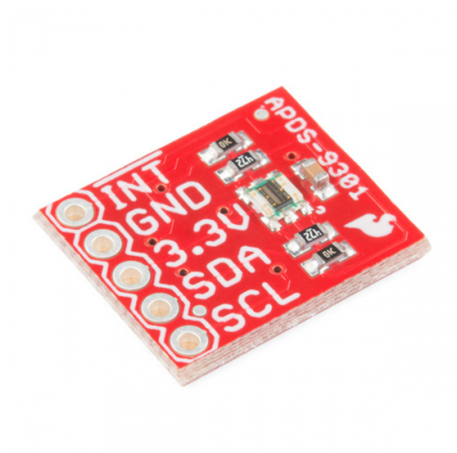
APDS-9301 Sensor Hookup Guide

Introduction

The APDS-9301 is an I²C compatible luminosity sensor which returns readings in lux. It is non-instantaneous, requiring some integration time to take a measurement. SparkFun provides a library making use of the part very simple.



SparkFun Ambient Light Sensor Breakout - APDS-9301

● SEN-14350

Required Materials

Please check the wish list below for items required to follow this tutorial.

APDS-9301 Hookup Guide SparkFun Wish List

	SparkFun Ambient Light Sensor Breakout - APDS-9301 SEN-14350
	SparkFun FTDI Basic Breakout - 3.3V DEV-09873 This is the newest revision of our [FTDI Basic](http://www.sparkfun.co…
	SparkFun USB Mini-B Cable - 6 Foot CAB-11301 This is a USB 2.0 type A to Mini-B 5-pin cable. You know, the mini-B…
	Female Header Pack

PRT-11269
This pack of female headers includes two 6-pin headers and two 8-pi…

Arduino Pro 328 - 3.3V/8MHz
DEV-10914
It's blue! It's skinny! It's the Arduino Pro! SparkFun's minimal design a…

Breadboard - Mini Modular (Red)
PRT-12044
This red Mini Breadboard is a great way to prototype your small proje…

Jumper Wires - Connected 6" (M/F, 20 pack)
PRT-12794
These are 6" long jumper wires terminated as male to female. Use th…

Break Away Male Headers - Right Angle
PRT-00553
A row of right angle male headers - break to fit. 40 pins that can be cu…

Tools

No special tools are required to follow this tutorial. You will need a soldering iron, solder, and general soldering accessories.

Soldering Iron - 30W (US, 110V)
TOL-09507

Solder Lead Free - 15-gram Tube
TOL-09163

Suggested Reading

We suggest reviewing the tutorials below to ensure that you're up-to-date with all of the skills necessary to follow this hookup guide.

How to Solder: Through-Hole Soldering
This tutorial covers everything you need to know about through-hole soldering.

Installing an Arduino Library
How do I install a custom Arduino library? It's easy!

Logic Levels

I2C

Learn the difference between 3.3V and 5V devices and logic levels.

An introduction to I2C, one of the main embedded communications protocols in use today.

Hardware Overview

The APDS-9301 breakout board is fairly simple, with only a few ancillary passive components in addition to the ADPS-9301 sensor IC itself.

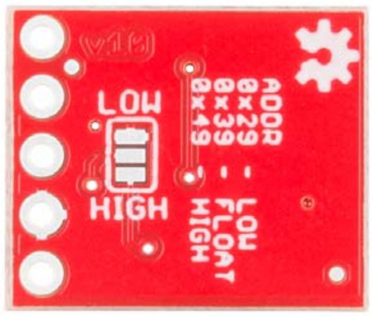
APDS-9301 sensor - This is the sensor IC. Its operating voltage only extends up to 3.6V, so to use it with a 5V Arduino or Arduino clone, you'll need some kind of voltage translation!

I²C pullup resistors - The board includes pullup resistor so you don't need to add them externally.

INT pin - The APDS-9301 can be programmed to generate an interrupt under certain conditions. This pin will be asserted low (i.e., pulled to ground) when those conditions are met. Note that this is an open collector pin, so you'll need to enable the pullup resistor on the processor for it to work.

SparkFun standard I²C header - Most boards which can be communicated to via I²C use this pinout, making it easy to stack them or connect them in a daisy chain.

Address Select Jumpers - On the back of the board, the only item of interest is the address select jumper. By default, this jumper is open, resulting in an I²C address of **0x39**. If the HIGH side of the jumper is closed, the address will be **0x29**. If the LOW side of the jumper is closed, the address will be **0x49**. If a big old blob of solder closes both sides of the jumper, it will still work, and the address will be 0x49.



Library Overview

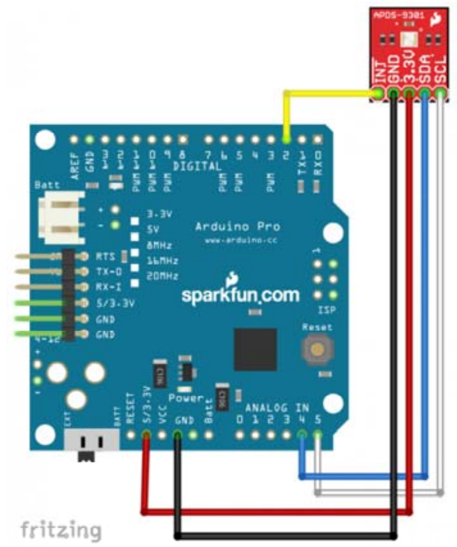
Here’s a list of the most critical functions supported by the library.

- `begin(address)` - enables the IC, sets the gain and integration times to minimum (i.e., lowest sensitivity), and disables the interrupt.
- `setGain(gainLevel)` - there are two possible parameters to pass to this function: `APDS9301::LOW_GAIN` and `APDS9301::HIGH_GAIN`. High gain is 16x more sensitive than low gain.
- `gain getGain()` - returns one of the two values specified in `setGain()` above. This function actually reads the gain from the sensor and returns the true value currently being used.
- `setIntegrationTime(integrationTime)` - there are three possible parameters to pass to this function: `APDS9301::INT_TIME_13_7_MS`, `APDS9301::INT_TIME_101_MS`, and `APDS9301::INT_TIME_402_MS`. Sensitivity to light increases with integration time, and the rate at which new data is generated by the sensor is also determined by integration time. By default, the integration time is set to the lowest value (13.7ms).
- `intTime getIntegrationTime()` - returns one of the three values specfied in `setIntegrationTime()` above. This function actually reads the gain from the sensor and returns the true value currently being used.
- `enableInterrupt(intMode)` - pass either `APDS9301::INT_ON` or `APDS9301::INT_OFF` to this function to enable or disable the interrupt functionality. By default, the interrupt is disabled. If enabled, the following three functions will determine the circumstances under which an interrupt will be issued and the INT pin will be set high.
- `setCyclesForInterrupt(cycles)` - sets number of ADC cycles values must be in interrupt range for an interrupt to occur. Pass `0` to interrupt on every ADC cycle (ADC cycle time is defined by integration time as discussed above). Pass `1` to interrupt if the reading is ever above the high threshold or below the low threshold (see next two functions for information regarding threshold settings). Pass `2` through `15` to require that many cycles above or below the respective threshold before issuing an interrupt.
- `setLowThreshold(threshold)` - pass an `unsigned int` to this function between `0` and `65535`. Readings on CH0 of the sensor (which detects both visible and IR light) below this threshold for the time set by the `setCyclesForInterrupt()` will trigger an interrupt on the INT pin. To disable the low threshold, simply write `0` to this function.
- `unsigned int getLowThreshold()` - returns the current low threshold setting, read from the sensor directly.
- `setHighThreshold(threshold)` - pass an `unsigned int` to this function between `0` and `65535`. Readings on CH0 of the sensor (which detects both visible and IR light) above this threshold for the time set by the `setCyclesForInterrupt()` will trigger an interrupt on the INT pin. To disable the high threshold, simply write `65535` to this function.
- `unsigned int getHighThreshold()` - returns the current low threshold setting, read from the sensor directly.
- `float readLuxLevel()` - returns a floating point number representing the current light level in lux. Note that due to inherent inaccuracy in the sensor, this value is only accurate to within 35% -40% of the actual absolute lux value.

Example

Hardware Hookup

We use a hookup as pictured below for our example project. However, this basic Arduino code should work for any number of different Arduino compatible boards. In this case, we show it on an Arduino Pro 3.3V, to allow the setup to work without any level translation between the Arduino and the APDS-9301.



Note that this setup requires some soldering. New to soldering? Check out our through hole soldering guide!

Notes on Operation

The APDS-9301 has two internal light sensing elements: CH0, which responds to infrared and visible light, and CH1, which responds to only visible light. By combining these two channels, the sensor is able to compensate for local infrared light and provide a more accurate estimate of the current lux value.

The calculation of lux reading from CHO and CH1 sensor output readings is not straightforward. The function is piecewise, with different coefficients depending on the ratio CH1/CH0. This function is automatically implemented in the `readLuxLevel()` library function, but can be found in the datasheet if you're curious.

The APDS-9301 sensor works by multiplying the light input signal by a gain (either 1x or 16x) and then using an integrating ADC to measure the light input over some integration time. As such, the integration time imposes an inherent limit on the maximum reading of the ADC. For the 13.7ms integration time, this limit is 5047. For the 101ms integration time, this limit is 37177. For the full 402ms integration time, the maximum is actually defined by the size of the output variable, an 16-bit unsigned integer, at 65535. This fact should be taken into account when selecting a value for the interrupt high and low threshold values. For instance, if the integration time is 13.7ms and a high threshold of 6000 is set, that threshold will never be reached.

Example Code

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.

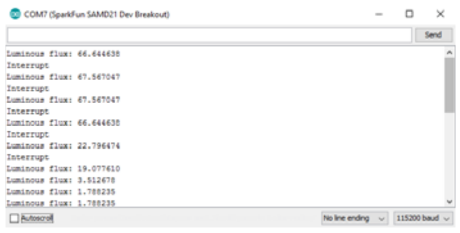
Make sure to install the APDS-9301 Arduino library before using the example code. The library can be found in the APDS-9301 GitHub repository:

SPARKFUN APDS-9301 GITHUB REPOSITORY

Below is a simple example code for Arduino that uses the APDS-9301 library. First, the example code sets the gain and integration time. Then enables and sets thresholds for the interrupt. Finally, the code reads the current lux level and prints it to the serial port. It also prints a message when the light level exceeds a certain threshold. It should get you up and running in no time!

```
#include "Wire.h"
#include <Sparkfun_APDS9301_Library.h>

APDS9301 apds;

#define INT_PIN 2 // We'll connect the INT pin from our senso
r to the
                  // INT0 interrupt pin on the Arduino.
bool lightIntHappened = false; // flag set in the interrupt t
o let the
                  //  mainline code know that an interrupt occ
urred.

void setup()
{
  SerialUSB.begin(115200);
  Wire.begin();

  // APDS9301 sensor setup.
  apds.begin(0x39);  // We're assuming you haven't changed th
e I2C
                     //  address from the default by solderin
g the
                     //  jumper on the back of the board.
  apds.setGain(APDS9301::LOW_GAIN); // Set the gain to low. St
rictly
                     //  speaking, this isn't necessary, as th
e gain
                     //  defaults to low.
  apds.setIntegrationTime(APDS9301::INT_TIME_13_7_MS); // Set
the
                     //  integration time to the shortest inte
rval.
                     //  Again, not strictly necessary, as thi
s is
                     //  the default.
  apds.setLowThreshold(0); // Sets the low threshold to 0, eff
ectively
                     //  disabling the low side interrupt.
  apds.setHighThreshold(50); // Sets the high threshold to 50
0. This
                     //  is an arbitrary number I pulled out o
f thin
                     //  air for purposes of the example. Whe
n the CH0
                     //  reading exceeds this level, an interr
upt will
                     //  be issued on the INT pin.
  apds.setCyclesForInterrupt(1); // A single reading in the th
reshold
                     //  range will cause an interrupt to trig
ger.
  apds.enableInterrupt(APDS9301::INT_ON); // Enable the interr
upt.
  apds.clearIntFlag();

  // Interrupt setup
  pinMode(INT_PIN, INPUT_PULLUP); // This pin must be a pullu
p or have
                     //  a pullup resistor on it as the interr
upt is a
                     //  negative going open-collector type ou
tput.
  attachInterrupt(digitalPinToInterrupt(2), lightInt, FALLIN
G);
}

void loop()
{
  static unsigned long outLoopTimer = 0;
  apds.clearIntFlag();

  // This is a once-per-second timer that calculates and print
s off
  //  the current lux reading.
  if (millis() - outLoopTimer >= 1000)
  {
    outLoopTimer = millis();

    SerialUSB.print("Luminous flux: ");
    SerialUSB.println(apds.readLuxLevel(),6);

    if (lightIntHappened)
    {
      SerialUSB.println("Interrupt");
      lightIntHappened = false;
    }
  }
}

void lightInt()
{
  lightIntHappened = true;
}
```

After uploading code, try opening the Arduino serial monitor at 115200 baud and observe the sensor's output.



```
Luminous flux: 66.644638
Interrupt
Luminous flux: 67.567047
Interrupt
Luminous flux: 67.567047
Interrupt
Luminous flux: 66.644638
Interrupt
Luminous flux: 22.796474
Interrupt
Luminous flux: 19.077610
Luminous flux: 3.512678
Luminous flux: 1.788235
Luminous flux: 1.788235
```

As you can see, once per second, the sketch will print the luminous flux and whether or not the level selected for an interrupt has been exceeded.

Resources and Going Further

Now that you've successfully got your APDS-9301 sensor up and running, it's time to incorporate it into your own project!

For more information on the APDS-9301 sensor, check out the resources below:

- Wikipedia: Lux - To help you better understand what it is you're measuring, exactly.
- APDS-9301 Breakout Schematic (PDF)
- APDS-9301 Breakout Eagle Files (ZIP)
- APDS-9301 Datasheet (AV02-2315EN0.PDF) - If you want to move beyond the limits of the library and explore the base features of the chip itself.
- APDS-9301 Sensor GitHub Repository - Eagle files and example code for the APDS-9301 Sensor.

Need some inspiration for your next project? Check out some of these related tutorials:

Soil moisture-sensing by hacking a solar light
How to take a solar powered pathway light from a hardware store and make it into a crude soil moisture sensor.

TEMT6000 Ambient Light Sensor Hookup Guide
Bring the ability to detect light levels to any project with the SparkFun TEMT6000 Ambient Light Sensor Breakout.

SCiO Pocket Molecular Scanner Teardown
A teardown of the SCiO, a pocket molecular scanner.

LED PomPom Headbands
Follow this tutorial to make your own light up PomPom headband! Try the beginner version if you are new to electronics or the advanced version if you have some more experience!