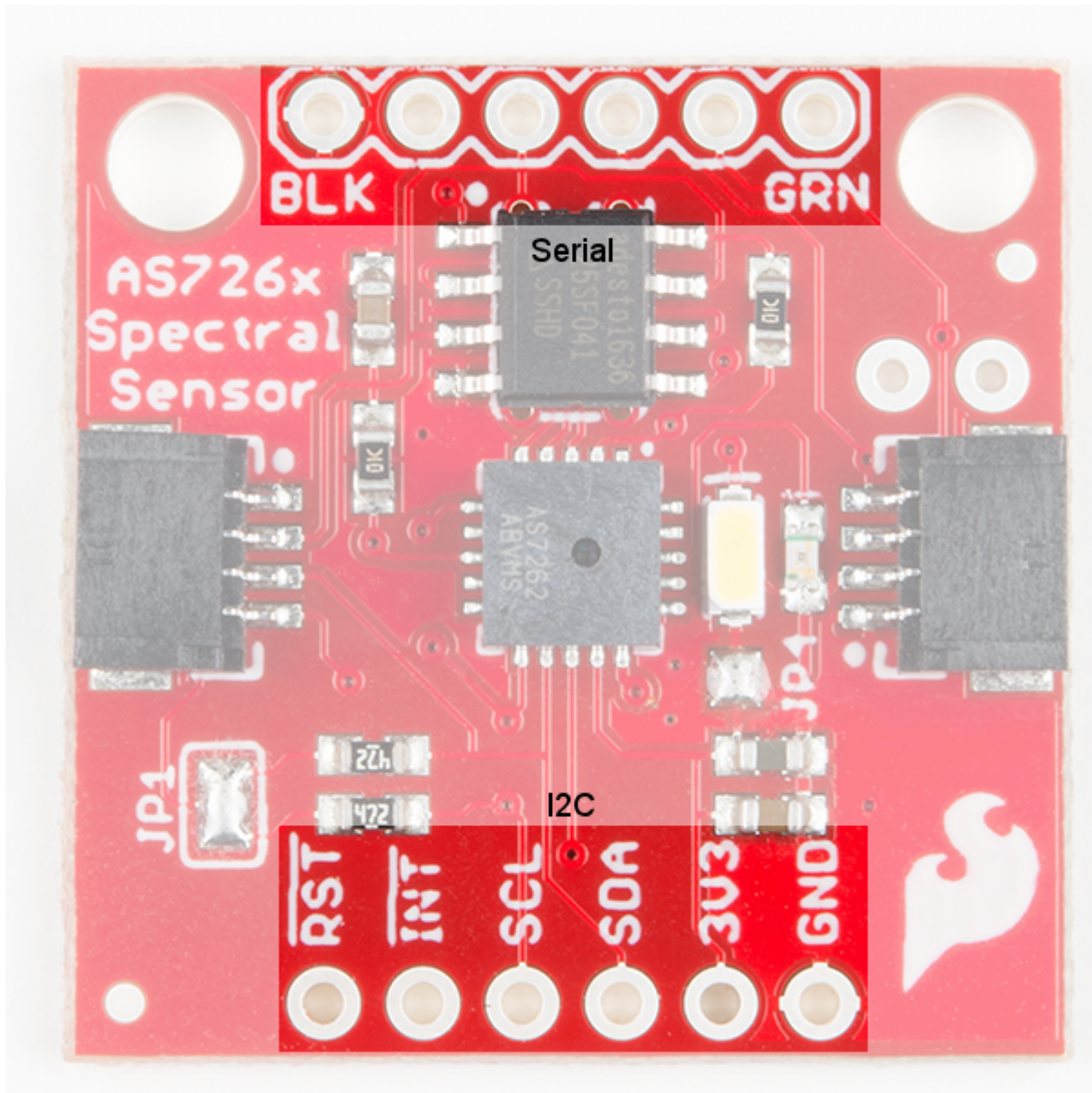


Hardware Overview

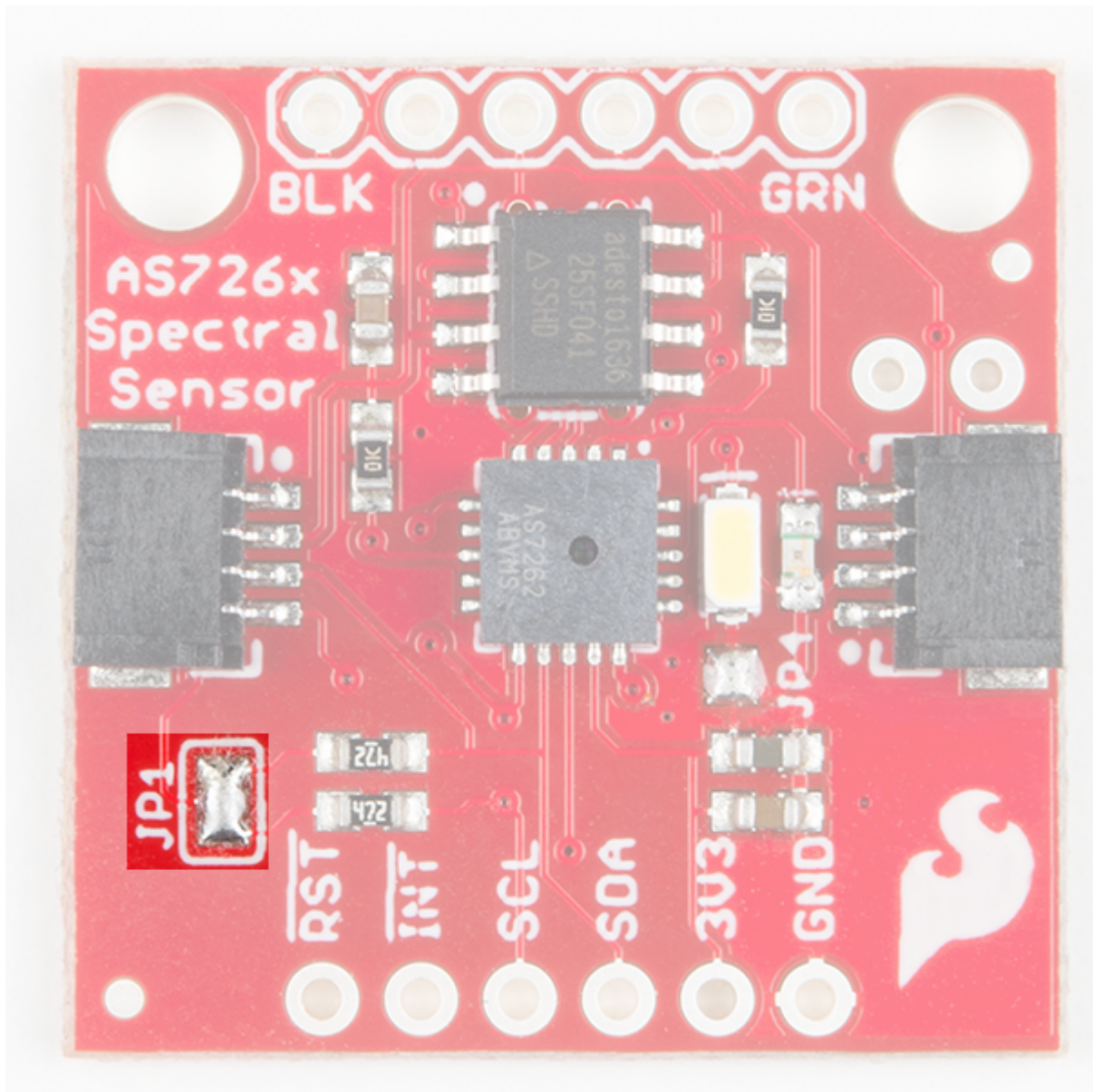
The AS7262 detects 450, 500, 550, 570, 600, and 650nm of light each with 40nm of full-width half-max detection. The AS7263 can detect 610, 680, 730, 760, 810, and 860nm of light each with 20nm of full-width half-max detection.

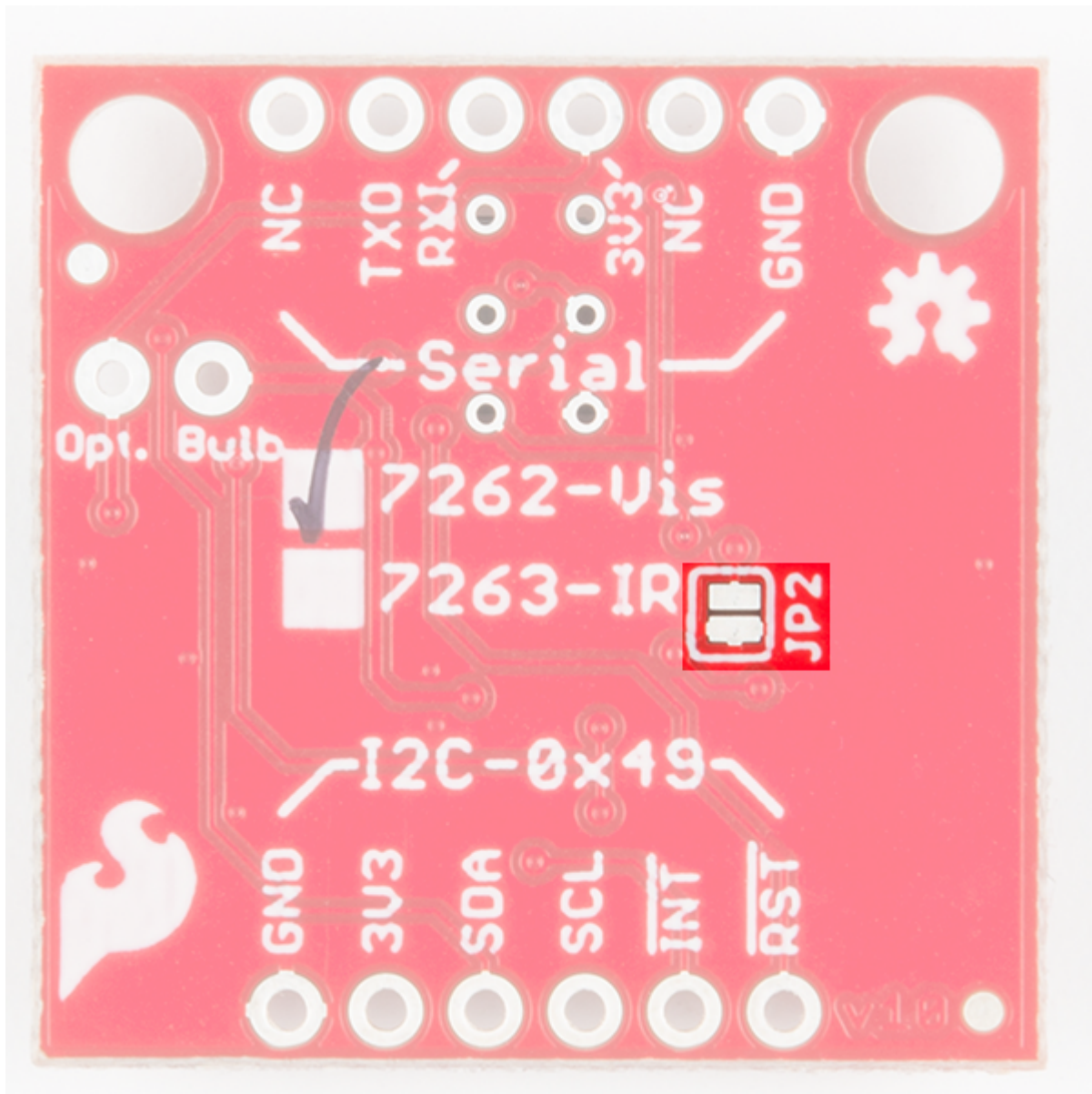
Communication

The AS726X is unique as it can communicate by both an I²C interface (through the onboard Qwiic connectors, or pins at the bottom of the board) and serial interface using AT commands (pins at the top of the board).



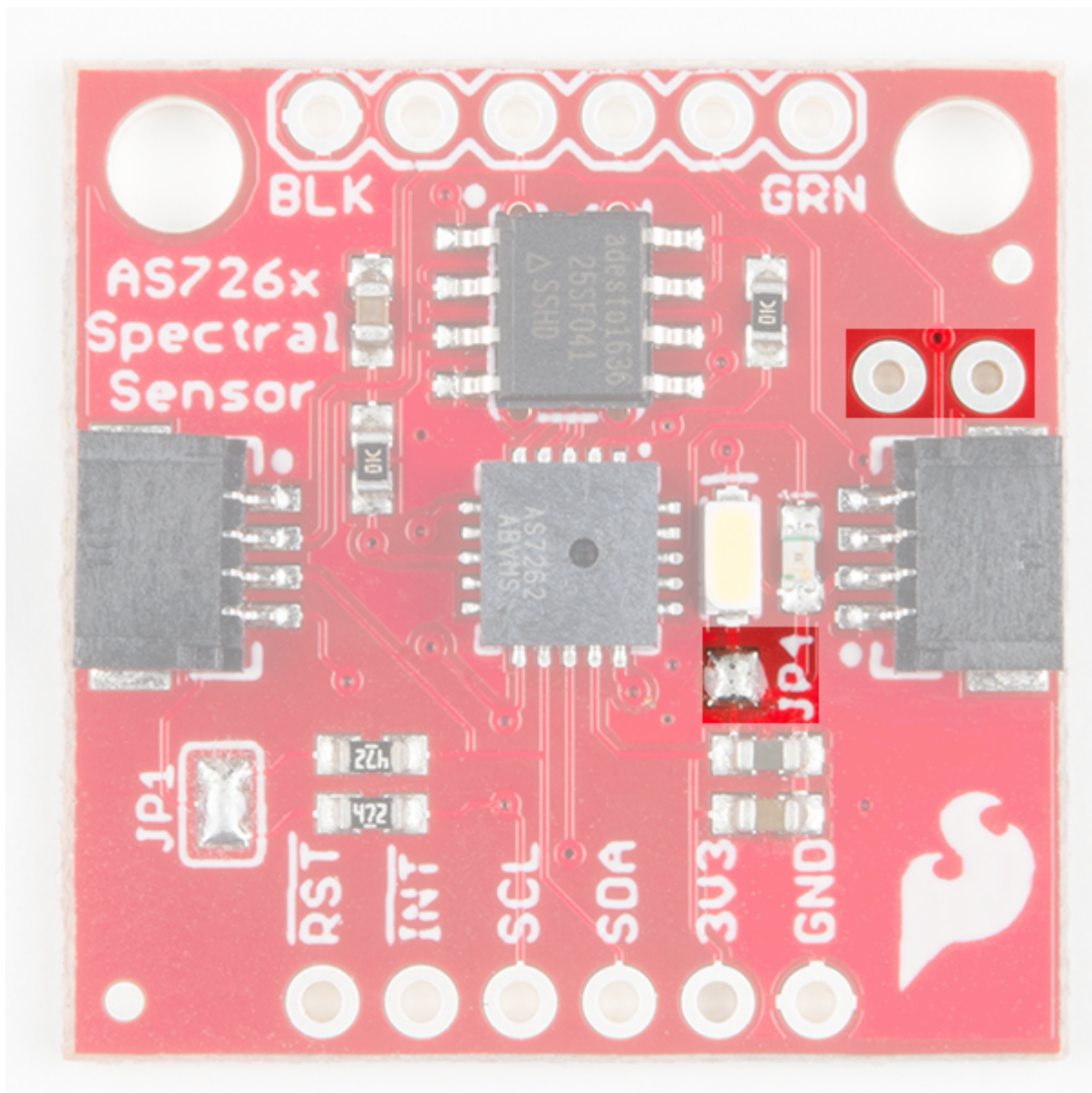
While I²C is the default setting (The default I²C address is 0X49 for both AS7262 and AS7263) Serial communication can be enabled by removing solder from the jumpers labeled JP1, adding solder to the jumper labeled JP2 (on the back of the board), and using Sparkfun's [USB-to-Serial](#) breakout to interface directly with the computer.





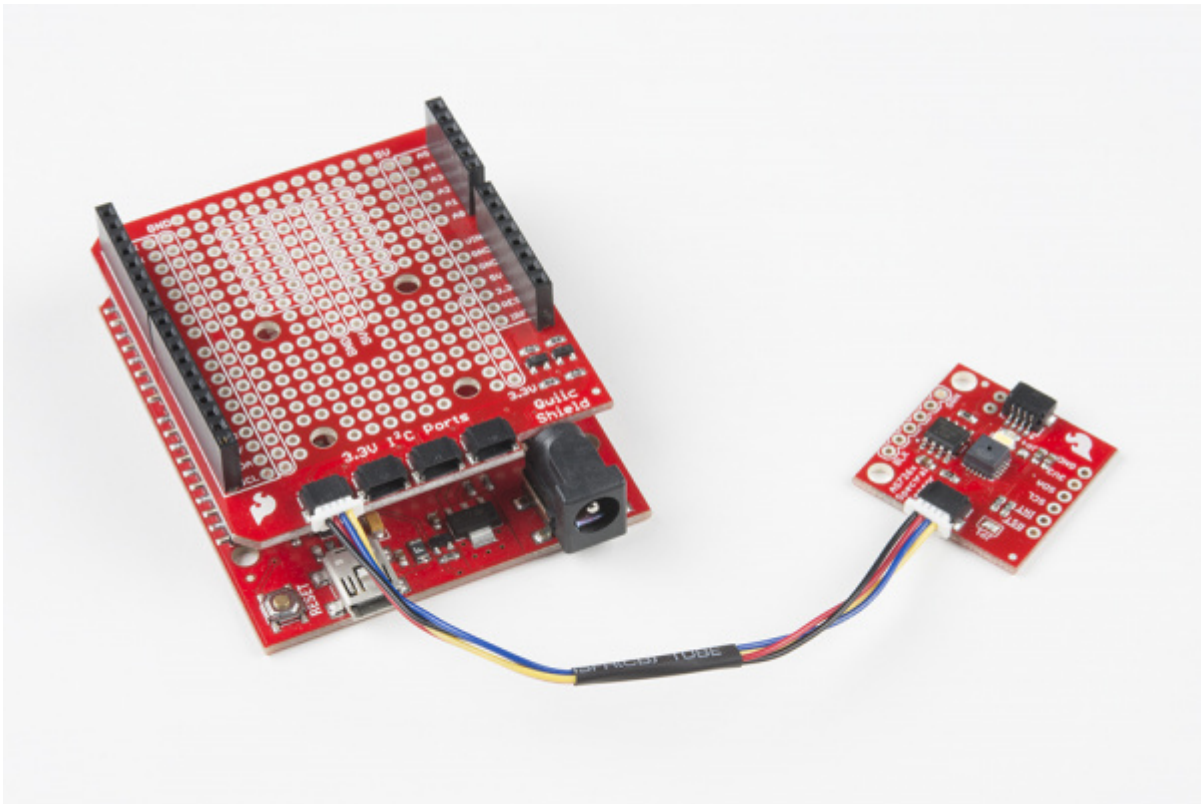
LED

The board also has multiple ways for you to illuminate the object that you are trying to measure for a more accurate spectroscopy reading. There is an onboard, 5700K LED that has been picked out specifically for this task. However, if you aren't satisfied with the onboard LED, you can grab your own through hole incandescent. While you should find a bulb rated for 3.3V, a bulb rated for higher voltage, like 5V, will still work, but will not run as bright as it normally would with 5V. We've found that [Mouser](#) is a good place to look for these. If you are going to go that route and use your own bulb, be sure to disable the onboard LED by removing the solder from the JP4 jumper.



Hardware Assembly

The beauty of Sparkfun's new Qwiic environment means that connecting the sensor could not be easier. Just plug one end of the Qwiic cable into the AS726X, the other into one of the Qwiic Shields, and stack the board on a development board. You'll be ready to upload a sketch to start taking spectroscopy measurements. It seems too easy, but that's why we made it this way! Here's an example using the RedBoard Programmed with Arduino.



Library Overview

Before we get started, we'll need to download and install SparkFun's AS726X Arduino library.

DOWNLOAD THE SPARKFUN AS726X LIBRARY

Note: This example assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on [installing the Arduino IDE](#). If you have not previously installed an Arduino library, please check out our [installation guide](#).

Before we get started developing a sketch, let's look at the available functions of the library.

- `void begin(TwoWire &wirePort, byte gain, byte measurementMode);` — Initializes the sensor with user given values for the wire port, gain, and measurement
- `void takeMeasurements();` — Sensor writes spectral measurements to memory locations.
- `void takeMeasurementsWithBulb();` — Illuminates onboard bulb, calls `takeMeasurements();`, then turns off the onboard bulb.
- `void printMeasurements();` — Fetches data from memory and outputs calibrated measurements using `Serial.print();`
- `void printUncalibratedMeasurements();` — Fetches data from memory and outputs uncalibrated measurements using `Serial.print();`
- `byte getTemperature();` — Fetches the temperature in degrees Celsius.
- `float getTemperatureF();` — Fetches the temperature in degrees Fahrenheit.
- `void setMeasurementMode(byte mode);` — Changes the measurement mode to 0, 1, 2, or 3
 - 0: Continuous reading of VBGY (Visible) / STUV (IR)
 - 1: Continuous reading of GYOR (Visible) / RTUX (IR)
 - 2: Continuous reading of all channels
 - 3: One-shot reading of all channels (power-on default)
- `boolean dataAvailable();` — Returns `true` or `false` based on whether or not data is available to be read.

- `void enableIndicator();` — Powers on the surface mounted blue indicator LED.
- `void disableIndicator();` — Powers off the surface mounted blue indicator LED.
- `void setIndicatorCurrent(byte current);` — Sets the current on the indicator LED. The default is `current = 3`, or 8 mA.
 - 0: 1 mA
 - 1: 2 mA
 - 2: 4 mA
 - 3: 8 mA
- `void enableBulb();` — Powers on the surface mounted blue indicator LED.
- `void disableBulb();` — Powers off the surface mounted blue indicator LED.
- `void setBulbCurrent(byte current);` — Sets the current limit on the indicator LED and optional bulb (They are connected in parallel) The default is `current = 0` or 12.5 mA.
 - 0: 12.5 mA
 - 1: 25 mA
 - 2: 50 mA
 - 3: 100 mA
- `void softReset();` — Gives the sensor a 1 second reset.
- `void setGain(byte gain);` — Pass in a 0, 1, 2 or 3 to change the gain.
 - 0: 1x
 - 1: 3.7x
 - 2: 16x
 - 3: 64x (power-on default)
- `void setIntegrationTime(byte integrationValue);` — This sets the time over which samples are taken.
 - Takes a value between 0 and 255.
 - Integration time will be $2.8 \text{ ms} * \text{integrationValue}$.
- `void enableInterrupt();` — Pulls the interrupt pin low. (Note: not yet implemented)
- `void disableInterrupt();` — Pulls the interrupt pin high.
- If you'd like access to just one channel, getting uncalibrated and calibrated spectral readings for the AS7262 (Visible) sensor can be accomplished with the following commands:
 - `int getViolet();`
 - `int getBlue();`
 - `int getGreen();`
 - `int getYellow();`
 - `int getOrange();`
 - `int getRed();`
 - `float getCalibratedViolet();`
 - `float getCalibratedBlue();`
 - `float getCalibratedGreen();`
 - `float getCalibratedYellow();`
 - `float getCalibratedOrange();`
 - `float getCalibratedRed();`
- A similar set of functions is available for accessing individual channels on the AS7263 (Near Infrared) sensor.
 - `int getR();`
 - `int getS();`
 - `int getT();`
 - `int getU();`
 - `int getV();`
 - `int getW();`
 - `float getCalibratedR();`
 - `float getCalibratedS();`
 - `float getCalibratedT();`
 - `float getCalibratedU();`
 - `float getCalibratedV();`
 - `float getCalibratedW();`

Software

Example 1 — Basic Readings

The below sketch will get you up and running taking calibrated spectral readings on all 6 channels. Once this sketch is uploaded, open the **serial monitor** with a baud rate of 115200 to display the spectral data from the sensor.

```
#include "AS726X.h"
AS726X sensor;

void setup() {
  sensor.begin();
}

void loop() {
  sensor.takeMeasurements();
  sensor.printMeasurements();//Prints out all measurements (calibrated)
}
```

[COPY CODE](#)

If we want, we can change the gain, measurement mode, and Wire that I²C uses by calling the `begin()` function with a few arguments. First, let's look at what values we can assign to which characteristic.

Example 2 — Sensor Settings

The below example code will initialize our sensor with a gain of 16x, measurement mode of 0, and our regular I²C port (you can run the sensor on a different I²C port if you have the right hardware, a Teensy perhaps?).

```
#include "AS726X.h"
AS726X sensor;

byte GAIN = 2;
byte MEASUREMENT_MODE = 0;

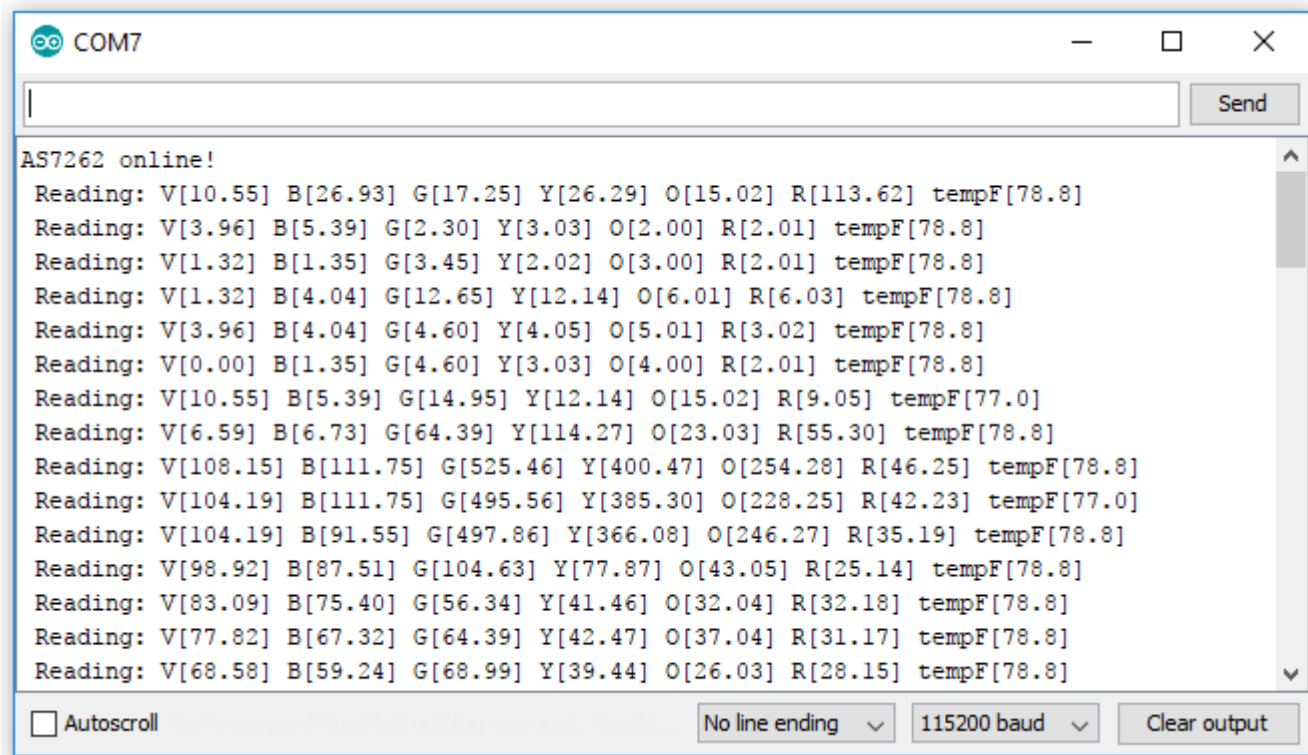
void setup() {
  sensor.begin(Wire, GAIN, MEASUREMENT_MODE);
}

void loop() {
  sensor.takeMeasurements();
  sensor.printMeasurements();
}
```

[COPY CODE](#)

Expected Output

Here is a picture of what to expect when bringing your AS726X online for either of the above sketches. (The program will print "AS7263 online!" instead if you have that sensor)



The screenshot shows a serial terminal window titled "COM7". At the top, there is a text input field and a "Send" button. The main area of the window displays the following text:

```
AS7262 online!  
Reading: V[10.55] B[26.93] G[17.25] Y[26.29] O[15.02] R[113.62] tempF[78.8]  
Reading: V[3.96] B[5.39] G[2.30] Y[3.03] O[2.00] R[2.01] tempF[78.8]  
Reading: V[1.32] B[1.35] G[3.45] Y[2.02] O[3.00] R[2.01] tempF[78.8]  
Reading: V[1.32] B[4.04] G[12.65] Y[12.14] O[6.01] R[6.03] tempF[78.8]  
Reading: V[3.96] B[4.04] G[4.60] Y[4.05] O[5.01] R[3.02] tempF[78.8]  
Reading: V[0.00] B[1.35] G[4.60] Y[3.03] O[4.00] R[2.01] tempF[78.8]  
Reading: V[10.55] B[5.39] G[14.95] Y[12.14] O[15.02] R[9.05] tempF[77.0]  
Reading: V[6.59] B[6.73] G[64.39] Y[114.27] O[23.03] R[55.30] tempF[78.8]  
Reading: V[108.15] B[111.75] G[525.46] Y[400.47] O[254.28] R[46.25] tempF[78.8]  
Reading: V[104.19] B[111.75] G[495.56] Y[385.30] O[228.25] R[42.23] tempF[77.0]  
Reading: V[104.19] B[91.55] G[497.86] Y[366.08] O[246.27] R[35.19] tempF[78.8]  
Reading: V[98.92] B[87.51] G[104.63] Y[77.87] O[43.05] R[25.14] tempF[78.8]  
Reading: V[83.09] B[75.40] G[56.34] Y[41.46] O[32.04] R[32.18] tempF[78.8]  
Reading: V[77.82] B[67.32] G[64.39] Y[42.47] O[37.04] R[31.17] tempF[78.8]  
Reading: V[68.58] B[59.24] G[68.99] Y[39.44] O[26.03] R[28.15] tempF[78.8]
```

At the bottom of the window, there is a control bar with the following elements from left to right:

- An unchecked checkbox labeled "Autoscroll".
- A dropdown menu set to "No line ending".
- A dropdown menu set to "115200 baud".
- A "Clear output" button.