC values)
*/
/**************************************************************************/
void simpleRead(void)
{
  // Simple data read example. Just read the infrared, fullspecrtrum diode
  // or 'visible' (difference between the two) channels.
  // This can take 100-600 milliseconds! Uncomment whichever of the following you
want to read
  uint16_t x = tsl.getLuminosity(TSL2591_VISIBLE);
  //uint16_t x = tsl.getLuminosity(TSL2561_FULLSPECTRUM);
  //uint16_t x = tsl.getLuminosity(TSL2561_INFRARED);

  Serial.print("[ "); Serial.print(millis()); Serial.print(" ms ] ");
  Serial.print("Luminosity: ");
  Serial.println(x, DEC);
}
```


Downloaded from Arrow.com.

getFullLuminosity reads both the IR and full spectrum sensors at the same time to allow tigher correlation between the values, and then separates them in SW. The function returns a 32-bit value which needs to be split into two 16-bit values, as shown in the code below:

```
/**************************************************************************/
/*
    Show how to read IR and Full Spectrum at once and convert to lux
*/
/**************************************************************************/
void advancedRead(void)
{
  // More advanced data read example. Read 32 bits with top 16 bits IR, bottom 16
bits full spectrum
  // That way you can do whatever math and comparisons you want!
  uint32_t lum = tsl.getFullLuminosity();
  uint16_t ir, full;
  ir = lum &gt;&gt; 16;
  full = lum &amp; 0xFFFF;
  Serial.print("[ "); Serial.print(millis()); Serial.print(" ms ] ");
  Serial.print("IR: "); Serial.print(ir);  Serial.print("  ");
  Serial.print("Full: "); Serial.print(full); Serial.print("  ");
  Serial.print("Visible: "); Serial.print(full - ir); Serial.print("  ");
  Serial.print("Lux: "); Serial.println(tsl.calculateLux(full, ir));
}
```

calculateLux can be used to take both the infrared and visible spectrum sensor data and roughly correlate with the equivalent SI lux value, based on a formula from the silicon vendor that takes into account the sensor properties and the integration time and gain settings of the device.

To calculate the lux, simple call **calculateLux(full, ir)**, where 'full' and 'ir' are raw 16-bit values taken from one of the two raw data functions above. See the code sample above for an example of calculating lux.

Arduino Library Docs

Arduino Library Docs (https://adafru.it/AvJ)

Python & CircuitPython

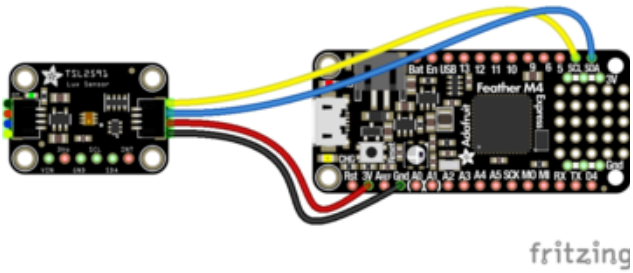
It's easy to use the TSL2591 sensor with Python and CircuitPython and the Adafruit CircuitPython TSL2591 (https://adafru.it/C4l) module. This module allows you to easily write Python code that reads the luminosity and more from the sensor.

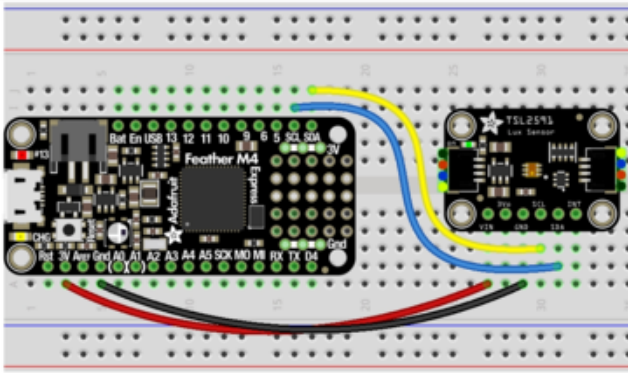
You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

Downloaded from Arrow.com.

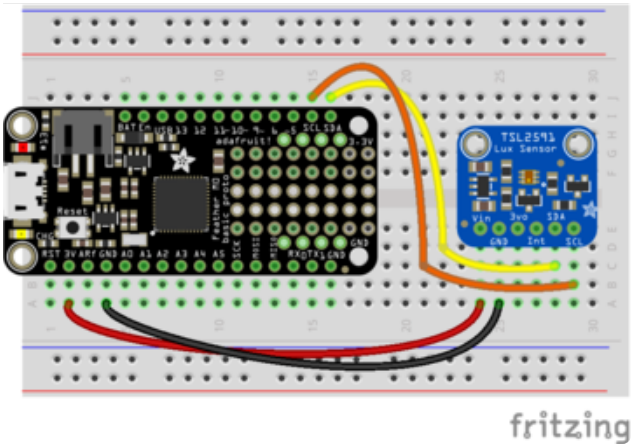
CircuitPython Microcontroller Wiring

First wire up a TSL2591 to your board exactly as shown on the previous pages for Arduino using an I2C connection. Here's an example of wiring a Feather M0 to the sensor with I2C:





Board 3V to **sensor VIN (red wire on STEMMA QT version)**
Board GND to **sensor GND (black wire on STEMMA QT version)**
Board SCL to **sensor SCL (yellow wire on STEMMA QT version)**
Board SDA to **sensor SDA (blue wire on STEMMA QT version)**

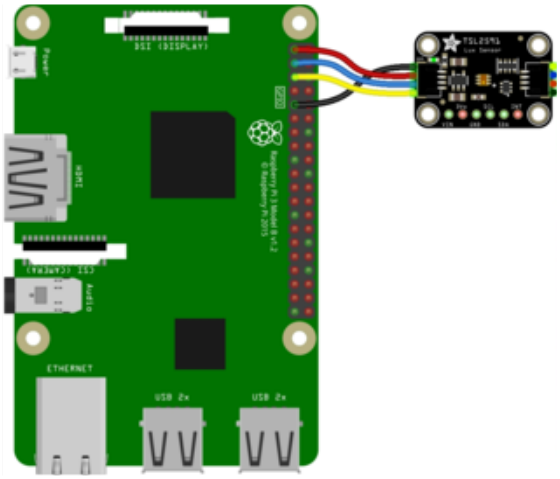


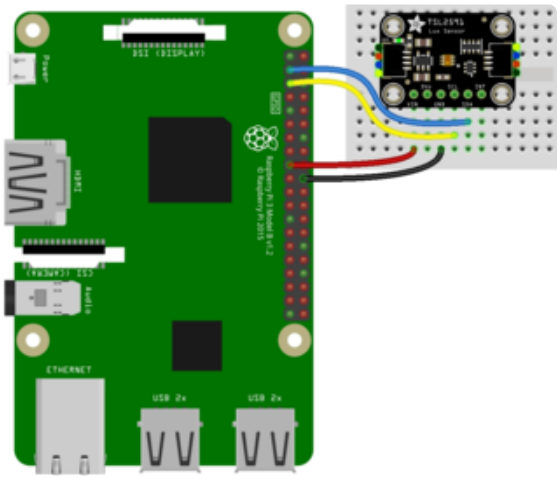
Downloaded from Arrow.com.

Python Computer Wiring

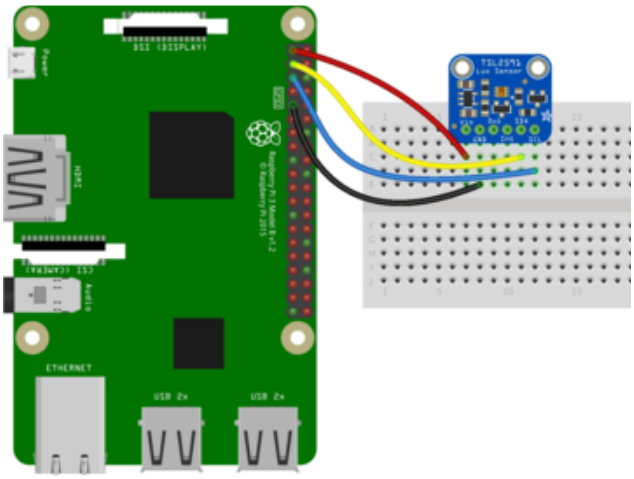
Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C:





Pi 3V3 to **sensor VIN (red wire on STEMMA QT version)**
Pi GND to **sensor GND (black wire on STEMMA QT version)**
Pi SCL to **sensor SCK (yellow wire on STEMMA QT version)**
Pi SDA to **sensor SDA (blue wire on STEMMA QT version)**



Downloaded from Arrow.com.

CircuitPython Installation of TSL2591

Next you'll need to install the Adafruit CircuitPython TSL2591 (https://adafru.it/C4I) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) 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 (https://adafru.it/zdx). Our introduction guide has a great page on how to install the library bundle (https://adafru.it/ABU) for both express and non-express boards.

Remember for non-express boards like the, you'll need to manually install the necessary libraries from the bundle:

- **adafruit_tsl2591.mpy**
- **adafruit_bus_device**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_tsl2591.mpy,** and **adafruit_bus_device** files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz)so you are at the CircuitPython >>> prompt.

CircuitPython and Python Usage

To demonstrate the usage of the sensor we'll initialize it and read the luminosity from the board's Python REPL. Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
import adafruit_tsl2591
i2c = board.I2C()
sensor = adafruit_tsl2591.TSL2591(i2c)
```

Now you're ready to read values from the sensor using any of these properties:

- **lux** - The computed light lux value measured by the sensor.

Downloaded from Arrow.com.

- **visible** - The visible light level measured by the sensor. This is a 32-bit unsigned value with no units, the higher the number the more visible light.
- **infrared** - The infrared light level measured by the sensor. This is a 16-bit unsigned value with no units, the higher the number the more infrared light.
- **full_spectrum** - The visible & infrared light level measured by the sensor as a single 32-bit value with no units. The higher the number the more full spectrum light.
- **raw_luminosity** - A 2-tuple of raw sensor visible+IR and IR only light levels. Each is a 16-bit value with no units where the higher the value the more light.

```
print('Light: {0}lux'.format(sensor.lux))
print('Visible: {0}'.format(sensor.visible))
print('Infrared: {0}'.format(sensor.infrared))
```

In addition you can read and write a few properties to change how the sensor behaves and is configured:

- `gain` - Set this to a value below to change the gain of the light sensor:
  - `adafruit_tsl2591.GAIN_LOW` - 1x gain
  - `adafruit_tsl2591.GAIN_MED` - 25x gain (the default)
  - `adafruit_tsl2591.GAIN_HIGH` - 428x gain
  - `adafruit_tsl2591.GAIN_MAX` - 9876x gain
- `integration_time` - Set the integration time of the sensor to a value of:
  - `adafruit_tsl2591.INTEGRATIONTIME_100MS` - 100ms (the default)
  - `adafruit_tsl2591.INTEGRATIONTIME_200MS` - 200ms
  - `adafruit_tsl2591.INTEGRATIONTIME_300MS` - 300ms
  - `adafruit_tsl2591.INTEGRATIONTIME_400MS` - 400ms
  - `adafruit_tsl2591.INTEGRATIONTIME_500MS` - 500ms
  - `adafruit_tsl2591.INTEGRATIONTIME_600MS` - 600ms

```
sensor.gain = adafruit_tsl2591.GAIN_LOW
sensor.integration_time = adafruit_tsl2591.INTEGRATIONTIME_200MS
```


Downloaded from Arrow.com.

```
>>> sensor.gain = adafruit_tsl2591.GAIN_LOW
>>> sensor.integration_time = adafruit_tsl2591.INTEGRATIONTIME_200MS
>>> print('Light: {0}lux'.format(sensor.lux))
Light: 281.112lux
>>> print('Visible: {0}'.format(sensor.visible))
Visible: 1966237
>>> print('Infrared: {0}'.format(sensor.infrared))
Infrared: 30
>>>
```

That's all there is to using the TSL2591 sensor with CircuitPython!

Below is a complete example that will print the light level every second. Save this as **code.py** on the board and view the output in the REPL.

Full Example Code

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

# Simple demo of the TSL2591 sensor.  Will print the detected light value
# every second.
import time
import board
import adafruit_tsl2591

# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller

# Initialize the sensor.
sensor = adafruit_tsl2591.TSL2591(i2c)

# You can optionally change the gain and integration time:
# sensor.gain = adafruit_tsl2591.GAIN_LOW (1x gain)
# sensor.gain = adafruit_tsl2591.GAIN_MED (25x gain, the default)
# sensor.gain = adafruit_tsl2591.GAIN_HIGH (428x gain)
# sensor.gain = adafruit_tsl2591.GAIN_MAX (9876x gain)
# sensor.integration_time = adafruit_tsl2591.INTEGRATIONTIME_100MS (100ms, default)
# sensor.integration_time = adafruit_tsl2591.INTEGRATIONTIME_200MS (200ms)
# sensor.integration_time = adafruit_tsl2591.INTEGRATIONTIME_300MS (300ms)
# sensor.integration_time = adafruit_tsl2591.INTEGRATIONTIME_400MS (400ms)
# sensor.integration_time = adafruit_tsl2591.INTEGRATIONTIME_500MS (500ms)
# sensor.integration_time = adafruit_tsl2591.INTEGRATIONTIME_600MS (600ms)

# Read the total lux, IR, and visible light levels and print it every second.
while True:
    # Read and calculate the light level in lux.
    lux = sensor.lux
    print("Total light: {0}lux".format(lux))
    # You can also read the raw infrared and visible light levels.
    # These are unsigned, the higher the number the more light of that type.
    # There are no units like lux.
    # Infrared levels range from 0-65535 (16-bit)
    infrared = sensor.infrared
    print("Infrared light: {0}".format(infrared))
    # Visible-only levels range from 0-2147483647 (32-bit)
    visible = sensor.visible
    print("Visible light: {0}".format(visible))
    # Full spectrum (visible + IR) also range from 0-2147483647 (32-bit)
    full_spectrum = sensor.full_spectrum
```

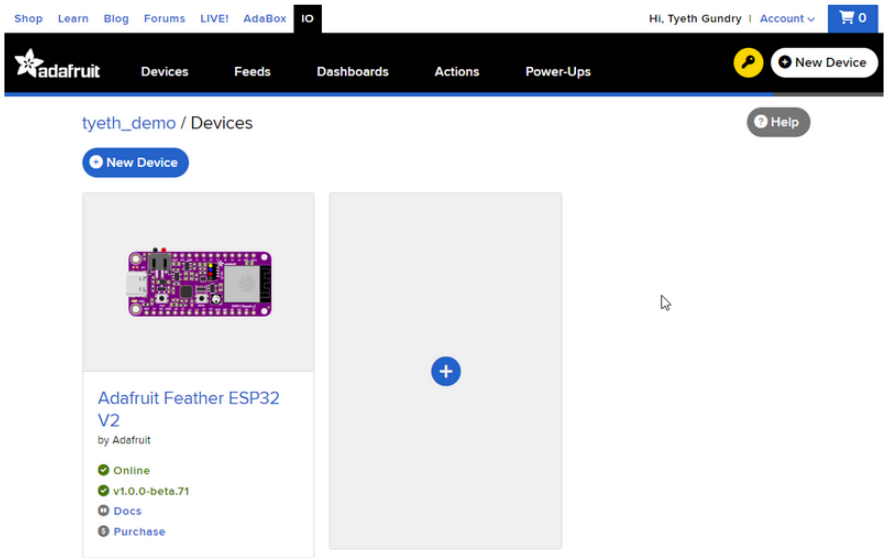

Downloaded from Arrow.com.

```
    print("Full spectrum (IR + visible) light: {0}".format(full_spectrum))
    time.sleep(1.0)
```

Python Docs

Python Docs (https://adafru.it/C4n)

WipperSnapper



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), a web platform designed (by Adafruit! (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.

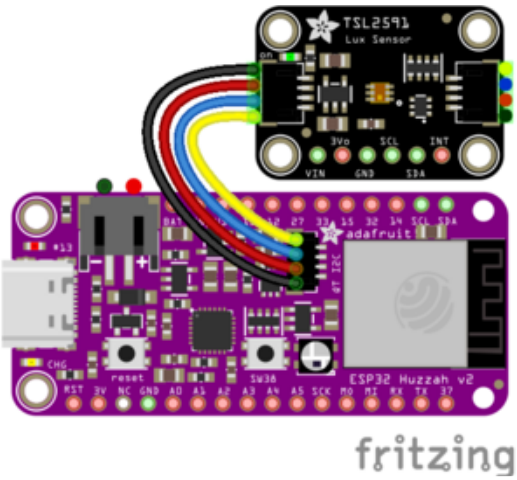
Downloaded from Arrow.com.

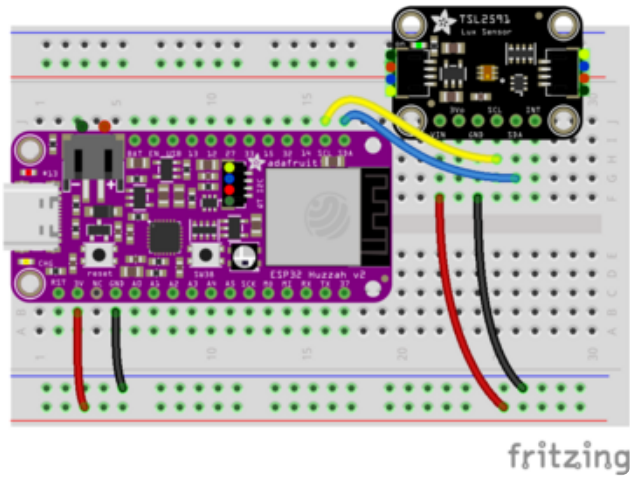
Quickstart: Adafruit IO WipperSnapper

https://adafru.it/Vfd

Wiring

First, wire up an TSL2591 to your board exactly as follows. Here is an example of the TSL2591 wired to an Adafruit ESP32 Feather V2 (http://adafru.it/5400) using I2C with a STEMMA QT cable (no soldering required) (http://adafru.it/4210)





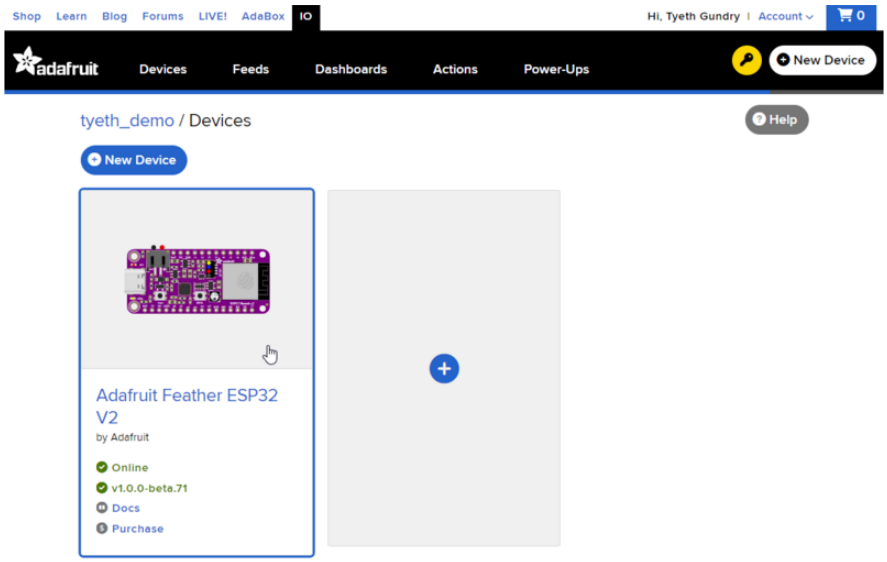
Board 3V to **sensor VIN (red wire on STEMMA QT)**
Board GND to **sensor GND (black wire on STEMMA QT)**
Board SCL to **sensor SCL (yellow wire on STEMMA QT)**
Board SDA to **sensor SDA (blue wire on STEMMA QT)**

Usage

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

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

Downloaded from Arrow.com.



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

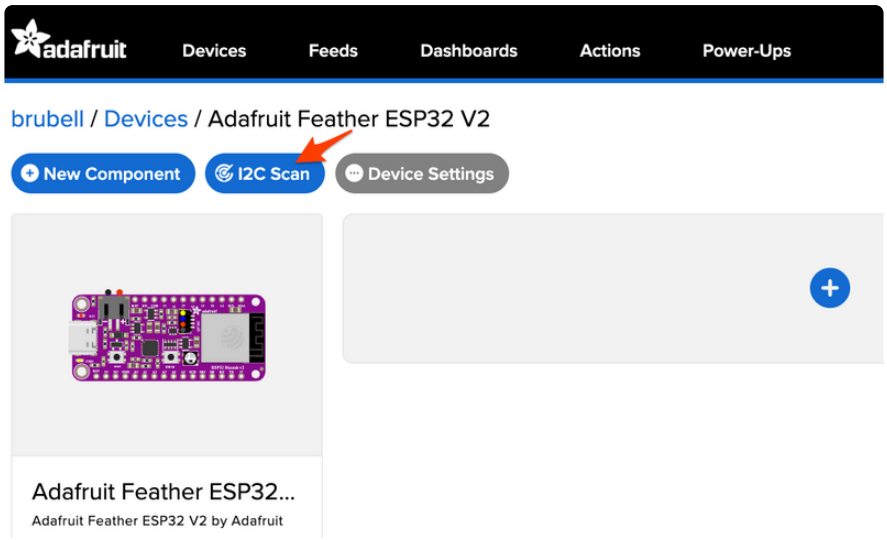
On the device page, quickly **check that you're running the latest version of the WipperSnapper firmware**.

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

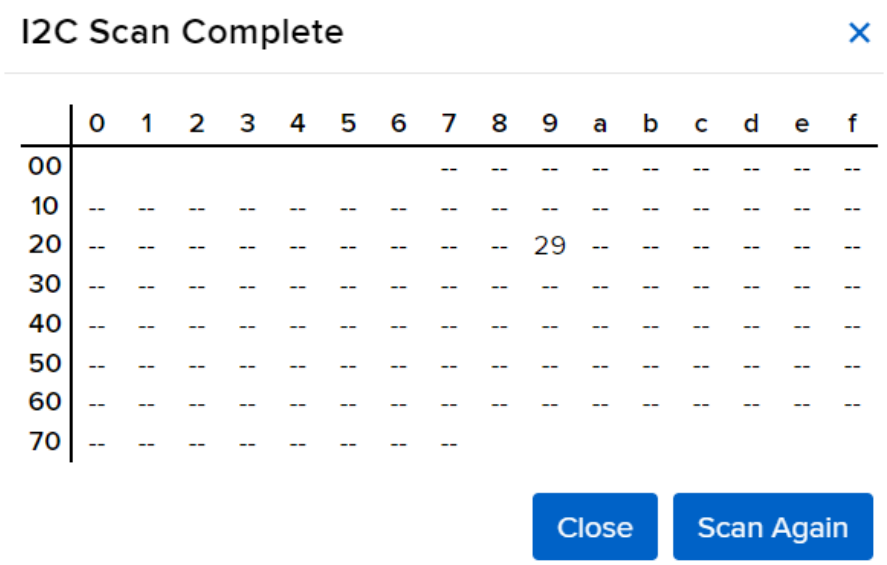
If the firmware version is green with a checkmark - continue with this guide.
If the firmware version is red with an exclamation mark "!" - update to the latest WipperSnapper firmware (https://adafru.it/Vfd) on your board before continuing.

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

Downloaded from Arrow.com.



You should see the TSL2591's default I2C address of `0x29` pop-up in the I2C scan list.



	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00								--	--	--	--	--	--	--	--	--
10	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20	--	--	--	--	--	--	--	--	--	29	--	--	--	--	--	--
30	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70	--	--	--	--	--	--	--	--								

I don't see the sensor's I2C address listed!

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

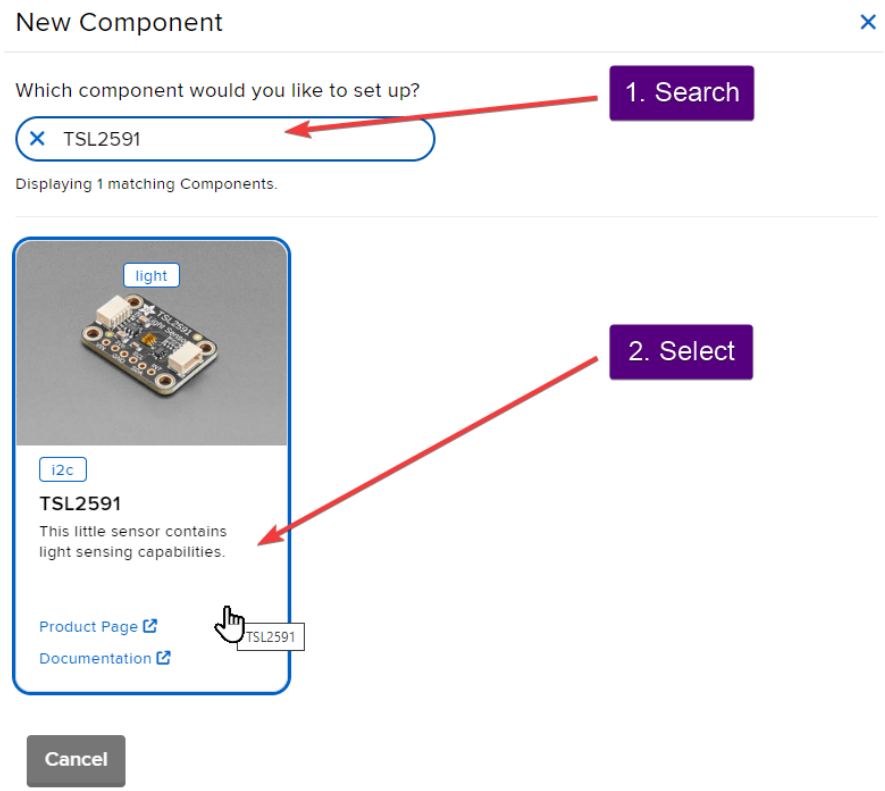
Then, reset the board and let it re-connect to Adafruit IO WipperSnapper.

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.

Downloaded from Arrow.com.

Adafruit IO supports a large amount of components. To quickly find your sensor, type `TSL2591` into the search bar, then select the **TSL2591** component.

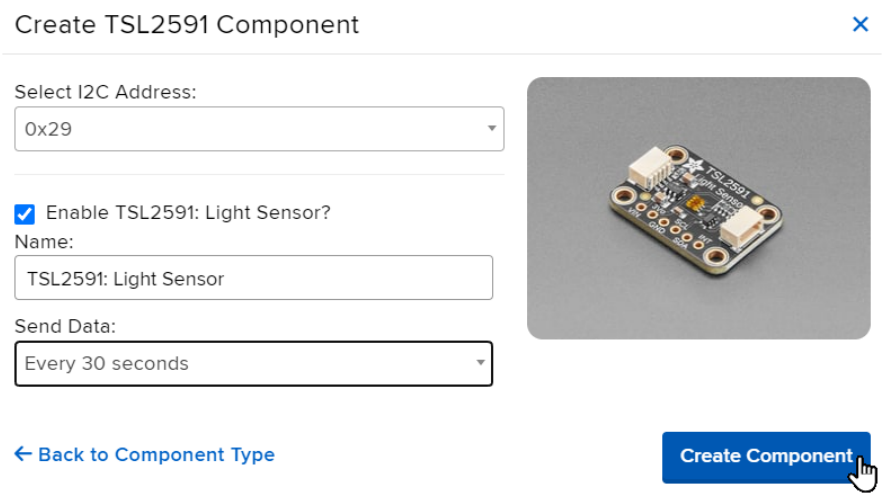


On the component configuration page, the TSL2591's sensor address should be listed along with the sensor's settings.

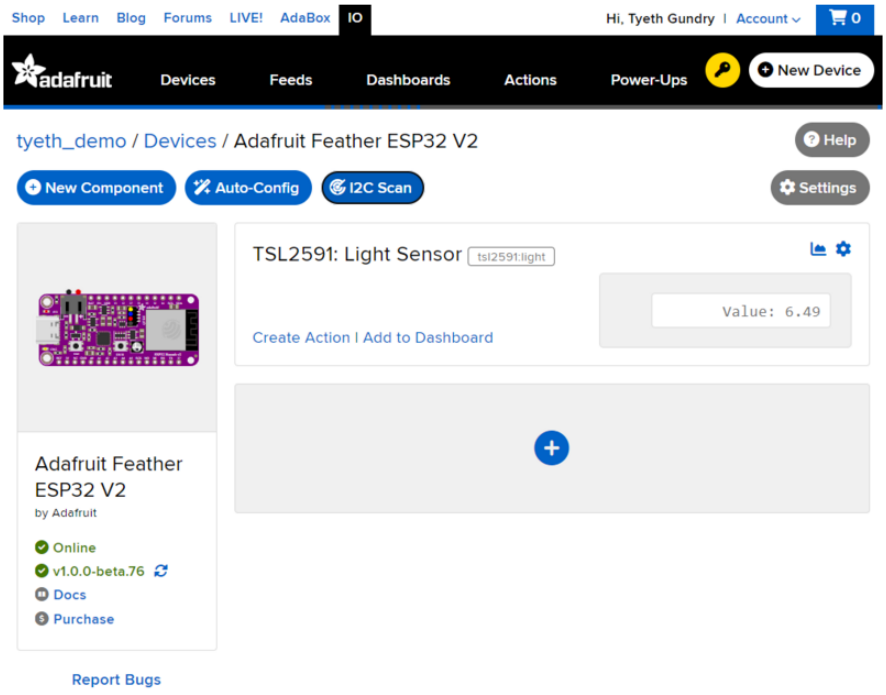
The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the TSL2591 sensor and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

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

Downloaded from Arrow.com.

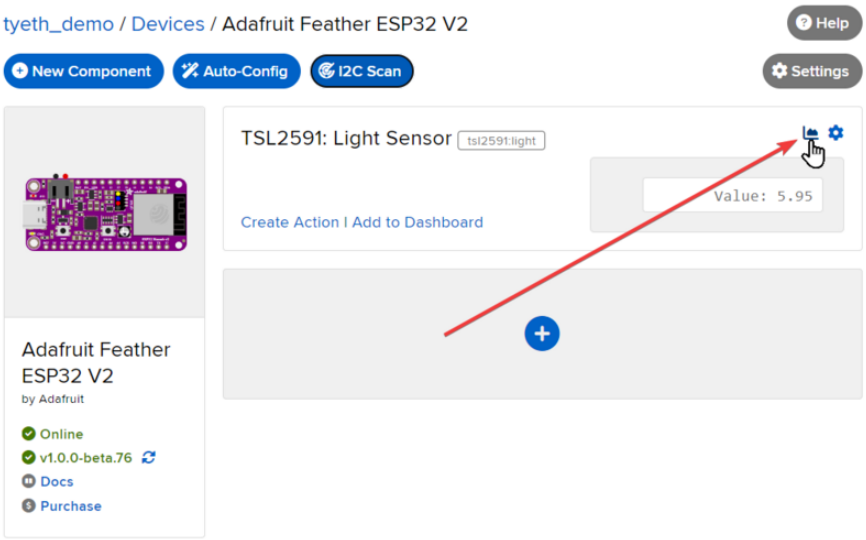


Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.

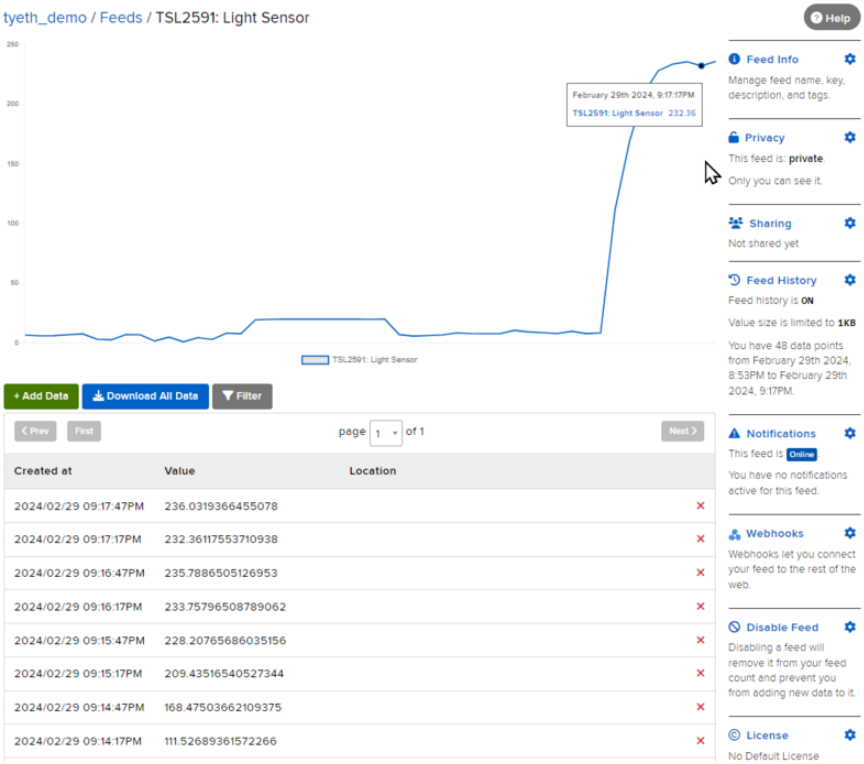


To view the data that has been logged from the sensor, click on the graph next to the sensor name.

Downloaded from Arrow.com.



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).



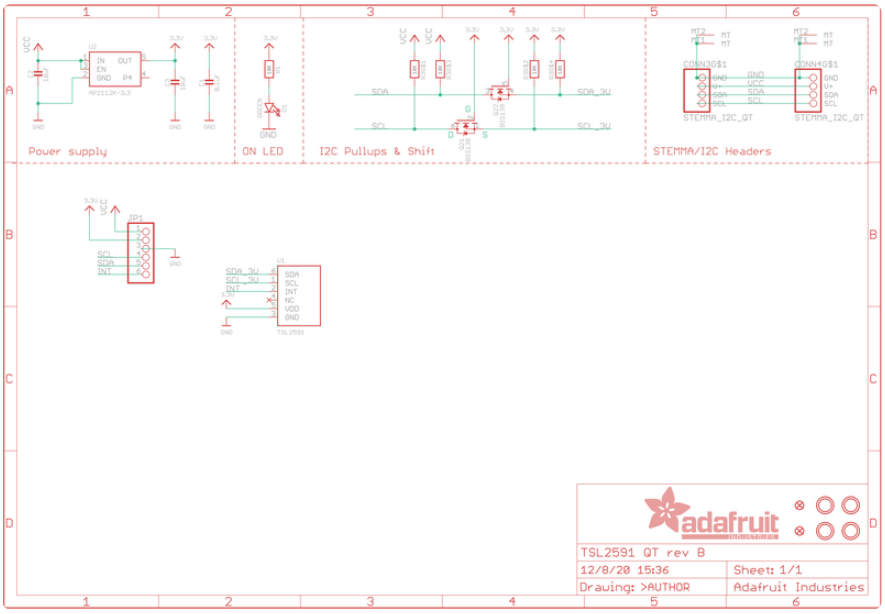
Downloads

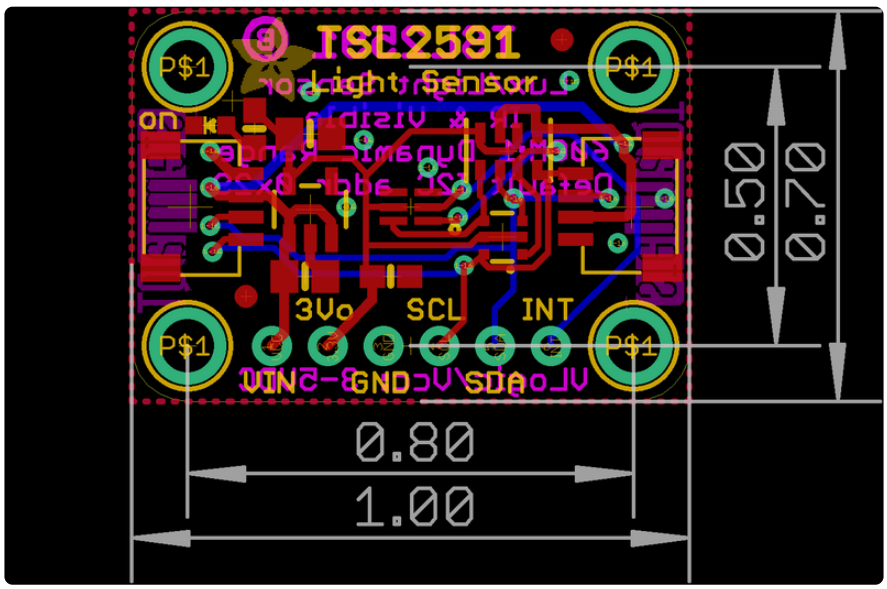
Datasheets & Files

- EagleCAD PCB files on GitHub (https://adafru.it/pOf)
- Fritzing object in Adafruit Fritzing library (https://adafru.it/NCK)

Downloaded from Arrow.com.

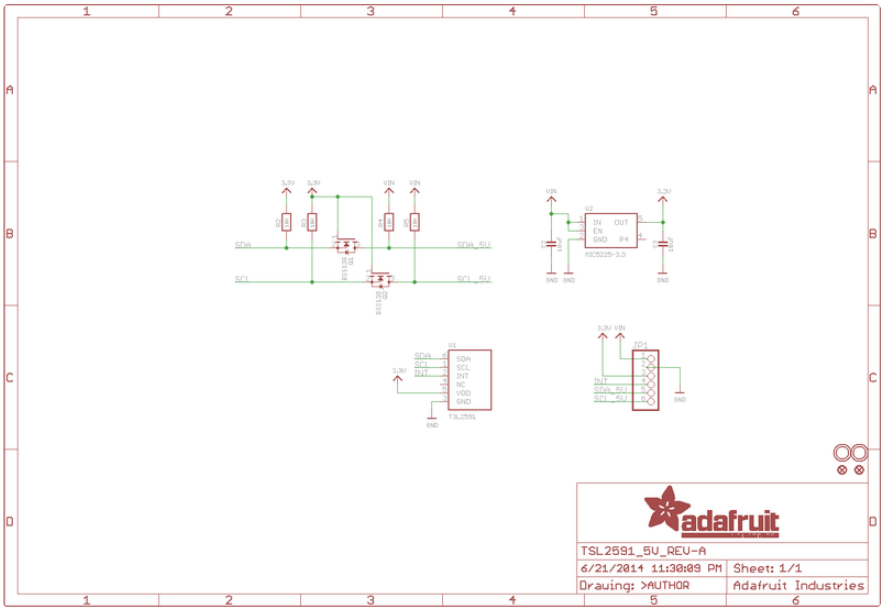
Schematic and Fab Print for STEMMA QT Version

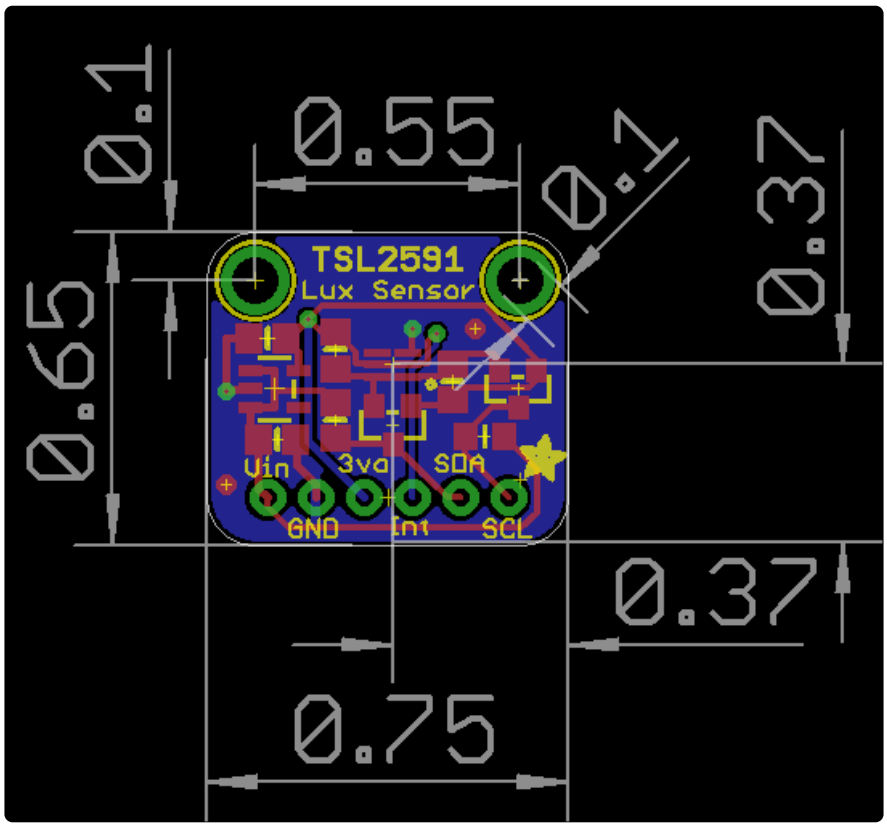




Downloaded from Arrow.com.

Schematic and Fab Print for Original Version





Downloaded from Arrow.com.