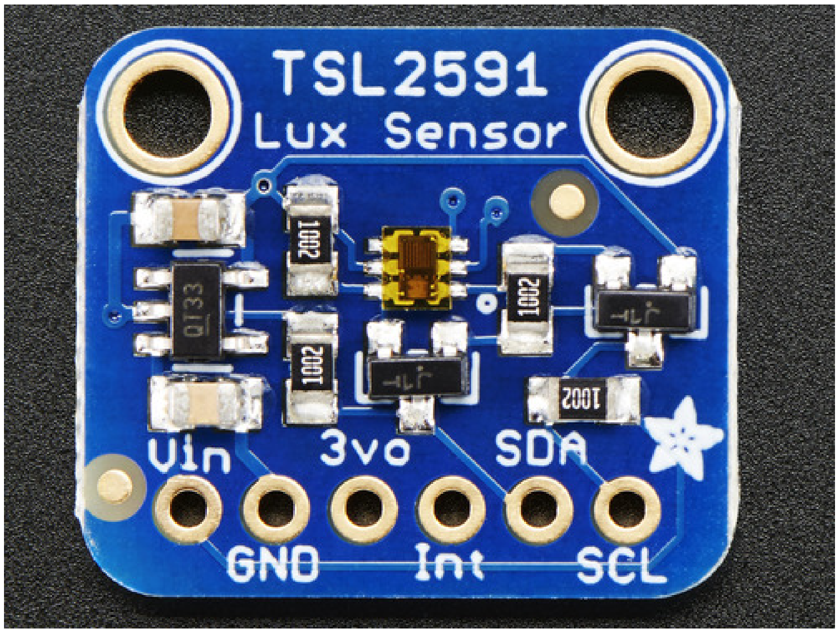
adafruit learning system

Adafruit TSL2591 High Dynamic Range Digital Light Sensor

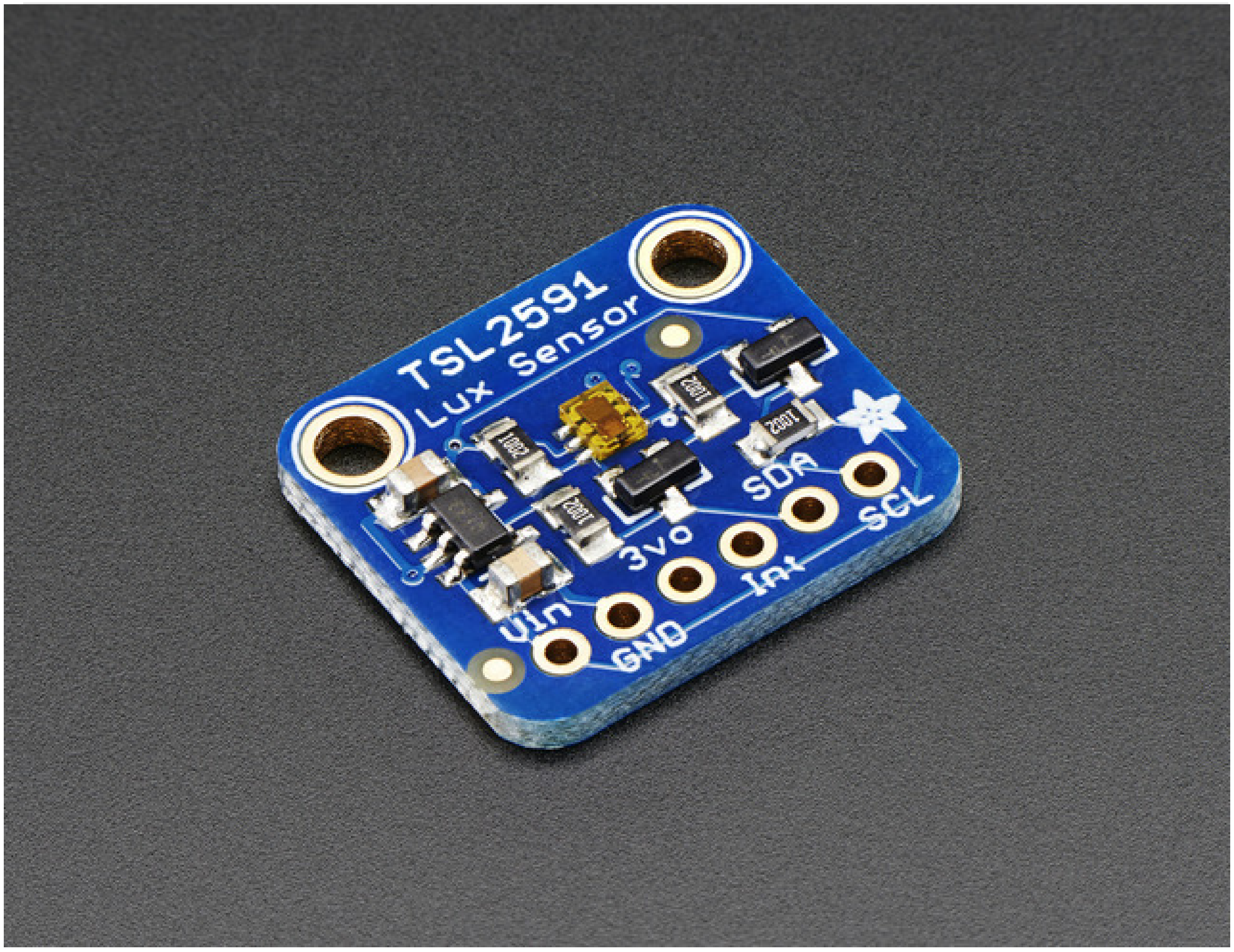
Created by lady ada



Last updated on 2014-07-11 02:15:07 PM EDT

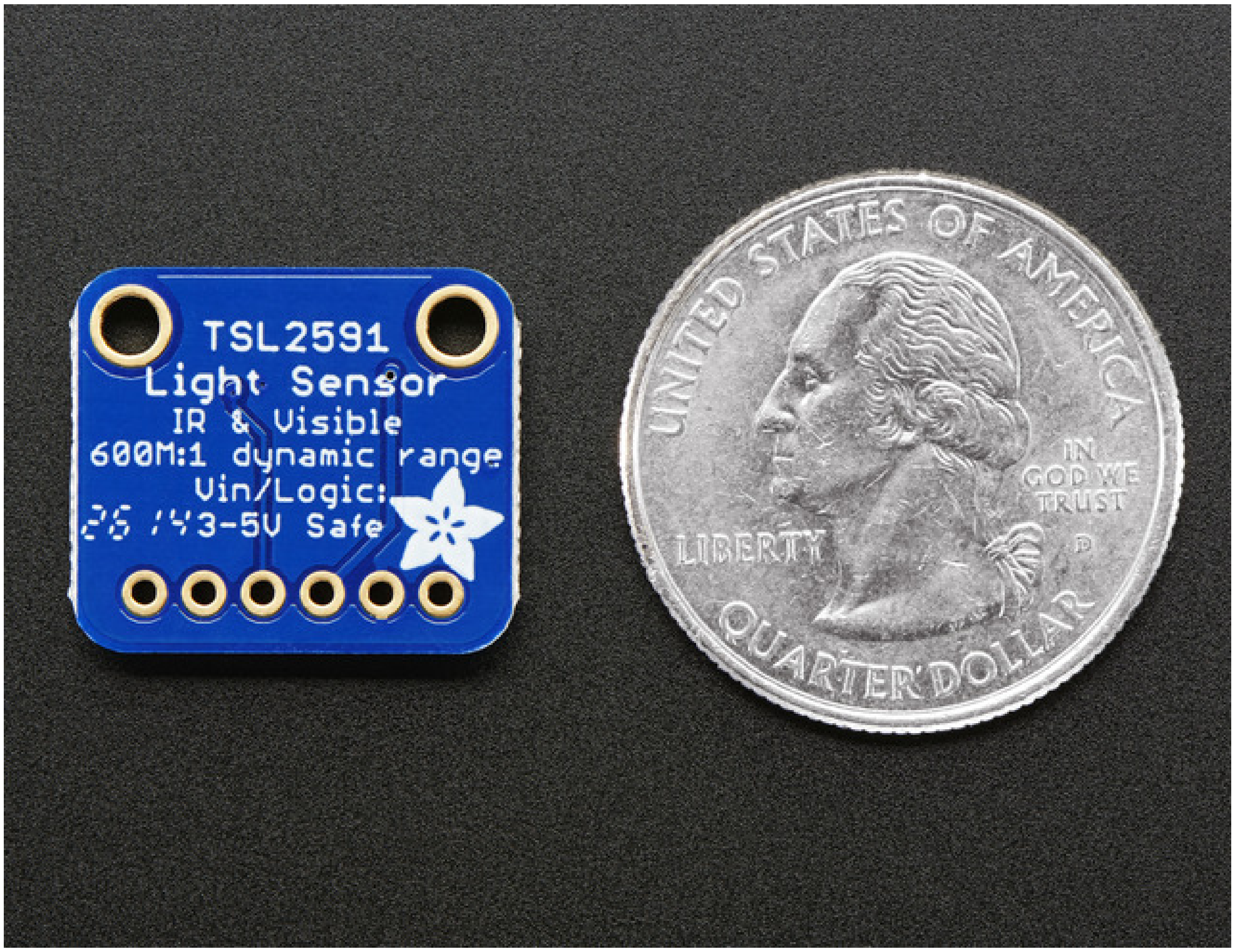
Guide Contents

Overview

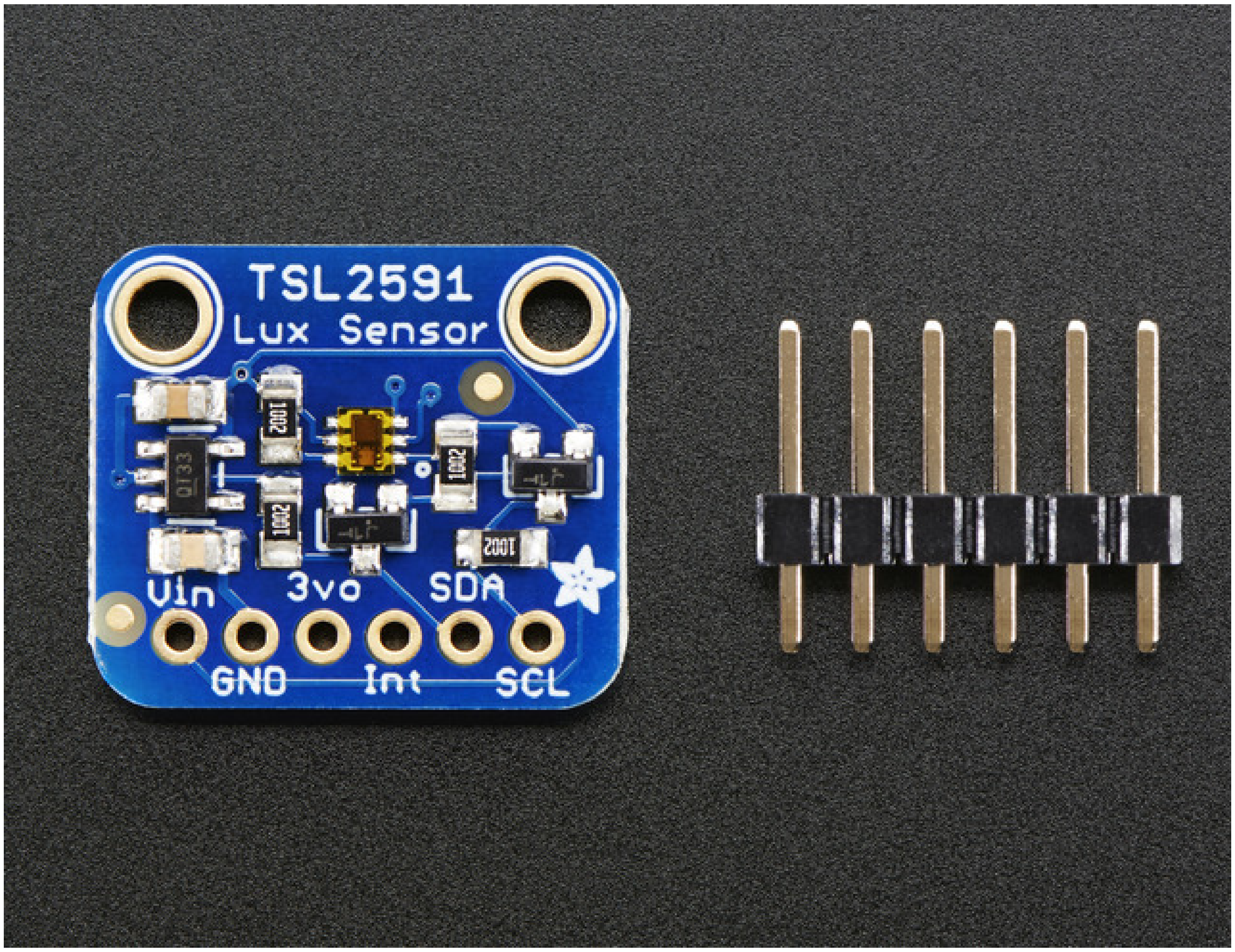


When the future is dazzlingly-bright, this ultra-high-range luminosity sensor will help you measure it. The TSL2591 luminosity sensor is an advanced digital light sensor, ideal for use in a wide range of light situations. Compared to low cost CdS cells, this sensor is more precise, allowing for exact 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)



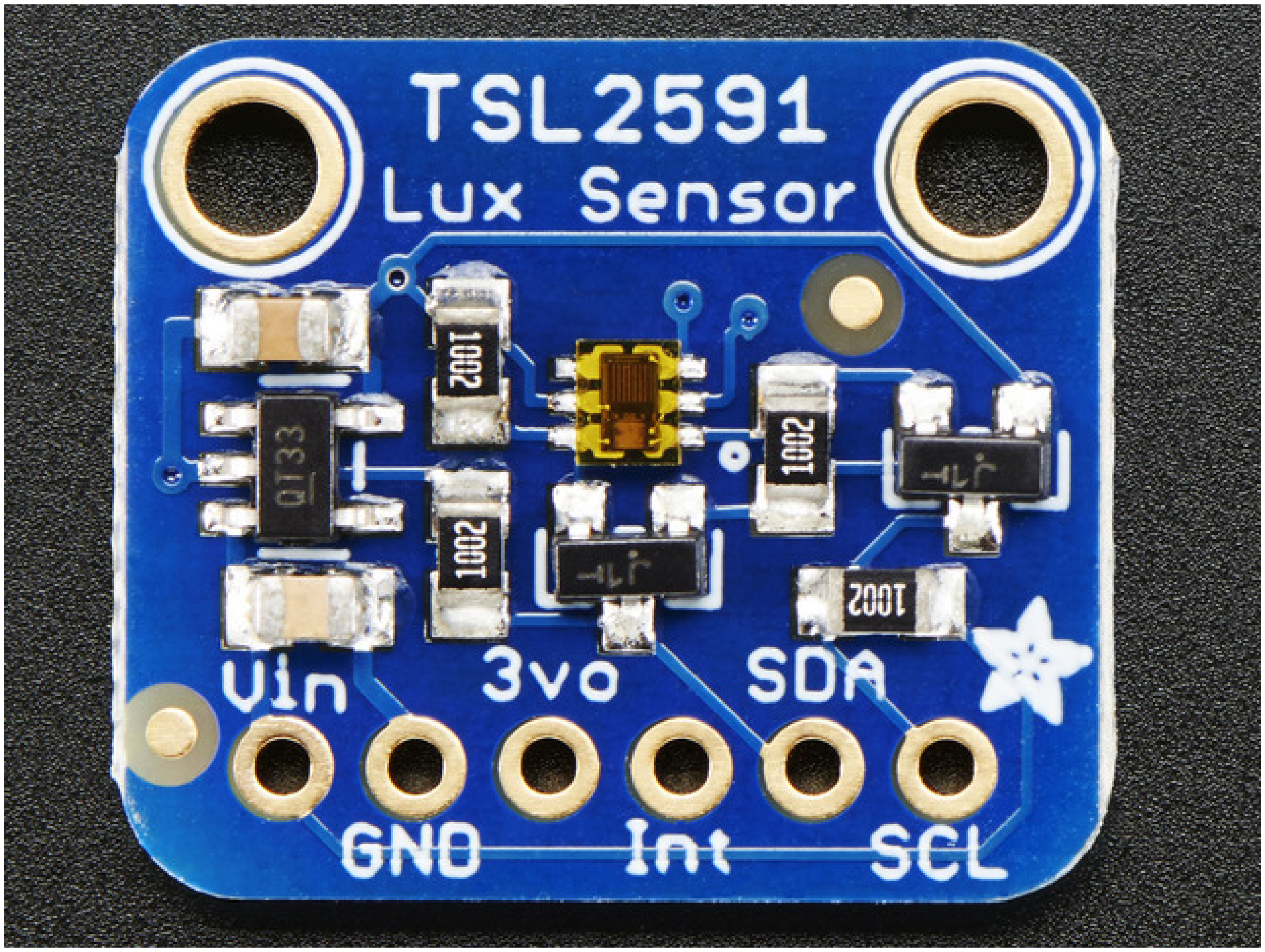
This sensor is much like the TSL2561 but with a wider range (and the interface code is different). This sensor has a massive 600,000,000:1 dynamic range! Unlike the TSL2561 you cannot change the I2C address either, so keep that in mind.



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.

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!



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

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

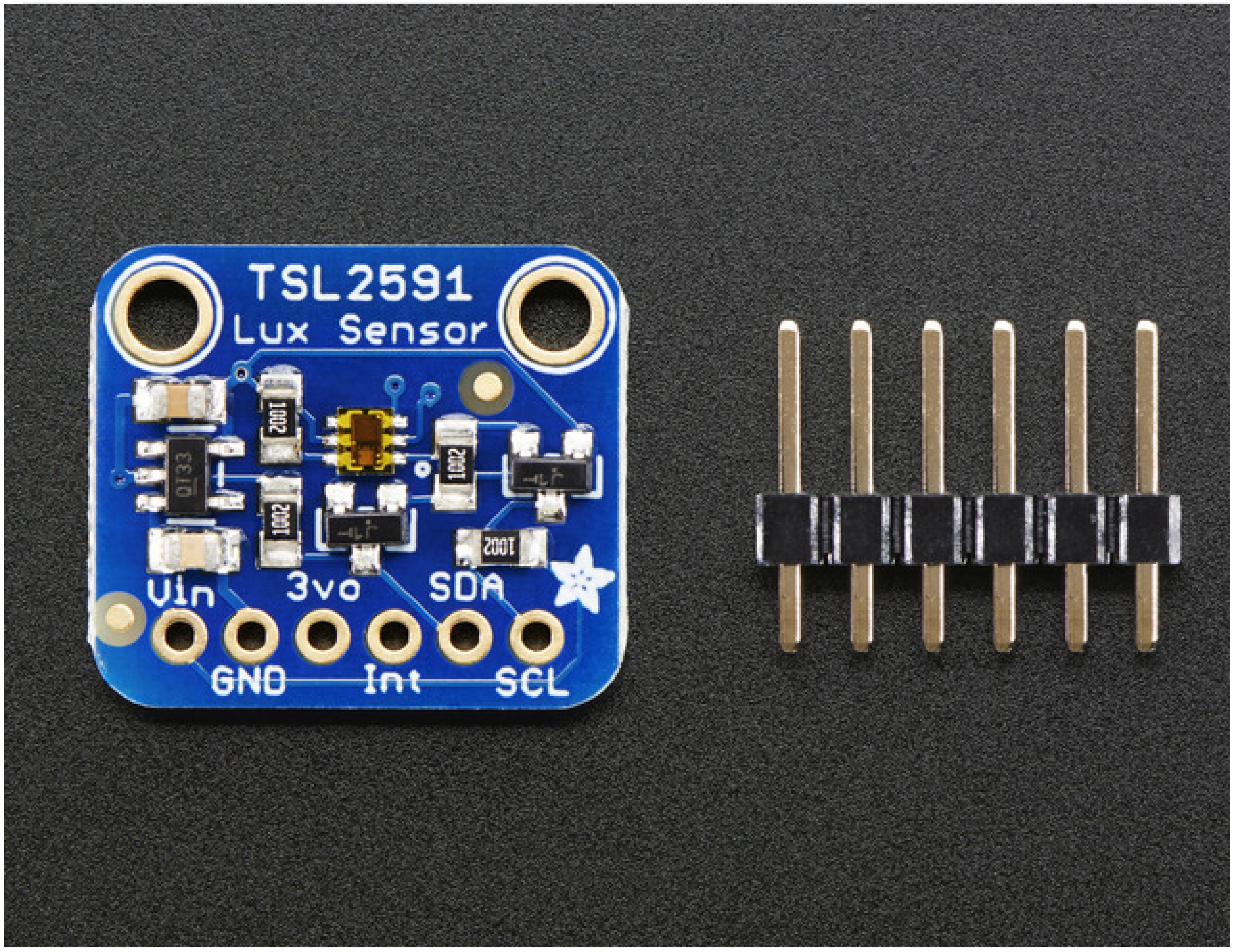
- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line.

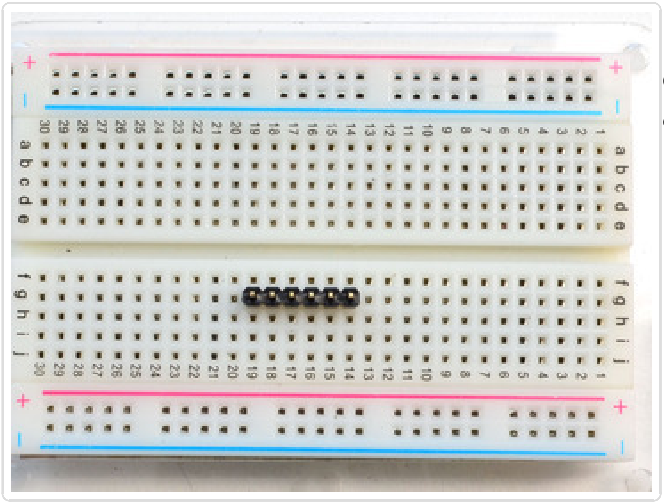
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line.

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

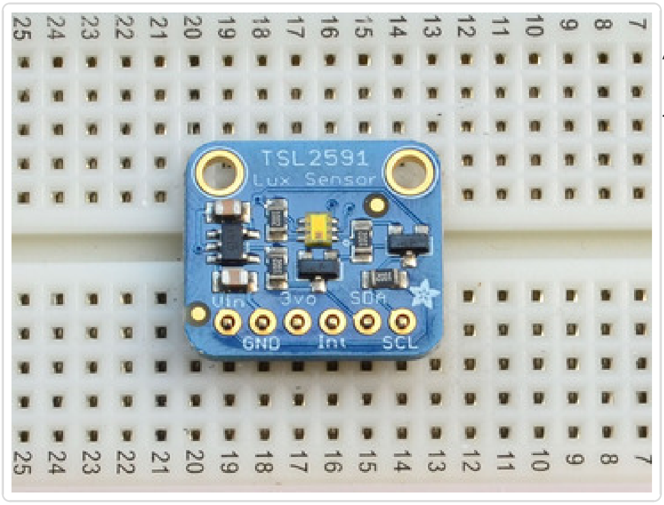
Assembly





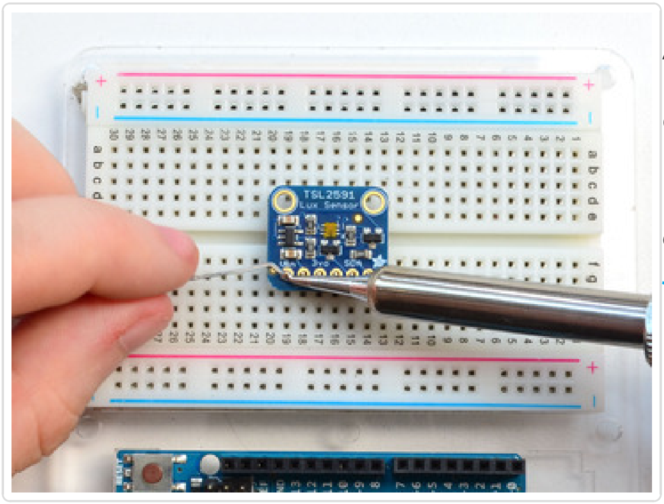
Prepare the header strip:

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



Add the breakout board:

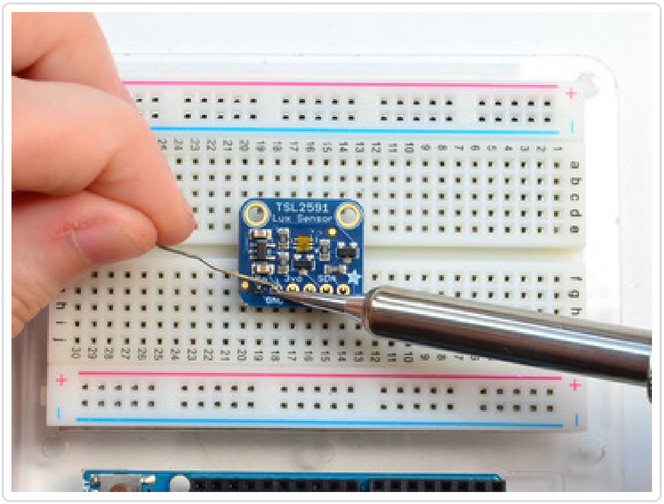
Place the breakout board over the pins so that the short pins poke through the breakout pads

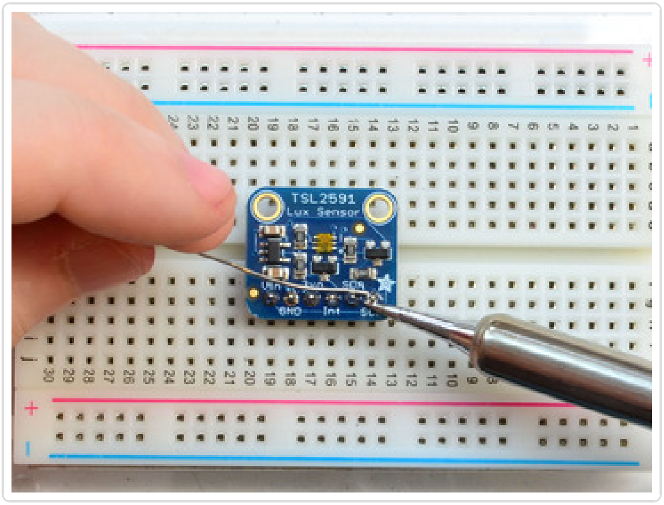


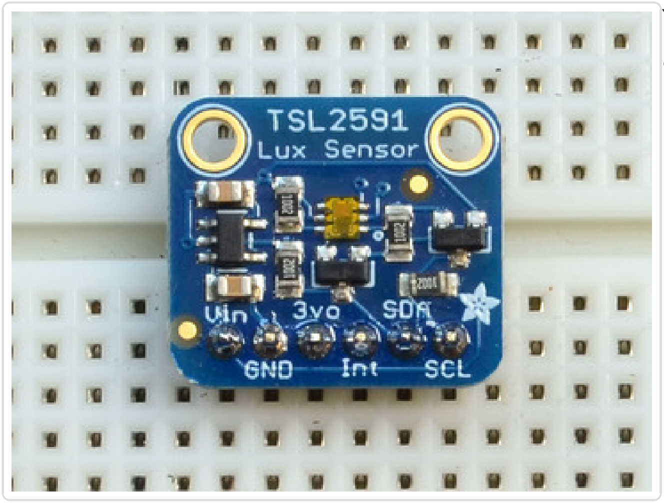
And Solder!

Be sure to solder all pins for reliable electrical contact.

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



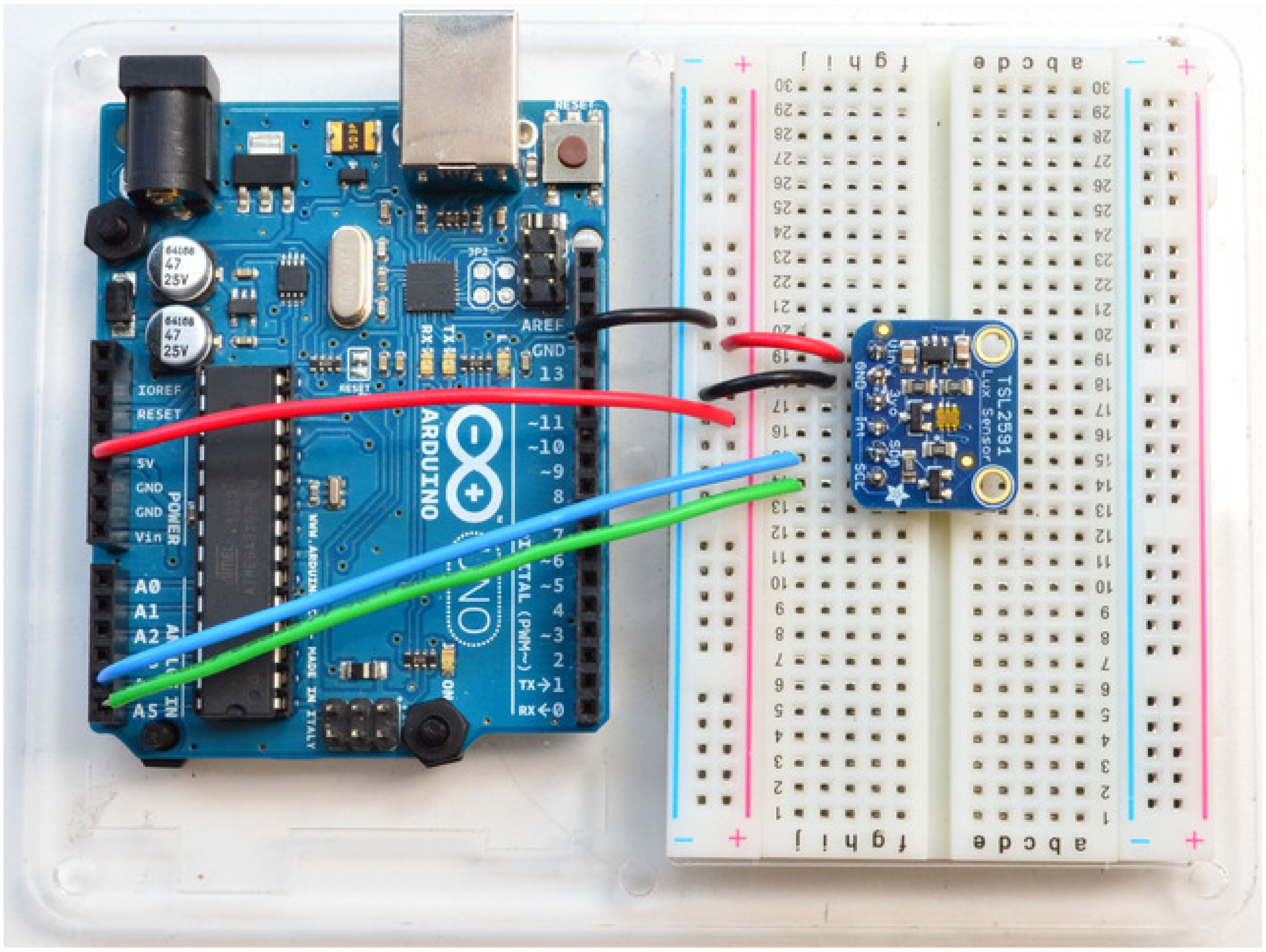




You're done! Check your solder joints visually and continue onto the next steps

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!



(http://adafru.it/dBn)

- Connect **Vin** to the power supply, 3-5V is fine. 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
- Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino. 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. 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 **0x29** and cannot be changed!

Download Adafruit_TSL2591

To begin reading sensor data, you will need to download Adafruit_TSL2591_Library from our github repository (http://adafru.it/dGz). You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

Download Adafruit TSL2591 Library

http://adafru.it/dGA

Rename the uncompressed folder **Adafruit_TSL2591** and check that the **Adafruit_TSL2591** folder contains **Adafruit_TSL2591.cpp** and **Adafruit_TSL2591.h**

Place the **Adafruit_TSL2591** library folder your **arduinosketchfolder/libraries/** folder. You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

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

Download Adafruit_Sensor

The TSL2591 library uses the Adafruit_Sensor support backend so that readings can be normalized between sensors. You can grab Adafruit_Sensor from the github repo (http://adafru.it/aZm) or just click the button below.

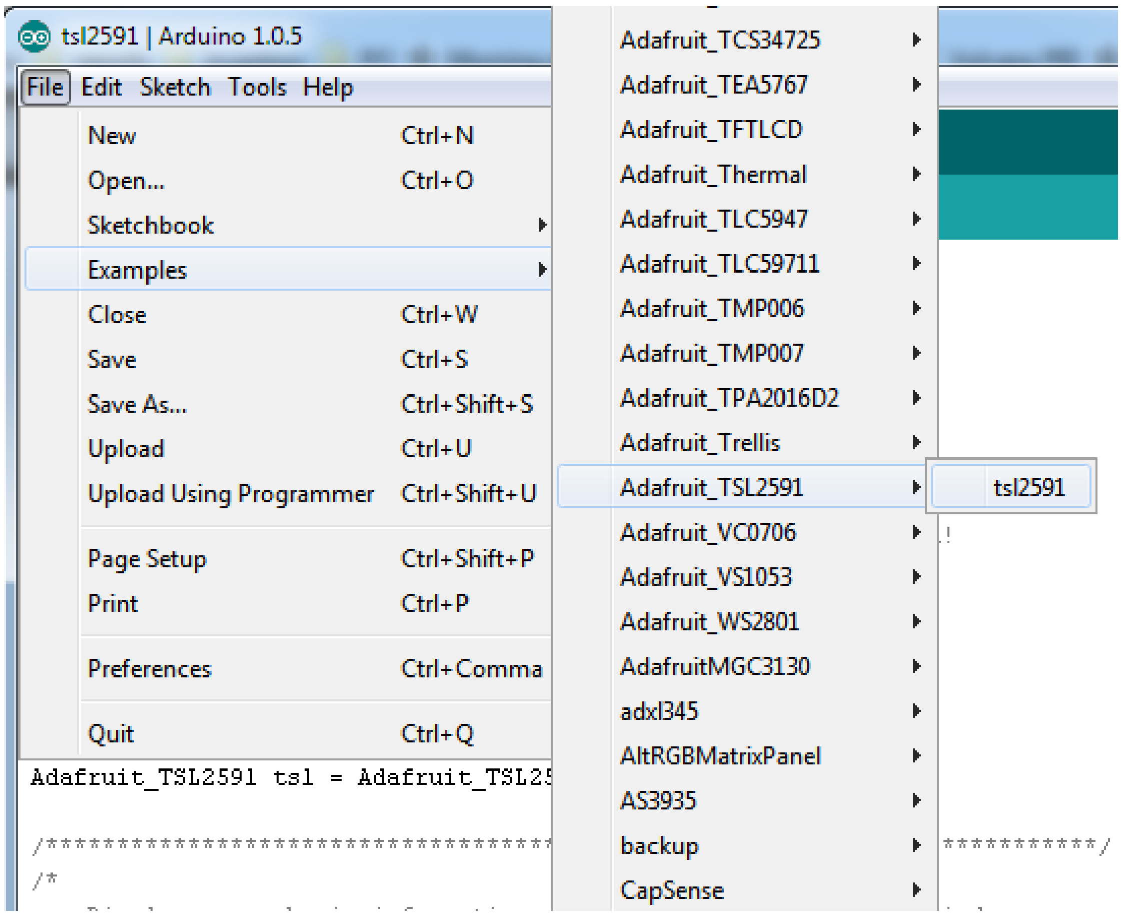
Download Adafruit_Sensor Library

http://adafru.it/cMO

Install like you did with Adafruit_TSL2591

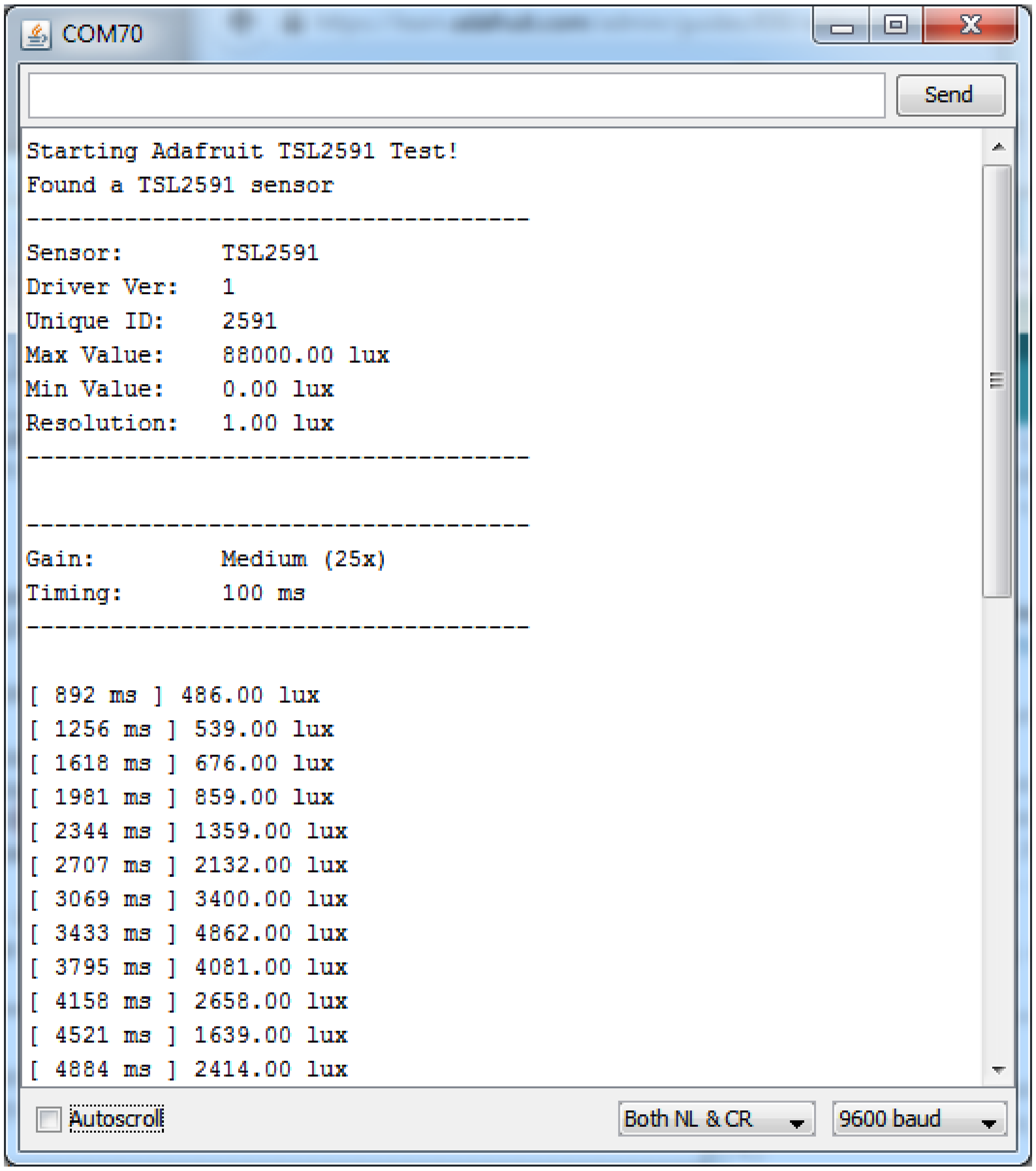
Load Demo

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



```
Adafruit_TSL2591 tsl = Adafruit_TSL25
```

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



```
Starting Adafruit TSL2591 Test!
Found a TSL2591 sensor
------------------------------------
Sensor:       TSL2591
Driver Ver:   1
Unique ID:    2591
Max Value:    88000.00 lux
Min Value:    0.00 lux
Resolution:   1.00 lux
------------------------------------


------------------------------------
Gain:         Medium (25x)
Timing:       100 ms
------------------------------------


[ 892 ms ] 486.00 lux
[ 1256 ms ] 539.00 lux
[ 1618 ms ] 676.00 lux
[ 1981 ms ] 859.00 lux
[ 2344 ms ] 1359.00 lux
[ 2707 ms ] 2132.00 lux
[ 3069 ms ] 3400.00 lux
[ 3433 ms ] 4862.00 lux
[ 3795 ms ] 4081.00 lux
[ 4158 ms ] 2658.00 lux
[ 4521 ms ] 1639.00 lux
[ 4884 ms ] 2414.00 lux
```

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**
- **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("");
}
```

Unified Sensor API

The Adafruit_TSL2591 library makes use of the Adafruit unified sensor framework (http://adafru.it/dGB) to provide sensor data in a standardized format and scale. If you wish to make use of this framwork, 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(&event);

  /* Display the results (light is measured in lux) */
  Serial.print("[ "); Serial.print(event.timestamp); Serial.print(" ms ] ");
  if ((event.light == 0) |
      (event.light > 4294966000.0) |
      (event.light <-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:

```
/**************************************************************************/
/*
```

```
    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(&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);
}
```

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 >> 16;
  full = lum & 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.

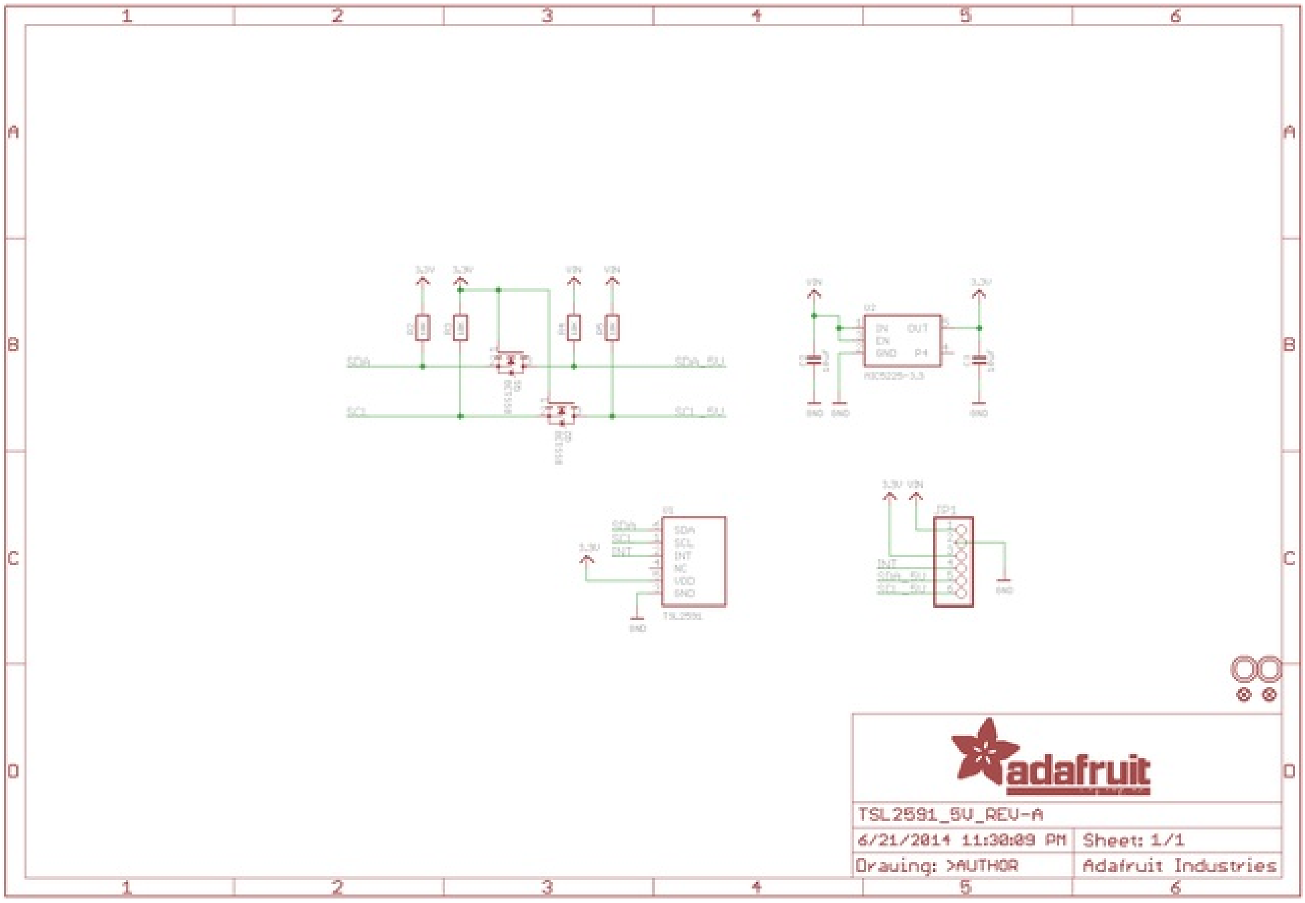
To calculate the lux, simple call **calculateLux(full, ir)**, where 'full' and 'ir' are raw 16-bit values taken from one of the two raw data functions above. See the code sample above for an example of calculating lux.

Downloads

Datasheets

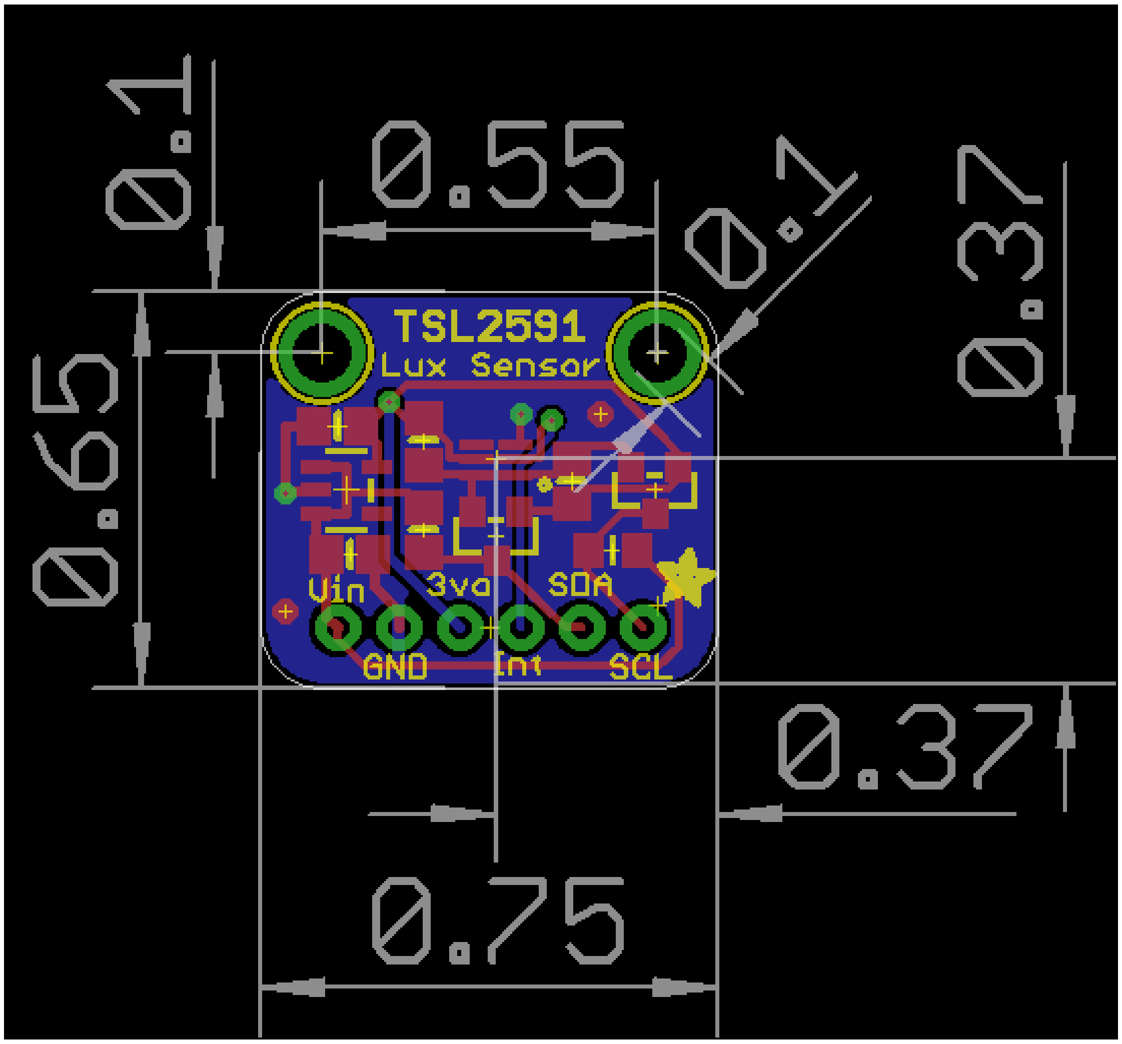
- TSL2591 Datasheet (http://adafru.it/dGs)

Schematic



Layout

(Dimensions are in Inches)


0.1
0.55
0.1
0.37
0.65
TSL2591
Lux Sensor
Vin
3vo
SDA
GND
Int
SCL
0.37
0.75