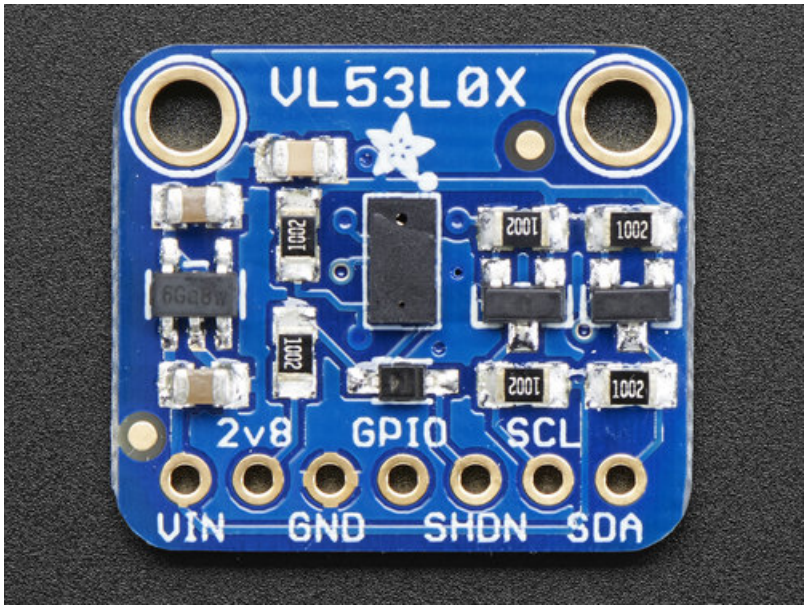
adafruit learning system

Adafruit VL53L0X Time of Flight Micro-LIDAR Distance Sensor Breakout

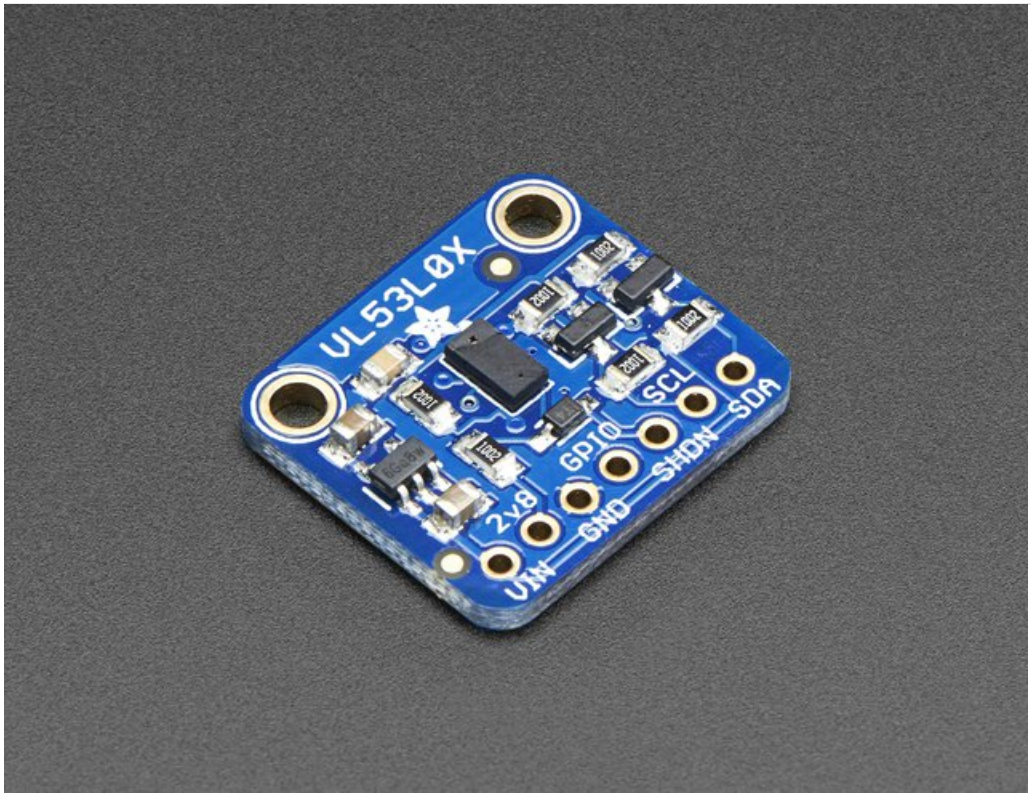
Created by lady ada



Last updated on 2018-08-22 03:57:47 PM UTC

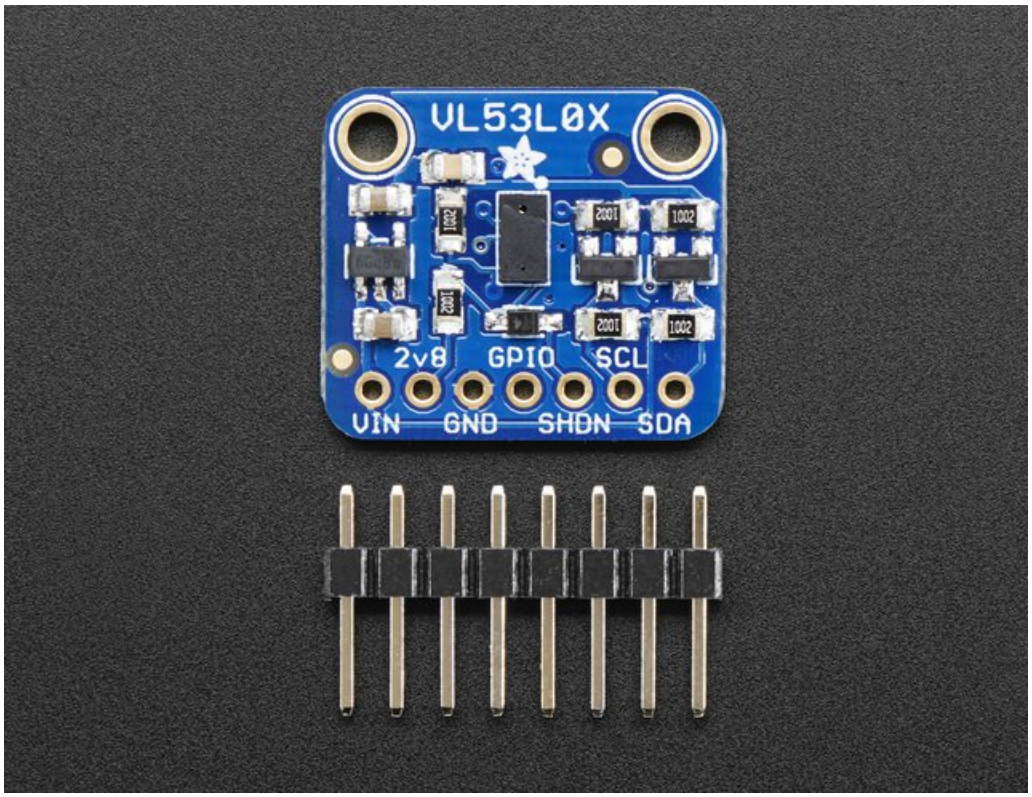
Guide Contents

Overview

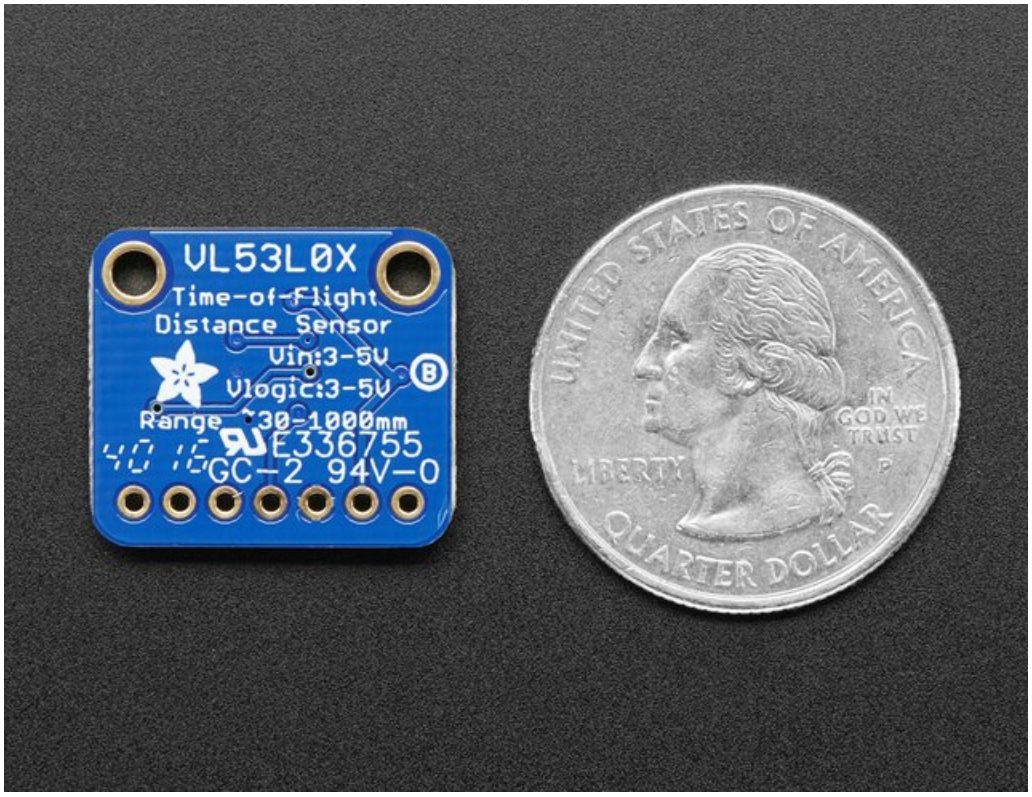


The VL53L0X is a *Time of Flight* distance sensor like no other you've used! The sensor contains a very tiny invisible laser source, and a matching sensor. The VL53L0X can detect the "time of flight", or how long the light has taken to bounce back to the sensor. Since it uses a very narrow light source, it is good for determining distance of only the surface directly in front of it. Unlike sonars that bounce ultrasonic waves, the 'cone' of sensing is very narrow. Unlike IR distance sensors that try to measure the amount of light bounced, the VL53L0x is much more precise and doesn't have linearity problems or 'double imaging' where you can't tell if an object is very far or very close.

This is the 'big sister' of the VL6180X ToF sensor, and can handle about 50 - 1200 mm of range distance. If you need a smaller/closer range, check out the VL6180X which can measure 5mm to 200mm.



The sensor is small and easy to use in any robotics or interactive project. Since it needs 2.8V power and logic we put the little fellow on a breakout board with a regulator and level shifting. You can use it with any 3-5V power or logic microcontroller with no worries. Each order comes with a small piece of header. Solder the header onto your breakout board with your iron and some solder and wire it up for instant distance-sensing-success!

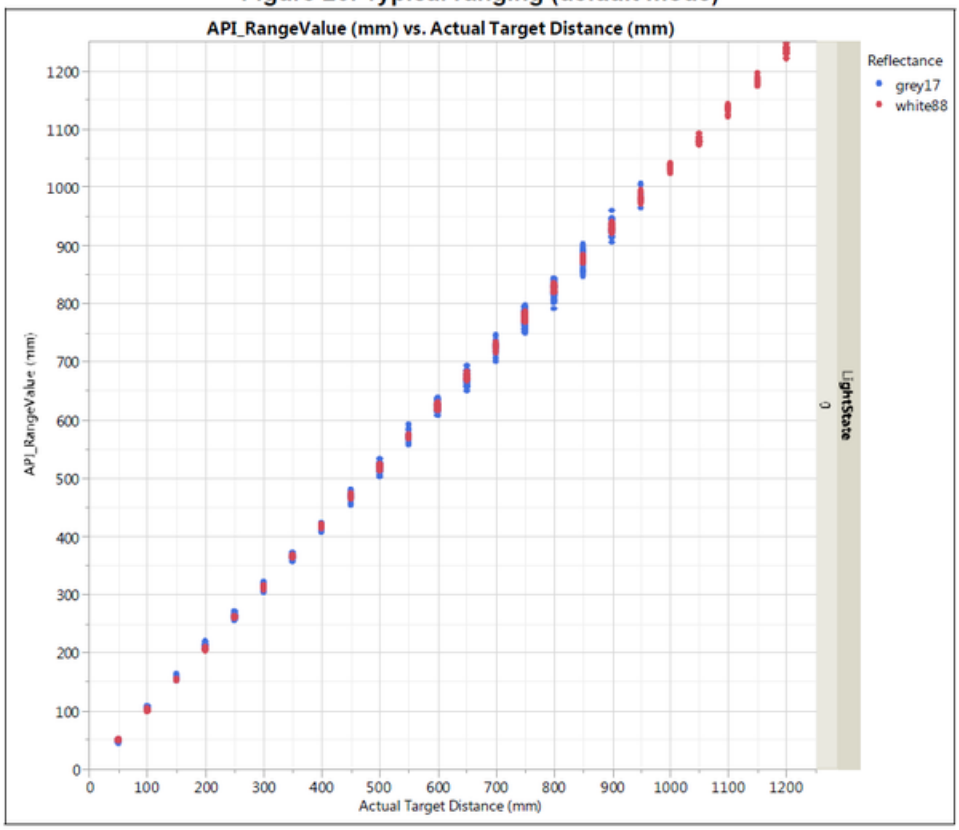


Communicating to the sensor is done over I2C with an API written by ST, so its not too hard to port it to your favorite microcontroller. We've written a wrapper library for Arduino so you can use it with any of your Arduino-compatible boards.

Sensing Capablities

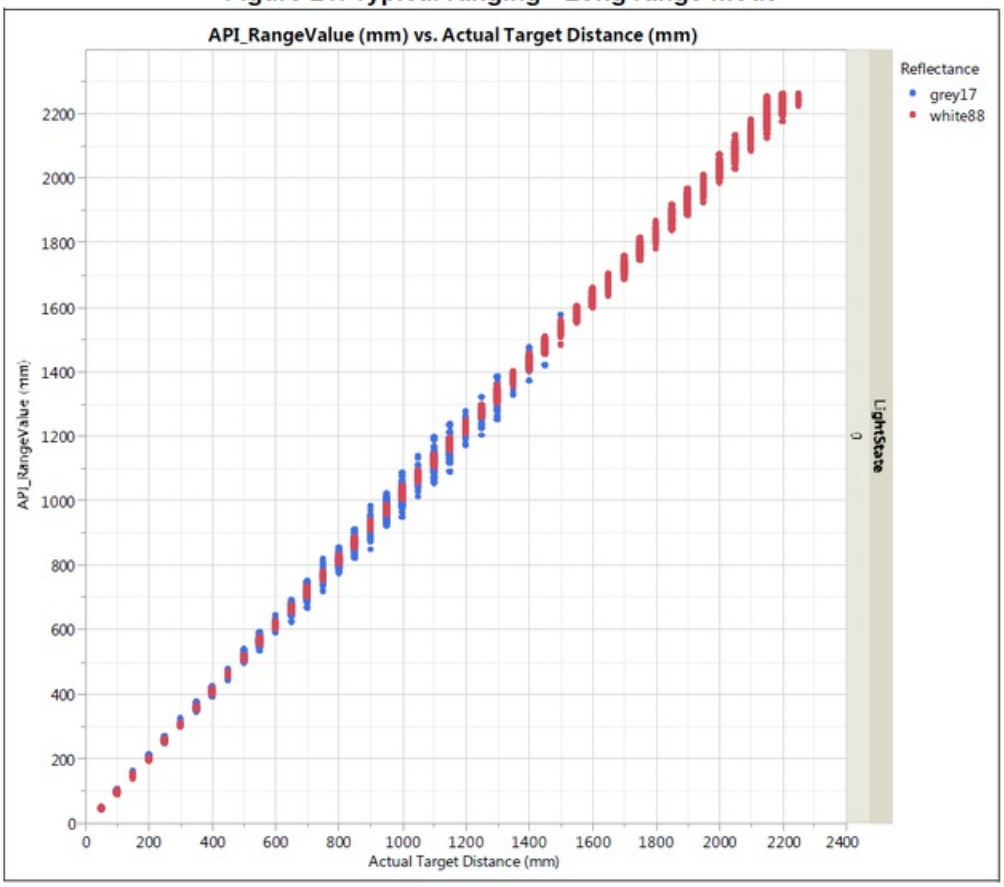
The sensor can measure approximately 50mm to 1.2 meter in default mode.

Figure 20. Typical ranging (default mode)



In 'long range' mode you can detect as far as 1.5 to 2 meters on a nice white reflective surface.

Figure 21. Typical ranging - Long range mode



Depending on ambient lighting and distance, you'll get 3 to 12% ranging accuracy - better lighting and shiny surfaces will give you best results. Some experimentation will be necessary since if the object absorbs the laser light you won't get good readings.

Table 11. Max ranging capabilities with 33ms timing budget

Target reflectance level (Full FOV)	Conditions	Indoor (2)	Outdoor overcast (2)
White Target (88%)	Typical	200cm+ (1)	80cm
	Minimum	120cm	60cm
Grey Target (17%)	Typical	80cm	50cm
	Minimum	70cm	40cm

Note (1): using long range API profile

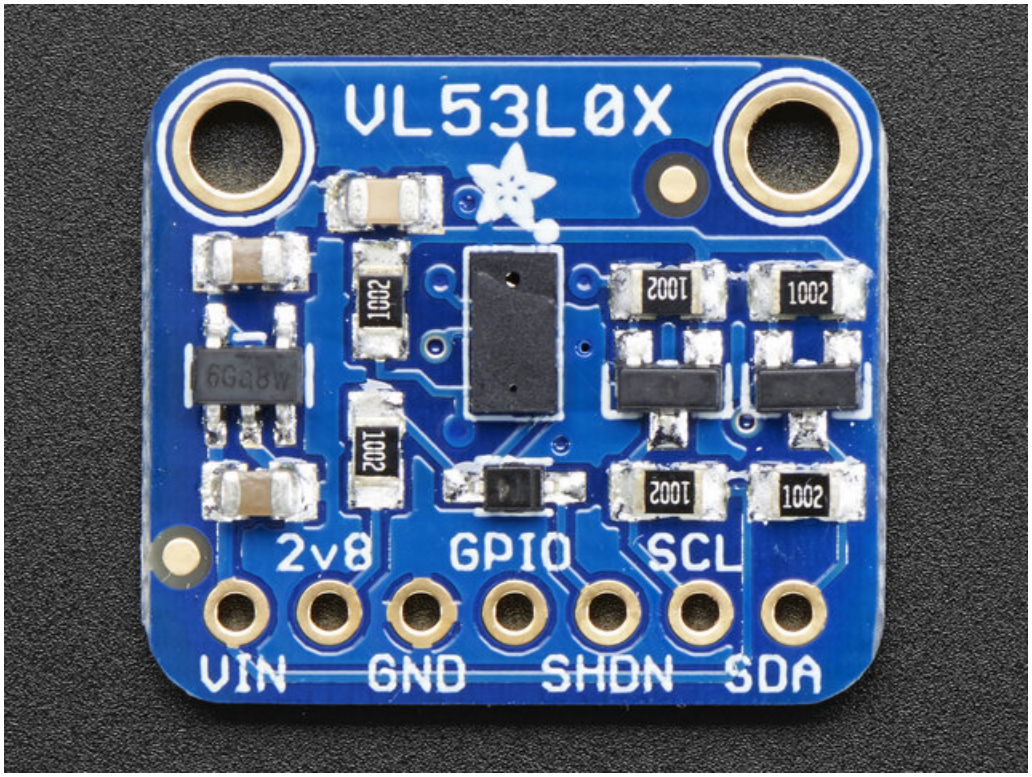
Table 12. Ranging accuracy

	Indoor (no infrared)			Outdoor		
Target reflectance level (Full FOV)	Distance	33ms	66ms	Distance	33ms	66ms
White Target (88%)	at 120cm	4%	3%	at 60cm	7%	6%
Grey Target (17%)	at 70cm	7%	6%	at 40cm	12%	9%

Pinouts

The VL53L0X 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 default I2C address is **0x29**. You *can* change it, but only in software. That means you have to wire the SHUTDOWN pin and hold all but one sensor in reset while you reconfigure one sensor at a time



Power Pins:

- **Vin** - this is the power pin. Since the chip uses 2.8 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
- **2v8** - this is the 2.8V output from the voltage regulator, you can grab up to 100mA from this if you like
- **GND** - common ground for power and logic

I2C Logic pins:

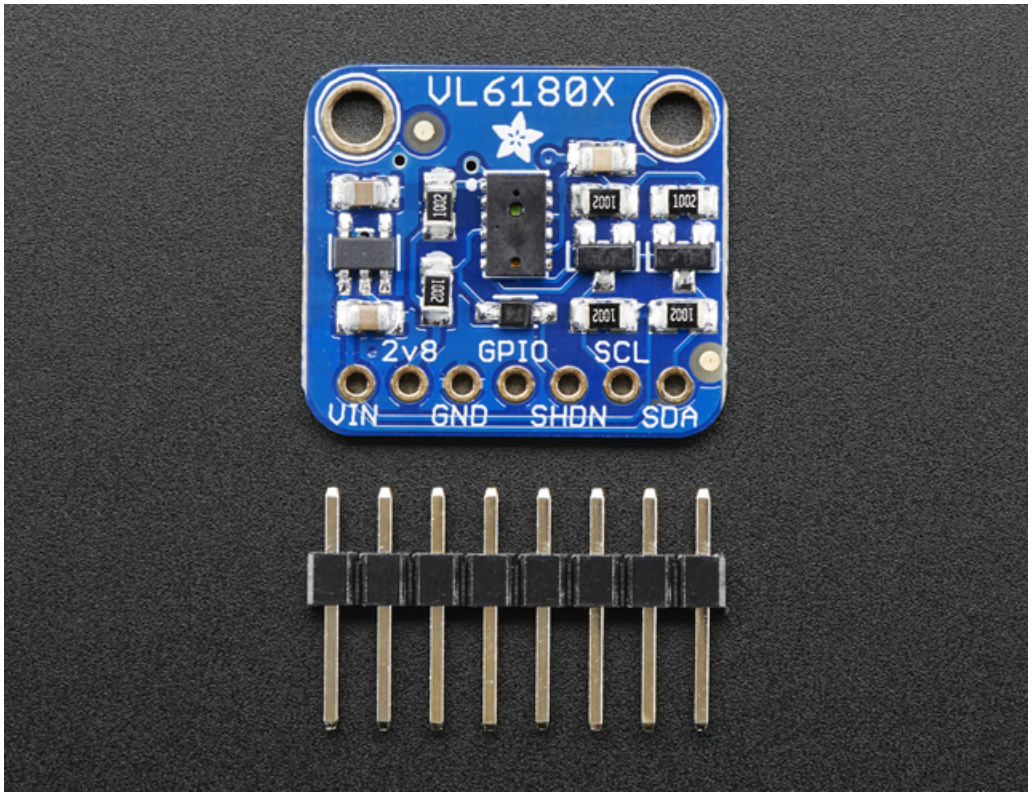
- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line.
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line.

Control Pins:

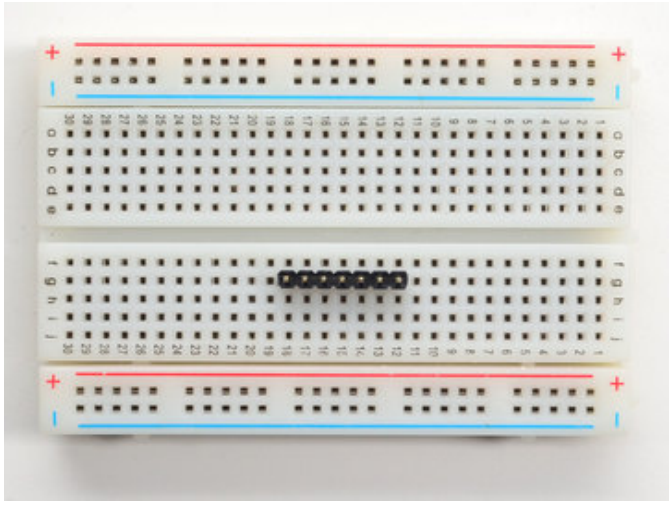
- **GPIO** - this is a pin that is used by the sensor to indicate that data is ready. It's useful for when doing continuous sensing. Note there is no level shifting on this pin, you may not be able to read the 2.8V-logic-level voltage on a 5V microcontroller (we could on an arduino UNO but no promises). Our library doesn't make use of this pin but for advanced users, it's there!
- **SHDN** - the shutdown pin for the sensor. By default it's pulled high. There's a level-shifting diode so you can use 3-5V logic on this pin. When the pin is pulled low, the sensor goes into shutdown mode.

Assembly

Don't forget to remove the protective cover off the sensor, it may be a clear or slightly tinted plastic! Otherwise you will get incorrect readings

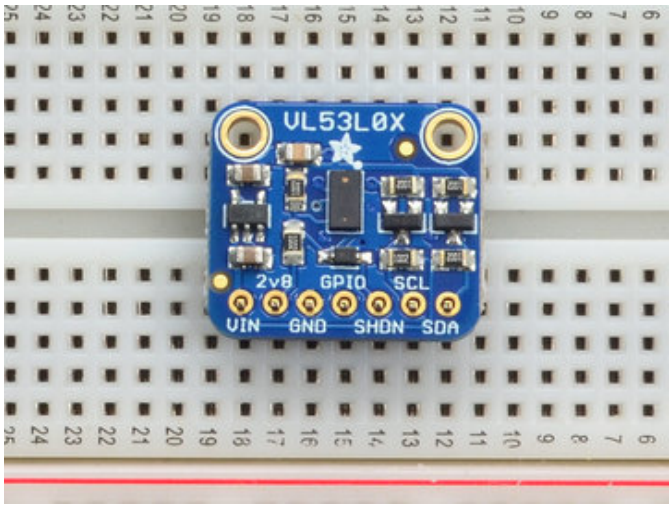


This page shows the VL53L0X or VL6180X sensor - the procedure is identical!

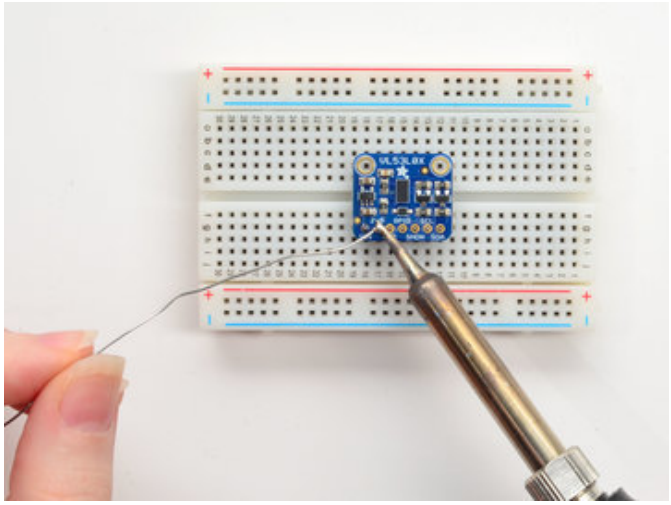


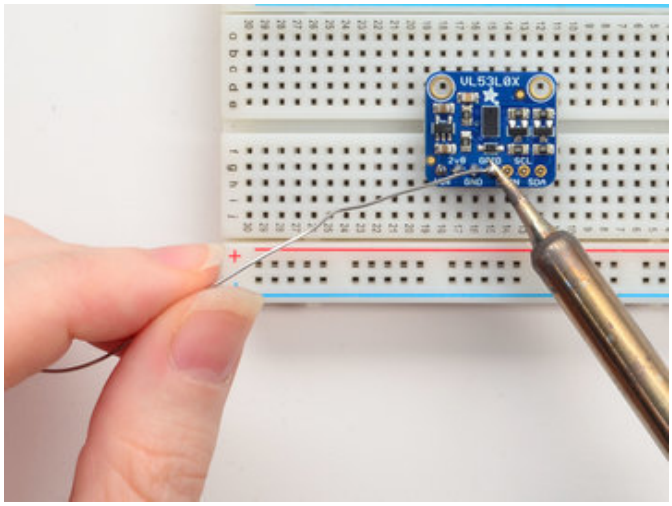
Prepare the header strip:

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



Place the Breakout board on top so the shorter ends of the pins line up though all the pads

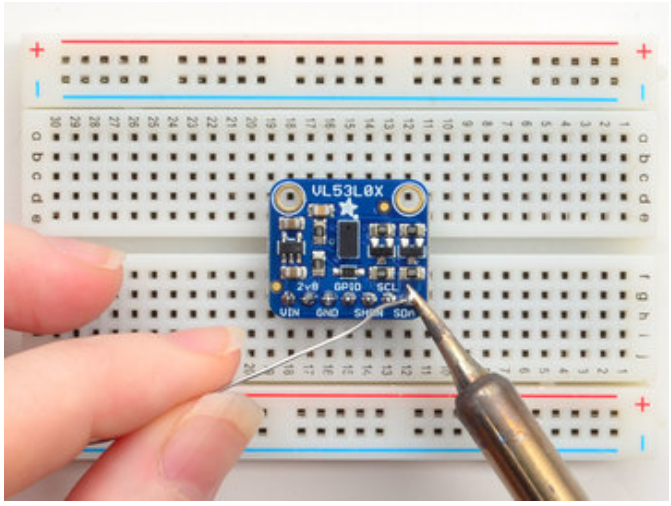


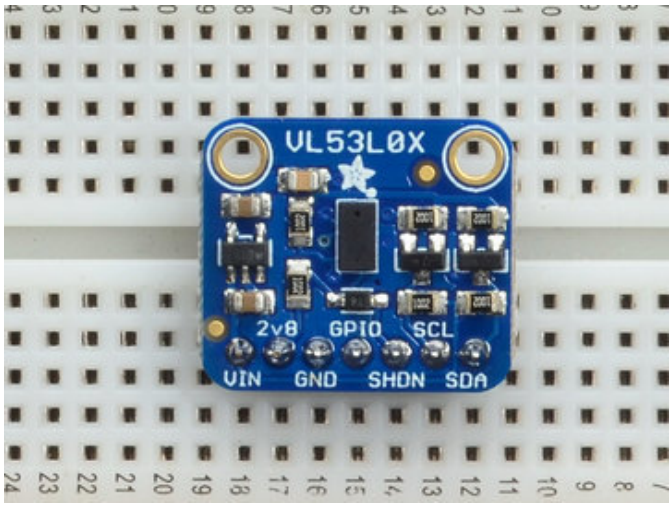


Solder!

Be sure to solder all pins for reliable electrical contact.

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

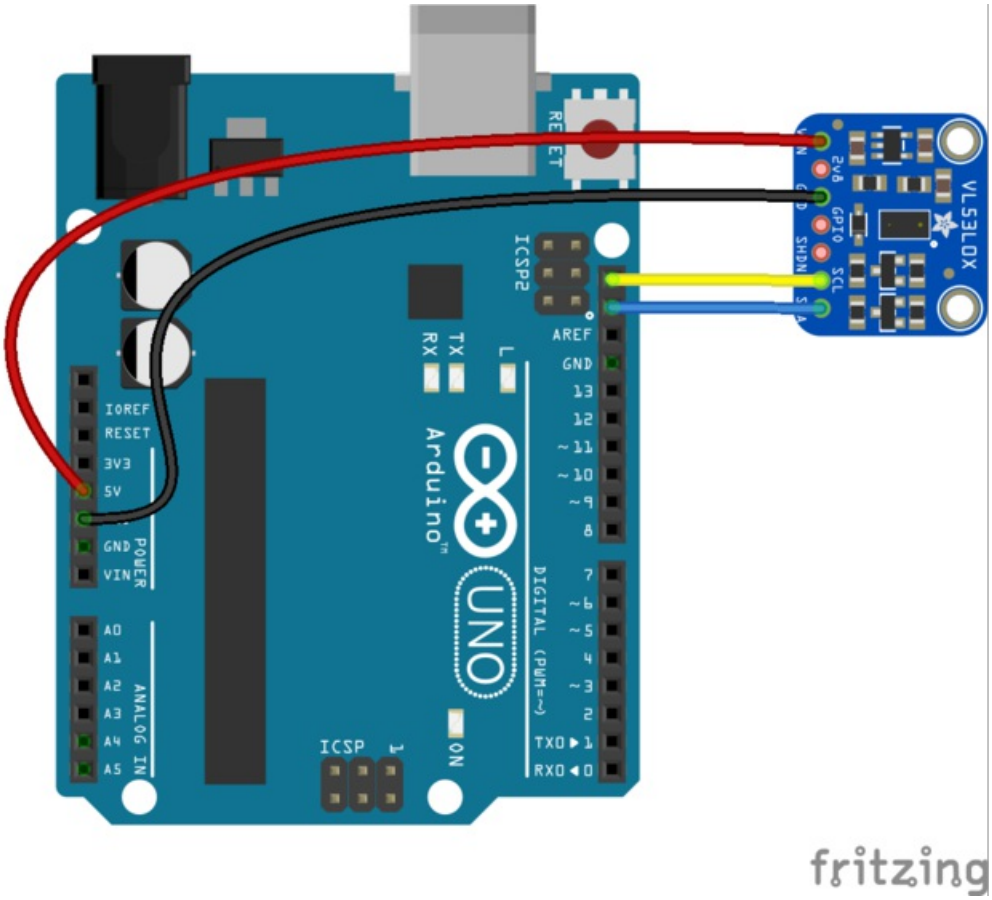




OK You're done! Check your work over and continue on to the next steps

Arduino Code

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 API code. We strongly recommend using an Arduino to start though!



- 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**

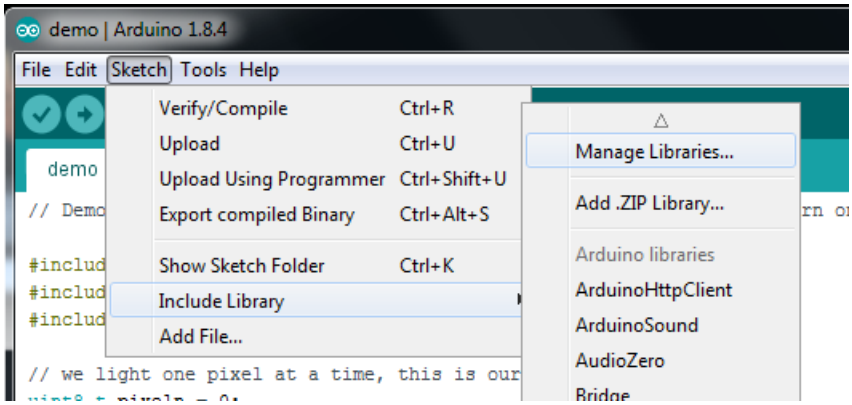
The VL53L0X has a default I2C address of **0x29**!

You *can* change it, but only in software. That means you have to wire the SHUTDOWN pin and hold all but one sensor in reset while you reconfigure one sensor at a time

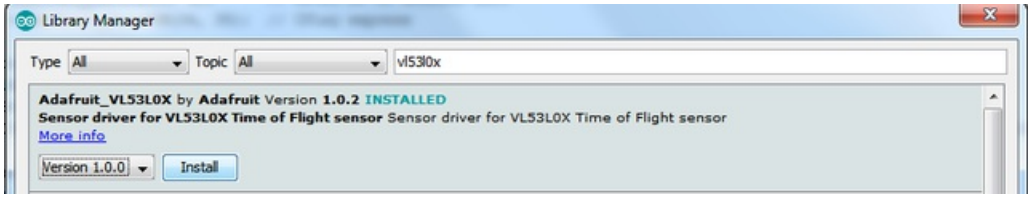
Download Adafruit_VL53L0X

To begin reading sensor data, you will need to install the Adafruit_VL53L0X Library (https://adafru.it/sDz).

The easiest way to do that is to open up the **Manage Libraries...** menu in the Arduino IDE



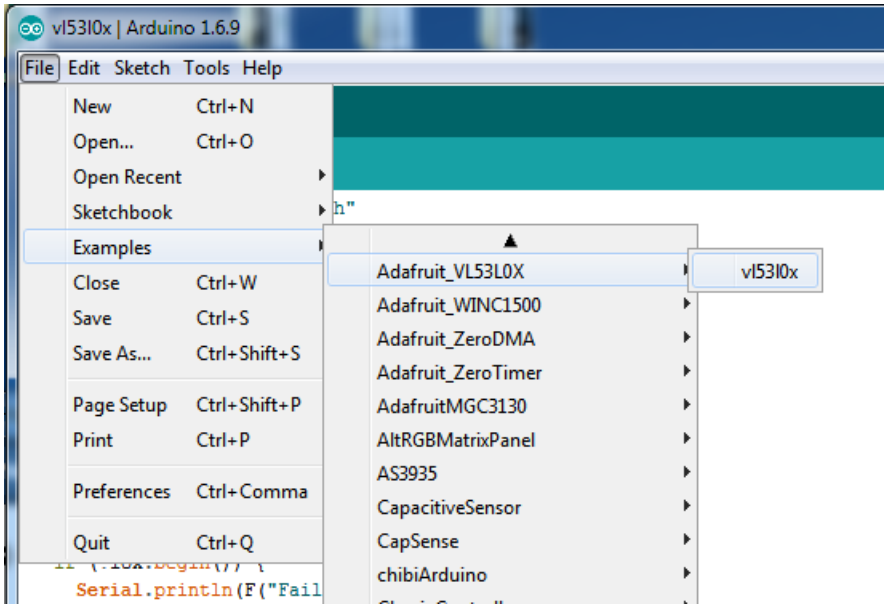
Then search for **Adafruit VL53L0X** and click **Install**



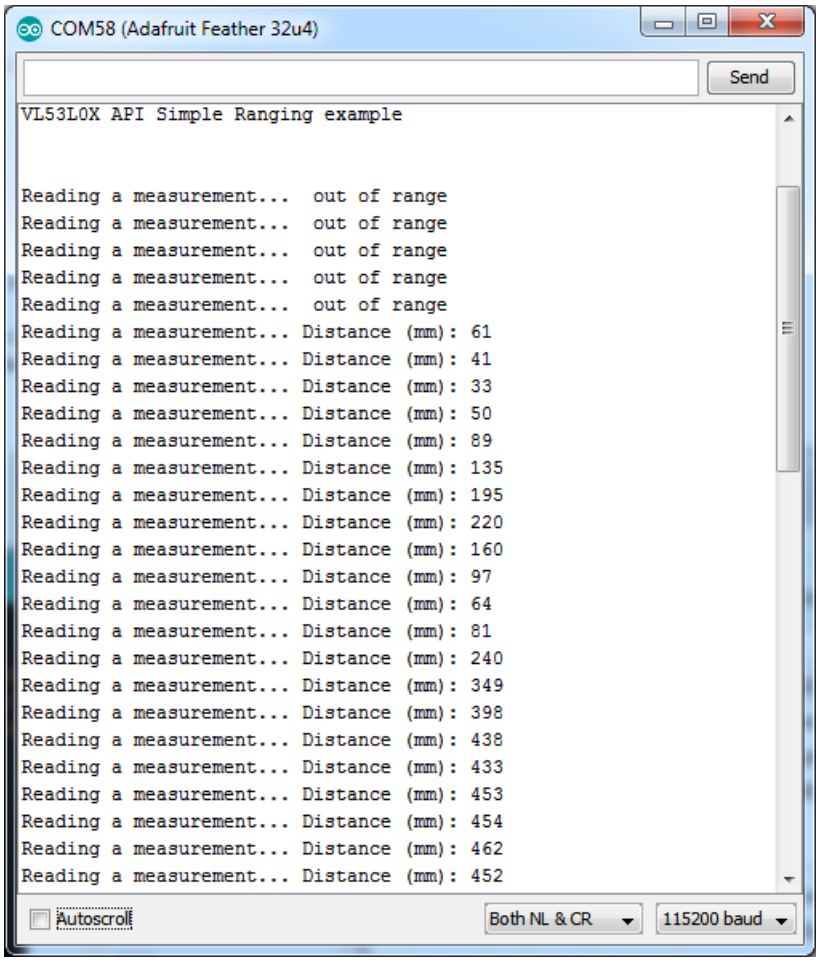
We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (https://adafru.it/aYM)

Load Demo

Open up **File->Examples->Adafruit_VL53L0X->vl53l0x** and upload to your Arduino wired up to the sensor



Thats it! Now open up the serial terminal window at 115200 speed to begin the test.



Move your hand up and down to read the sensor data. Note that when nothing is detected, it will say the reading is out of range

Don't forget to remove the protective plastic cover from the sensor before using!

Connecting Multiple Sensors

I2C only allows one address-per-device so you have to make sure each I2C device has a unique address. The default address for the VL53L0X is **0x29** but you *can* change this in software.

To set the new address you can do it one of two ways. During initialization, instead of calling lox.begin(), call `lox.begin(0x30)` to set the address to 0x30. Or you can, later, call `lox.setAddress(0x30)` at any time.

The good news is its easy to change, the annoying part is each *other* sensor has to be in shutdown. You can shutdown each sensor by wiring up to the **XSHUT** pin to a microcontroller pin. Then perform something like this pseudo-code:

1. Reset all sensors by setting all of their XSHUT pins low for delay(10), then set all XSHUT high to bring out of reset
2. Keep sensor #1 awake by keeping XSHUT pin high
3. Put all other sensors into shutdown by pulling XSHUT pins low
4. Initialize sensor #1 with lox.begin(*new_i2c_address*) Pick any number but 0x29 and it must be under 0x7F. Going with 0x30 to 0x3F is probably OK.
5. Keep sensor #1 awake, and now bring sensor #2 out of reset by setting its XSHUT pin high.
6. Initialize sensor #2 with lox.begin(*new_i2c_address*) Pick any number but 0x29 and whatever you set the first

sensor to

7. Repeat for each sensor, turning each one on, setting a unique address.

Note you must do this *every* time you turn on the power, the addresses are not permanent!

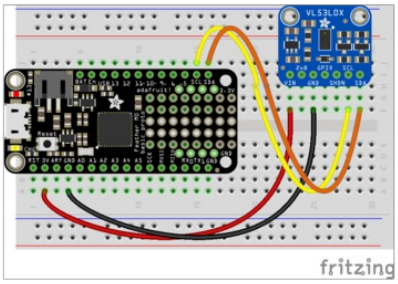
Python & CircuitPython

It's easy to use the VL53L0X sensor with Python and CircuitPython, and the Adafruit CircuitPython VL53L0X (https://adafru.it/C62) module. This module allows you to easily write Python code that reads the range from the sensor.

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 (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

First wire up a VL53L0X to your board exactly as shown on the previous pages for Arduino. Here's an example of wiring a Feather M0 to the sensor with an I2C connection:

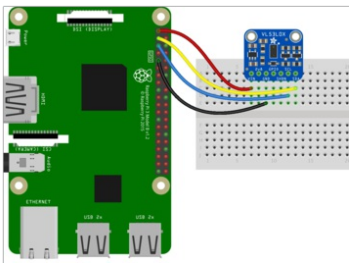


- **Board 3V** to **sensor VIN**
- **Board GND** to **sensor GND**
- **Board SCL** to **sensor SCL**
- **Board SDA** to **sensor SDA**

Python Computer Wiring

Since there's *dozens* of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C:



- **Pi 3V3** to **sensor VIN**
- **Pi GND** to **sensor GND**
- **Pi SCL** to **sensor SCL**
- **Pi SDA** to **sensor SDA**

CircuitPython Installation of VL53L0X Library

You'll need to install the Adafruit CircuitPython VL53L0X (https://adafru.it/C62) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/zdx). Our introduction guide has a great page on how to install the library bundle (https://adafru.it/ABU) for both express and non-express boards.

Remember for non-express boards like the, you'll need to manually install the necessary libraries from the bundle:

- **adafruit_vl53l0x.mpy**
- **adafruit_bus_device**

You can also download the **adafruit_vl53l0x.mpy** from its releases page on Github (https://adafru.it/C63).

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_vl53l0x.mpy,** and **adafruit_bus_device** files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz)so you are at the CircuitPython >>> prompt.

Python Installation of VL53L0X 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 (https://adafru.it/BSN)!

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

- `sudo pip3 install adafruit-circuitpython-vl53l0x`

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

CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and read the range and more from the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
import busio
import adafruit_vl53l0x
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_vl53l0x.VL53L0X(i2c)
```

Now you're ready to read values from the sensor using any of these properties:

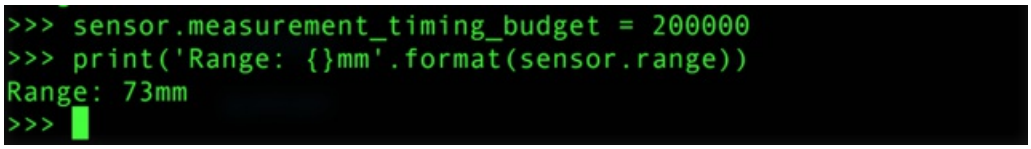
- **range** - The distance in millimeter to an object in front of the sensor.

```
print('Range: {}mm'.format(sensor.range))
```

```
>>> print('Range: {}mm'.format(sensor.range))
Range: 65mm
>>>
```

In addition you can adjust the measurement timing budget to change the speed and accuracy of the sensor. Get and set the **measurement_timing_budget** property with a value in nanoseconds. For example to increase the timing budget to a more accurate but slower 200ms value:

```
sensor.measurement_timing_budget = 200000
```



See the simpletest.py example (https://adafru.it/C64) for a complete demo of printing the range every second. Save this as **code.py** on the board and examine the REPL output to see the range printed every second.

That's all there is to using the VL53L0X with CircuitPython!

Full Example Code

```
# Simple demo of the VL53L0X distance sensor.
# Will print the sensed range/distance every second.
import time

import board
import busio

import adafruit_vl53l0x

# Initialize I2C bus and sensor.
i2c = busio.I2C(board.SCL, board.SDA)
vl53 = adafruit_vl53l0x.VL53L0X(i2c)

# Optionally adjust the measurement timing budget to change speed and accuracy.
# See the example here for more details:
#   https://github.com/pololu/vl53l0x-arduino/blob/master/examples/Single/Single.ino
# For example a higher speed but less accurate timing budget of 20ms:
#vl53.measurement_timing_budget = 20000
# Or a slower but more accurate timing budget of 200ms:
#vl53.measurement_timing_budget = 200000
# The default timing budget is 33ms, a good compromise of speed and accuracy.

# Main loop will read the range and print it every second.
while True:
    print('Range: {0}mm'.format(vl53.range))
    time.sleep(1.0)
```

Python Docs

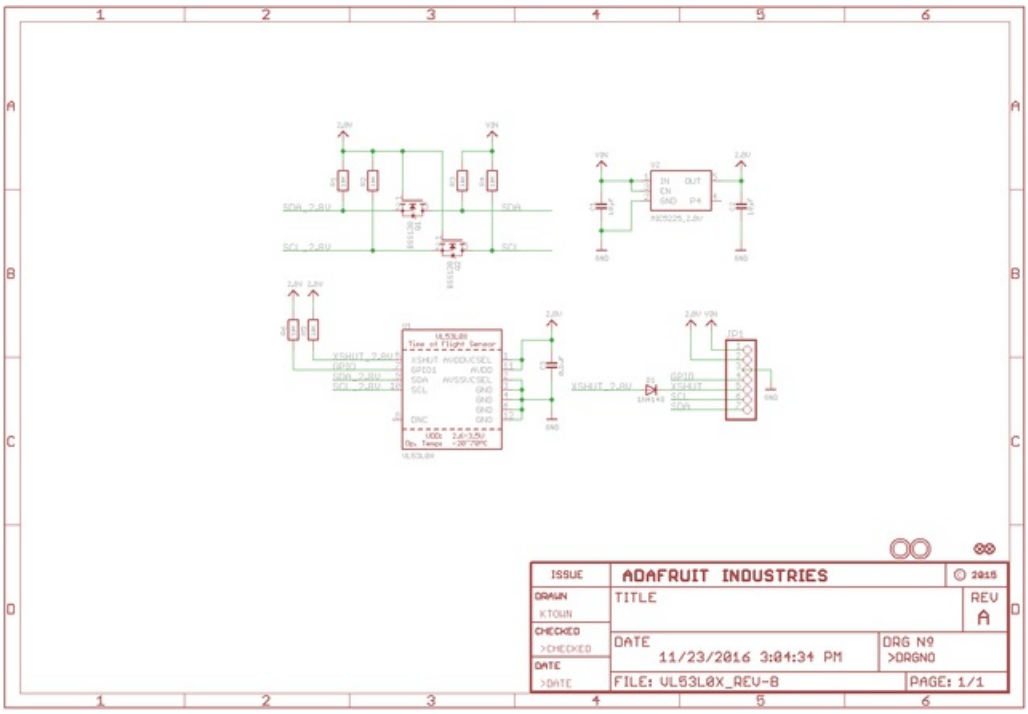
Python Docs (https://adafru.it/C65)

Downloads

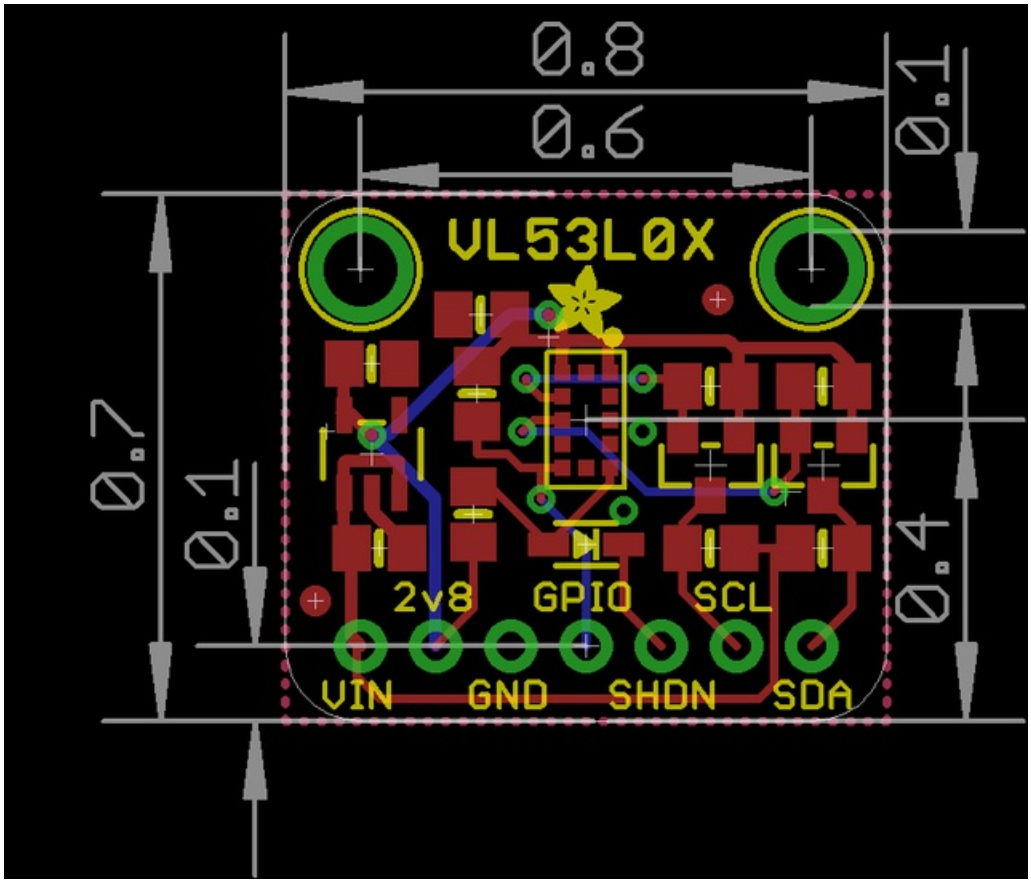
Files & Datasheets

- Datasheet for the VL53L0X (https://adafru.it/sDw)
- ST product page (https://adafru.it/sDx) with more details including API downloads
- Fritzing object in Adafruit Fritzing library (https://adafru.it/aP3)
- EagleCAD PCB files in GitHub (https://adafru.it/sDy)
- Adafruit VL53L0X library (https://adafru.it/sDz)
- Pololu VI53L0X library (https://adafru.it/sTf)

Schematic & Fabrication Print



Use M2.5 or #2-56 screws to mount


0.8
0.6
0.1
VL53L0X
0.7
0.1
0.4
2v8
GPIO
SCL
VIN
GND
SHDN
SDA