Adafruit LTR-329 and LTR-303 Light Sensors

Created by Kattni Rembor


Last updated on 2023-08-29 04:51:17 PM EDT
# Table of Contents

## Overview

## Pinouts
- Power Pins
- I2C Logic Pins
- On LED and LED Jumper
- LTR-303 Interrupt Pin

## Python & CircuitPython
- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of LTR329_LTR303 Library
- CircuitPython Usage
- Python Usage
- Example Code for LTR-329
- LTR-303 Example
- LTR-303 Wiring
- Example Code for LTR-303

## Python Docs

## Arduino
- Wiring
- Library Installation
- Example Code

## Arduino Docs

## Downloads
- Files
- Schematic and Fab Print LTR-329
- Schematic and Fab Print LTR-303
Overview

The Adafruit LTR-329 and LTR-303 Light Sensors are simple and popular low-cost I2C digital light sensors that are easy to integrate into your project for reliable and wide-ranging light measurements.

Perfect for a wide range of project use cases: Should we turn up the brightness of our display or dim it to save power? Which direction should your robot move to stay in an area with the most light? Is it day or night? All of these questions can be answered with the help of these sensors. They're small, capable and inexpensive light sensors.
that you can include in your next project to add the detection and measurement of light.

The LTR-329 and LTR-303 provide 16-bit light measurements for both Infrared and Visible+IR spectrums. Subtract one from the other to get human-visible light. Thanks to configurable gain and integration time settings, both sensors can measure from 0 to 65K+ lux.

The 303 (compared to the '329) also comes with alert and IRQ capabilities so you can have a pin trigger when the reading goes above or below threshold values. Great if you want ultra-low power usage. If you don't need a version with interrupt
output, check out the Adafruit LTR-329 which is basically the same sensor but without the IRQ capability.

Sensors tend to come in small packages and the LTR-329 and LTR-303 are no different. Not much bigger than a grain of rice, these handy light-sensing friends need some help to be used by folks experimenting and without the desire or tools to work with surface mount parts. We're here to help!

Packaged on a PCB in our STEMMA QT form factor, these sensors come integrated with a voltage regulator and level shifting circuitry to allow them to be used with 3.3V devices, like a Feather M4 or Raspberry Pi, or 5V devices, such as an Arduino. Rather
than working with the itty bitty little contacts on the sensor, the PCB it's packaged on breaks out all the pins to a standard 0.1" / 2.54mm pitch header.

To make things easier and a bit more flexible, we've also included :SparkFun Qwiic () compatible STEMMA QT () connectors for the I2C bus so you don't even need to solder! Just plug in a compatible cable and attach it to your MCU of choice, and you’re ready to load up some software and measure some light. **QT Cable is not included, but we have a variety in the shop**.

To help with the software part we’ve written a library that you can use with CircuitPython / Python devices as well as computers like the Raspberry Pi by installing it using PyPi (). You can even use it on a full-sized computer by using an MCP2221 breakout ().
Arduino users can use our LTR-303 Arduino library with lots of examples to get started fast.

## Pinouts

![LTR-303 Arduino light sensor](image)

The power pins, I2C logic pins, and I2C address are the same for both breakouts.

### Power Pins

- **VIN** - This is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V.
- **3V** - This is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like.
- **GND** - This is the common ground for power and logic.

### I2C Logic Pins

The I2C address is 0x29.

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
- **SDA** - I2C data pin, connect to your microcontroller I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
- **STEMMA QT** - These connectors allow you to connectors to dev boards with STEMMA QT connectors or to other things with various associated accessories.
On LED and LED Jumper

- **on LED** - On the front of the breakout, right above the STEMMA QT connector on the left, is the on LED. It is labeled on on the silk. This green LED lights up when the board is powered.

- **LED jumper** - On the back of the board is a jumper labeled LED. To disable the on LED, cut the trace between the two pads. To enable it again, bridge the two pads together with solder.

LTR-303 Interrupt Pin

- **INT** - This is the interrupt pin, available only on the LTR-303. The alert and IRQ capabilities enable you to have a pin trigger when the reading goes above or below threshold values.

Python & CircuitPython

It's easy to use the LTR-329 and the LTR-303 with Python or CircuitPython, and the [Adafruit_CircuitPython_LTR329_LTR303](https://github.com/adafruit/Adafruit_CircuitPython_LTR329_LTR303) module. This module allows you to easily write Python code that reads the values from the LTR-329 and LTR-303 light sensors. 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.

This page focuses heavily on the LTR-329, but using the LTR-303 is basically the same. At the end of the page you'll find an LTR-303-specific example utilizing the int pin.
CircuitPython Microcontroller Wiring

These wiring examples show the LTR-329, but connecting the LTR-303 is exactly the same.

Wire up an LTR-329 to your board exactly as shown below. Here's an example of wiring a Feather RP2040 to the sensor with I2C using one of the handy STEMMA QT connectors:

- Board 3V to sensor VIN (red wire)
- Board GND to sensor GND (black wire)
- Board SCL to sensor SCL (yellow wire)
- Board SDA to sensor SDA (blue wire)

You can also connect the LTR-329 using standard 0.1" pitch headers to wire it up on a breadboard:

- Board 3V to sensor VIN (red wire)
- Board GND to sensor GND (black wire)
- Board SCL to sensor SCL (yellow wire)
- Board SDA to sensor SDA (blue wire)

Python Computer Wiring

Since there's dozens of Linux computers/boards you can use, below shows wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported.
Here's the Raspberry Pi wired to the LTR-329 using I2C and a STEMMA QT connector:

Pi 3V to sensor VIN (red wire)
Pi GND to sensor GND (black wire)
Pi SCL to sensor SCL (yellow wire)
Pi SDA to sensor SDA (blue wire)

Here is the Raspberry Pi wired to the LTR-329 using a solderless breadboard:

Pi 3V to sensor VIN (red wire)
Pi GND to sensor GND (black wire)
Pi SCL to sensor SCL (yellow wire)
Pi SDA to sensor SDA (blue wire)

Python Installation of LTR329_LTR303 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready!

Once that's done, from your command line run the following command:

```
$ pip3 install adafruit-circuitpython-ltr329-ltr303
```

If your default Python is version 3, you may need to run `pip` instead. Make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!
CircuitPython Usage

To use with CircuitPython, you need to first install the LTR329_LTR303 library, and its dependencies, into the lib folder on your CIRCUITPY drive. Then you need to update code.py with the example script.

Thankfully, we can do this in one go. In the example below, click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY/lib folder should contain the following folders and file:

- adafruit_bus_device/
- adafruit_register/
- adafruit_ltr329_ltr303.mpy

Python Usage

Once you have the library pip3 installed on your computer, copy or download the following example to your computer, and run the following, replacing code.py with whatever you named the file:

```
pip3 code.py
```

Example Code for LTR-329

```python
# SPDX-FileCopyrightText: Copyright (c) 2022 ladyada for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

import time
import board
from adafruit_ltr329_ltr303 import LTR329
```
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
# microcontroller

time.sleep(0.1)  # sensor takes 100ms to 'boot' on power up
ltr329 = LTR329(i2c)

while True:
    print("Visible + IR:", ltr329.visible_plus_ir_light)
    print("Infrared    ":, ltr329.ir_light)
    print()
    time.sleep(0.5)  # sleep for half a second

If running CircuitPython: Once everything is saved to the CIRCUITPY drive, connect to
the serial console () to see the data printed out!

If running Python: The console output will appear wherever you are running Python.

Try covering the sensor with your finger to see the values decrease. Shine a flashlight
on the sensor to see them increase.

That's all there is to using the LTR-329 with CircuitPython!

LTR-303 Example

This example is specific to the LTR-303. It enables the interrupt functionality. To see it
working, you'll wire up an LED to the pin.

LTR-303 Wiring

Wire up the LTR-303 as shown here.
Board 3V to power rail on breadboard (red wire)
Sensor VIN to power rail on breadboard (red wire)
LED + (anode) to power rail on breadboard (red wire)
Board GND to ground rail on breadboard (black wire)
Sensor GND to ground rail on breadboard (black wire)
Board SCL to sensor SCL (yellow wire)
Board SDA to sensor SDA (blue wire)
LED - (cathode) to 470Ω resistor
Sensor INT to 470Ω resistor

Example Code for LTR-303

As explained above, click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

```python
import time
import board
import adafruit_ltr329_ltr303 as adafruit_ltr303

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a microcontroller

time.sleep(0.1)  # sensor takes 100ms to 'boot' on power up
ltr303 = adafruit_ltr303.LTR303(i2c)

# Can set the ALS light gain, can be: 1, 2, 4, 8, 48 or 96 times
# to range from 1-64 kLux to 0.01~600 Lux
ltr303.als_gain = 96
print("LTR-303 ALS gain:", ltr303.als_gain)

# Can set the ALS measurement integration time, how long the sensor
# 'looks' for light data. Default is 100ms.
# Set to: 50, 100, 150, 200, 250, 300, 350, or 400 millisec
# ltr303.integration_time = 50
print("LTR-303 integration time (ms):", ltr303.integration_time)

# Can set the ALS measurement rate, how often the data register updates
# Default is 500ms. Must be equal or greater than the integration time
# Set to: 50, 100, 200, 500, 1000, 2000 millisec
# ltr303.measurement_rate = 500
print("LTR-303 measurement rate (ms):", ltr303.measurement_rate)

# Can put into stand-by mode at any time, for low power usage
# self.active_mode = False
```
# The LTR-303, unlike the LTR-329, can also generate an IRQ output
# The interrupt output can be enabled
ltr303.enable_int = True
# We can also change whether the polarity is active LOW (False) or HIGH (True)
# Default is LOW (False)
ltr303.int_polarity = False
# Then set the low and high thresholds that would trigger an IRQ!
ltr303.threshold_low = 2000  # interrupt goes off if BELOW this number
ltr303.threshold_high = 40000  # or ABOVE this number!
print("Interrupt thresholds: ", ltr303.threshold_low, ltr303.threshold_high)

# Finally, we can set how many measurements must be above/below before
# we trigger an IRQ - basically avoid spurious readings. A setting of 1
# means every value triggers an int, 2 means two consecutive readings to
# trigger... up to 16!
ltr303.int_persistence = 4

while True:
    # The sensor will let us know when the measurement time has
    # created a new data reading!
    if ltr303.new_als_data_available:
        # The sensor can get 'overwhelmed' by bright light if the
        # gain isn't set right, in which case the data is invalid.
        # We can check the data invalid first and toss out the reading...
        if ltr303.als_data_invalid:
            ltr303.throw_out_reading()  # perform & toss out the reading
            continue  # try again next time!

        # OR we can 'try' to do the read and get an exception if the
        # data is invalid
        try:
            # If we're using 'new_als_data_available' we should
            # read both channels ONCE only! To do that use...
            visible_plus_ir, ir = ltr303.light_channels
            # this will get both channels at once! (It's also faster)

            # Now we can do various math...
            print("Visible + IR: ", visible_plus_ir)
            print("Infrared    : ", ir)
            print("ALS gain:   ", ltr303.als_data_gain)
            print()

        except ValueError:
            # we can also check `ltr303.als_data_invalid` if we
            # want, to verify that
            print("Light sensor data invalid, trying again!")
            time.sleep(0.1)

If running CircuitPython: Once everything is saved to the CIRCUITPY drive, [connect to the serial console](#) to see the data printed out!

If running Python: The console output will appear wherever you are running Python.
Try covering the sensor with your finger to drop it below the lower threshold. The LED should blink regularly indicating the interrupt has been triggered.

Try shining a flashlight on the sensor to bring it above the upper threshold. You should get the same result from the LED.

That's all there is to using the LTR-303 interrupt functionality!

---

**Python Docs**

[Python Docs](#)

---

**Arduino**

Using the LTR-329 and LTR303 with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit_LTR329_LTR303 library and running the provided example code.

**Wiring**

> These wiring examples show the LTR-329, but connecting the LTR-303 is exactly the same.

Wire as shown for a 5V board like an Uno. If you are using a 3V board, like an Adafruit Feather, wire the board's 3V pin to the LTR-329 VIN.

Here is an Adafruit Metro wired up to the LTR-329 using the STEMMA QT connector:
Here is an Adafruit Metro wired up using a solderless breadboard:

Library Installation

You can install the Adafruit LTR329 LTR303 library for Arduino using the Library Manager in the Arduino IDE.

Click the Manage Libraries ... menu item, search for LTR329_LTR303, and select the Adafruit LTR329 LTR303 library:
If asked about dependencies, click "Install all".

Example Code

```cpp
/*****************************************************
This is an example for the LTR329 light sensor that reads both channels
and demonstrates how to set gain and check data validity

Designed specifically to work with the LTR-329 light sensor from Adafruit
----> https://www.adafruit.com/product/5591

These sensors use I2C to communicate, 2 pins are required to
interface
*****************************************************/

#include "Adafruit_LTR329_LTR303.h"

Adafruit_LTR329 ltr = Adafruit_LTR329();

void setup() {
  Serial.begin(115200);
  Serial.println("Adafruit LTR-329 advanced test");

  if (!ltr.begin()) {
    Serial.println("Couldnt find LTR sensor!");
    while (1) delay(10);
  }
  Serial.println("Found LTR sensor!");

  ltr.setGain(LTR3XX_GAIN_2);
  Serial.print("Gain : ");
  switch (ltr.getGain()) {
    case LTR3XX_GAIN_1: Serial.println(1); break;
    case LTR3XX_GAIN_2: Serial.println(2); break;
    case LTR3XX_GAIN_4: Serial.println(4); break;
    case LTR3XX_GAIN_8: Serial.println(8); break;
  }

  Serial.println("Setting gain to 2...");

  Serial.println("Setting gain to 4...");

  Serial.println("Setting gain to 8...");

  Serial.println("Setting gain to 1...");

  Serial.println("Checking data validity...");

  while (1) {
    Serial.println(ltr.checkDataValidity());
  }
}
```
case LTR3XX_GAIN_2: Serial.println(2); break;
case LTR3XX_GAIN_4: Serial.println(4); break;
case LTR3XX_GAIN_8: Serial.println(8); break;
case LTR3XX_GAIN_48: Serial.println(48); break;
case LTR3XX_GAIN_96: Serial.println(96); break;
}
ltr.setIntegrationTime(LTR3XX_INTEGTIME_100);
Serial.print("Integration Time (ms): ");
switch (ltr.getIntegrationTime()) {
  case LTR3XX_INTEGTIME_50: Serial.println(50); break;
  case LTR3XX_INTEGTIME_100: Serial.println(100); break;
  case LTR3XX_INTEGTIME_150: Serial.println(150); break;
  case LTR3XX_INTEGTIME_200: Serial.println(200); break;
  case LTR3XX_INTEGTIME_250: Serial.println(250); break;
  case LTR3XX_INTEGTIME_300: Serial.println(300); break;
  case LTR3XX_INTEGTIME_350: Serial.println(350); break;
  case LTR3XX_INTEGTIME_400: Serial.println(400); break;
}
ltr.setMeasurementRate(LTR3XX_MEASRATE_200);
Serial.print("Measurement Rate (ms): ");
switch (ltr.getMeasurementRate()) {
  case LTR3XX_MEASRATE_50: Serial.println(50); break;
  case LTR3XX_MEASRATE_100: Serial.println(100); break;
  case LTR3XX_MEASRATE_200: Serial.println(200); break;
  case LTR3XX_MEASRATE_500: Serial.println(500); break;
  case LTR3XX_MEASRATE_1000: Serial.println(1000); break;
  case LTR3XX_MEASRATE_2000: Serial.println(2000); break;
}
}
void loop() {
  bool valid;
  uint16_t visible_plus_ir, infrared;
  if (ltr.newDataAvailable()) {
    valid = ltr.readBothChannels(visible_plus_ir, infrared);
    if (valid) {
      Serial.print("CH0 Visible + IR: ");
      Serial.print(visible_plus_ir);
      Serial.print("\t\tCH1 Infrared: ");
      Serial.println(infrared);
    }
  }
  delay(100);
}

Upload the sketch to your board and open up the Serial Monitor (Tools -> Serial Monitor) at 115200 baud. You should see the values from the light sensor being printed out. Cover it with your finger to see the values decrease. Shine a flashlight on it to see the values increase.
Downloads

Files

- LTR-329 Datasheet
- LTR-303 Datasheet
- LTR-329 Fritzing object in the Adafruit Fritzing Library
- LTR-303 Fritzing object in the Adafruit Fritzing library
- EagleCAD PCB files on GitHub

Schematic and Fab Print LTR-329
Schematic and Fab Print LTR-303