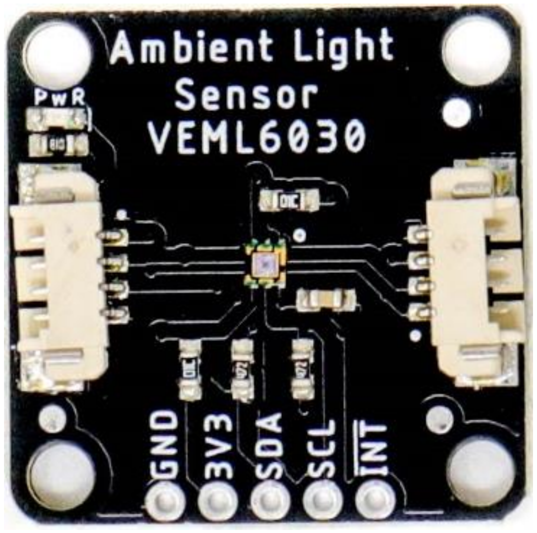
SmartElex Ambient Light Sensor - VEML6030

The Ambient Light Sensor (VEML6030) is an I²C enabled ambient light sensor with high sensitivity and high accuracy. It reads ambient light in Lux and boasts a number of nice features including: the ability to set high and low thresholds for an optional interrupt, power saving features that enable single digit micro-amp current draw, and a readable range from zero to 120,000 Lux. We've also written an Arduino library that gives full access to all features and includes example code demonstrating all its' abilities. Follow along and let's learn about all its features and how to use them!



Hardware Overview

Power

You can provide **3.3V** through the connector on the board or through the `3V3` labeled pin on the through hole header. When you correctly power the board, the on-board red power LED will turn on.

LED

There is one red LED on the product that will turn on when power is supplied to the board. You can disconnect this LED by cutting the jumper on the underside of the product labeled `LED`, see **Jumpers** below.

Connector or I²C Pins

There are two connectors on the board to easily connect to the sensor via I²C. Another option is to solder directly to the I²C plated through holes on the side of the board.

Interrupt Pin

A nice feature on the Ambient Light sensor is its ability to set
both **LOW** and **HIGH** thresholds that triggers an interrupt on the product. For example, we can know when the light in a room falls below a certain amount and conversely when the light comes back on! You don't have to settle for just the hardware interrupt though. We also have a software solution, check out the interrupt example code below.

Jumpers

There are three jumpers on the underside of this board. Starting in the lower left is a triple jumper labeled `I2C` that connects pull-up resistors to the I²C data lines. If you're daisy chaining many I²C devices together than you may need to consider cutting these traces.

To the right of that is the address jumper labeled `ADR` that allows you to select the Ambient Light Sensor's other I²C address: **0x10**. By default your Ambient Light Sensor is shipped with the I²C address **0x48**.
Finally the jumper in the upper right is the `LED` jumper which can be cut to disconnect the on board power LED.

Gain and Integration Time Settings

What does **gain** and **integration time** mean? You can think of gain as an electronic mechanism to amplify a weak signal. If you're in a dark room with very little light, the sensor needs a way to amplify the weak light source to get accurate lux calculations and the *gain* is what give is it's **oomph**. Likewise, **Integration time** is the amount of time the sensitive photo diodes within the sensor absorb light before beginning its' Lux calculation. Both of these together are necessary for sensing in a dark room, but what if it was really bright outside?! Now there's so much light in the envinronment that you actually want to scale down the electronic's response to that saturation so you need a *lower* gain. And since there's so much light, the photo diodes don't need to be exposed long to get what they need to make the calculations, so a *lower integration* time is also needed.

The Ambient Light Sensor can detect ranges of light in Lux from zero to 120,796! That's a gigantic range from dark to direct sun in the middle of the day. To accomplish this you have to set the **Gain** and **Integration Time** settings. This is trivial with the Arduino Library we've written and we'll walk you through it below in **Example1 Ambient Light Basics**. With each setting gives you a range of light that you can read. Check out the table below to see what's capable at each possible setting. Notice that slower integration and higher gain gives you a smallest range of (0->236) but the highest resolution (0.0036 lux/bit).

The datasheet recommends that you use a setting of 1/4 (.25) or 1/8 (.125) unless the Ambient Light Sensor is going to sit behind dark glass. This will help to prevent over saturation of the photodiodes within the sensor.

Maximum Light Detection Range: Lux

Integration Time (milliseconds)	GAIN 2	GAIN 1	GAIN 1/4	GAIN 1/8
800	236	472	1887	3775
400	472	944	3775	7550
200	944	1887	7550	15099
100	1887	3775	15099	30199
50	3775	7550	30199	60398
25	7550	15099	60398	120796

Resolution: Lux/Bit

Integration Time (milliseconds)	GAIN 2	GAIN 1	GAIN 1/4	GAIN 1/8

800	0.0036	0.0072	0.0288	0.0576
400	0.0072	0.0144	0.0576	0.1152
200	0.0144	0.0288	0.1152	0.2304
100	0.0288	0.0576	0.2304	0.4608
50	0.0576	0.1152	0.4608	0.9216
25	0.1152	0.2304	0.9216	1.8432

Power Save Modes

Another cool feature of the Ambient Light Sensor is its ability to run at *extremely* low currents. Power save modes should be used when you're continuously reading ambient light data. For example, if you're going to gather ambient light data every second, why not use a power save mode and save battery life? There are four power save modes that can be enabled with integration times of 100ms and above. Below is a table showing the power save mode, the current draw, and it's refresh rate. Check out Example 4 in the Arduino Library to see how to set it up and use the table below as a reference.

Note the **Refresh Time** is the time needed for a new reading to be ready. Make sure there is a delay in your code of at least this length between readings to ensure you're getting new data.

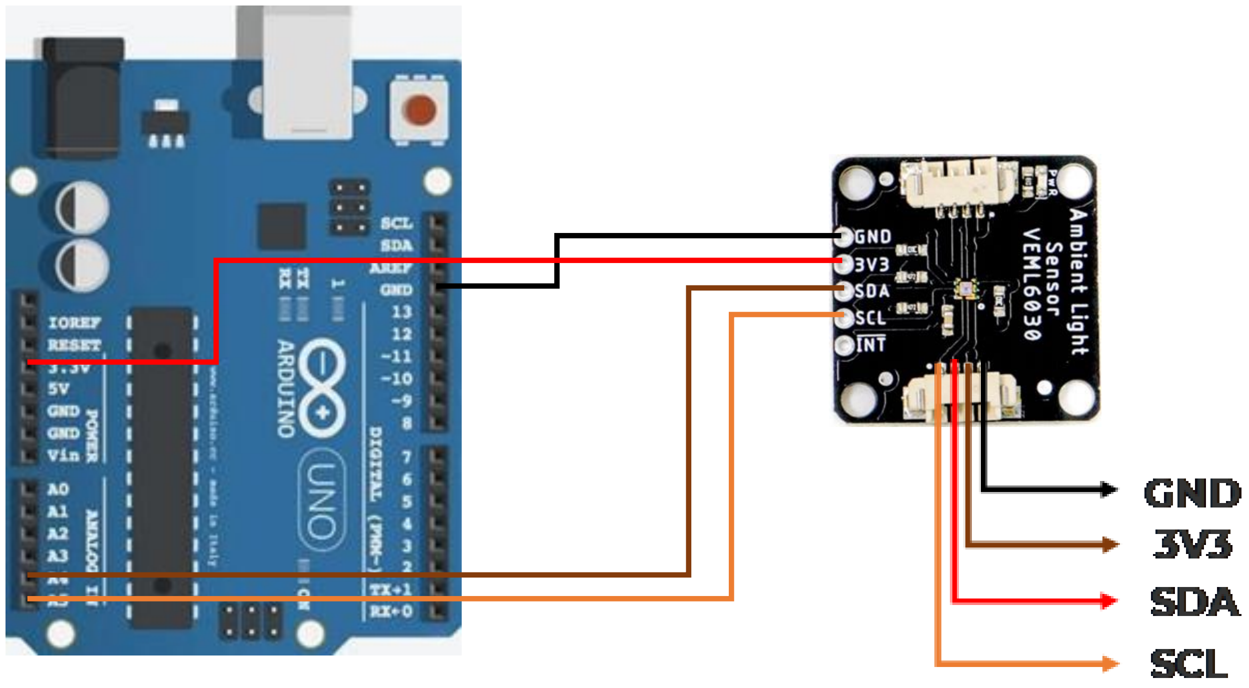
Integration Time	Power Save Mode	Refresh Time (milliseconds)	Current Draw (microamperes)	Resolution (lx/bit)
100	1	600	8	0.0288
100	2	1100	5	0.0288

100	3	2100	3	0.0288
100	4	4100	2	0.0288
200	1	700	13	0.0144
200	2	1200	8	0.0144
200	3	2200	5	0.0144
200	4	4200	3	0.0144
400	1	900	20	0.0072
400	2	1400	13	0.0072
400	3	2400	8	0.0072
400	4	4400	5	0.0072
800	1	1300	28	0.0036
800	2	1800	20	0.0036
800	3	2800	13	0.0036
800	4	4800	8	0.0036

Rounding Errors?!

Later on when you read back an interrupt threshold, you may notice that the interrupt lux values are off by one in some cases. This is because of the inherent rounding error with the Ambient Light Sensor. The final Lux value is calculated by multiplying the value of the bits that represent the ambient light by a decimal number (0.2304 for example). This decimal number is rounded to a whole number (e.g. 19.97 becomes 19) because the sensor does not care about fractions of a Lux. I chose to read back the rounded number because that's the interpreted value of the ambient light sensor and what is stored in its' registers.

Wiring:



Arduino	VEML6030
A5(SCL)	SCL
A4(SDA)	SDA
3.3V	3V3
GND	GND

Arduino Library

Sparkfun has written a library to make it even easier to get started with the Ambient Light Sensor. The library will give you the full functionality of the sensor and provides example code to get the most out of your project. You can obtain these libraries through the Arduino Library Manager by searching **SparkFun Ambient Light Sensor**.

Example Code:

```
#include <Wire.h>

#include "SparkFun_Qwiic_Scale_NAU7802_Arduino_Library.h" // Click here to get the
library: http://librarymanager/All#SparkFun_NAU7802

NAU7802 myScale; //Create instance of the NAU7802 class

void setup()
{
  Serial.begin(115200);
  Serial.println("Qwiic Scale Example");

  Wire.begin();

  if (myScale.begin() == false)
  {
    Serial.println("Scale not detected. Please check wiring. Freezing...");
    while (1);
  }
  Serial.println("Scale detected!");
}

void loop()
{
  if(myScale.available() == true)
  {
    int32_t currentReading = myScale.getReading();
    Serial.print("Reading: ");
    Serial.println(currentReading);
  }
}
////////////////////////////////////////END//////////////////////////////////////////
```

Example 1: Ambient Light Basics

In this first example, we'll get you comfortable with gathering ambient light and setting two vital properties of the sensor's ability to read light: the **gain** and the **integration time**. These two properties determine the resolution (accuracy) of the reading and the available ranges of light that you can read! For example, a gain of 1/8 and 800ms integration time cannot read anything above 3775 Lux. This means you'll max out your sensor outdoors but would be a proper setting for dim rooms due to it's higher resolution.

At the top of the example we have three variables *gain*, *time*, and *luxval*. The first two hold the value for the gain and integration time settings mentioned above. **Gain** settings can be: `2`, `1`, `1/4`, and `1/8`; typically 1/4 gain will capture everything you need with good resolution. Possible **integration times** can be `800`, `400`, `200`, `100`, `50`, `25` and by default the sensor is set to `100`; times are in milliseconds. Check the Gain and Integration Time table above under **Hardware Overview** to see maximum illumination capabilities and resolution for every setting.
If you have any doubt with which settings to pick, just keep the example code's default settings: a gain of **1/4 (0.125)** and an integration time of **100ms**. This will give you a range of up to 15,000 Lux with a decent resolution!

In the setup, we call `light.begin()` to check if we can communicate with the SparkFun Ambient Light Sensor. Next, we call the `light.setGain()` and `light.setIntegTime()` functions giving them the variables holding the **gain** and **time** values above. Next we'll read back those values to make sure that they were set correctly. That's it! We're now set to read some light!
One thing to keep in mind is that you need to set a `delay()` in between readings.
The **Integration Time** is the amount of time that the sensor uses to fill its' sensitive components with light. If you set an integration time of 100ms then make sure you're delay is at least that long. A longer integration time will need a longer delay.