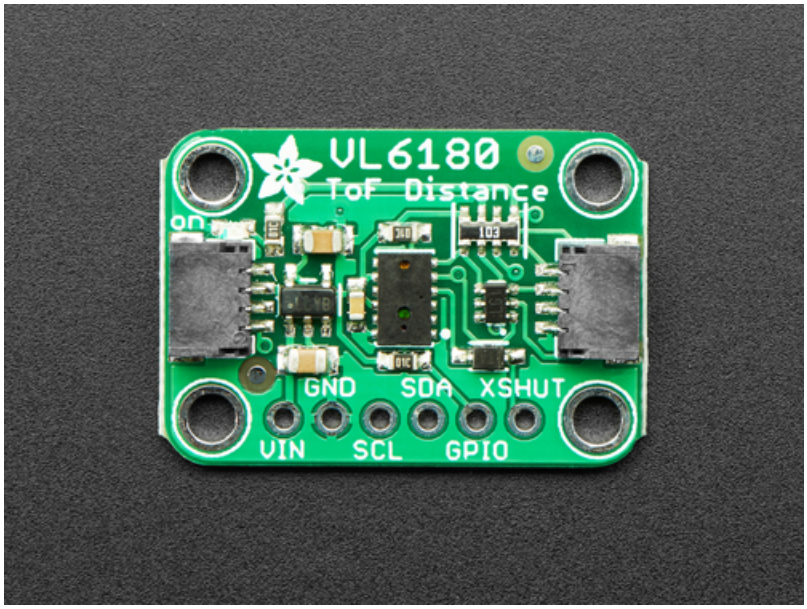
adafruit learning system

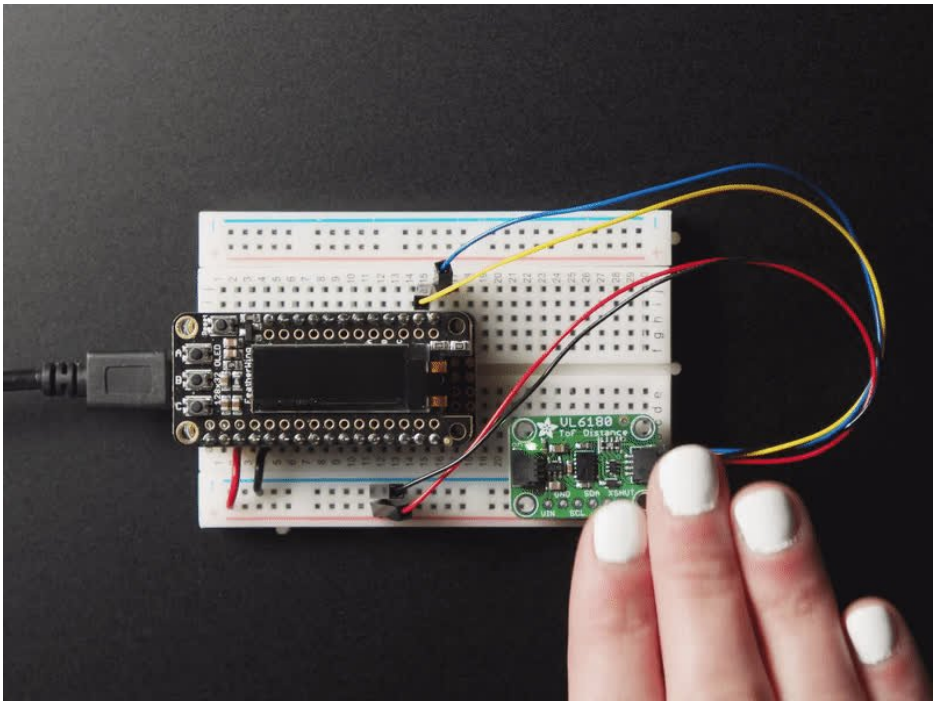
Adafruit VL6180X Time of Flight Micro-LIDAR Distance Sensor Breakout

Created by lady ada

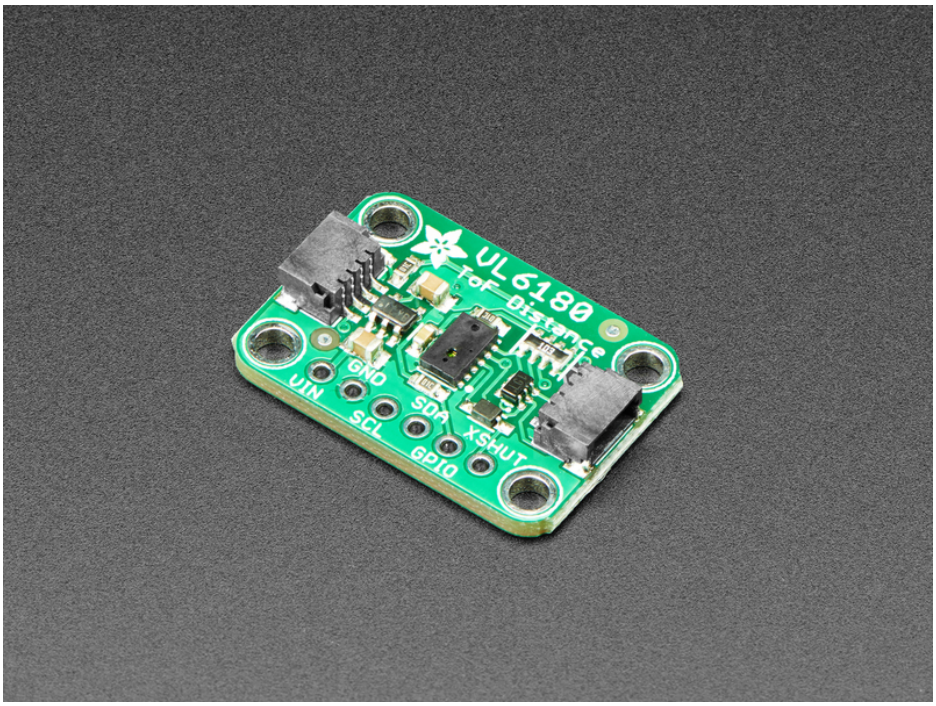


Last updated on 2020-03-27 11:52:04 AM EDT

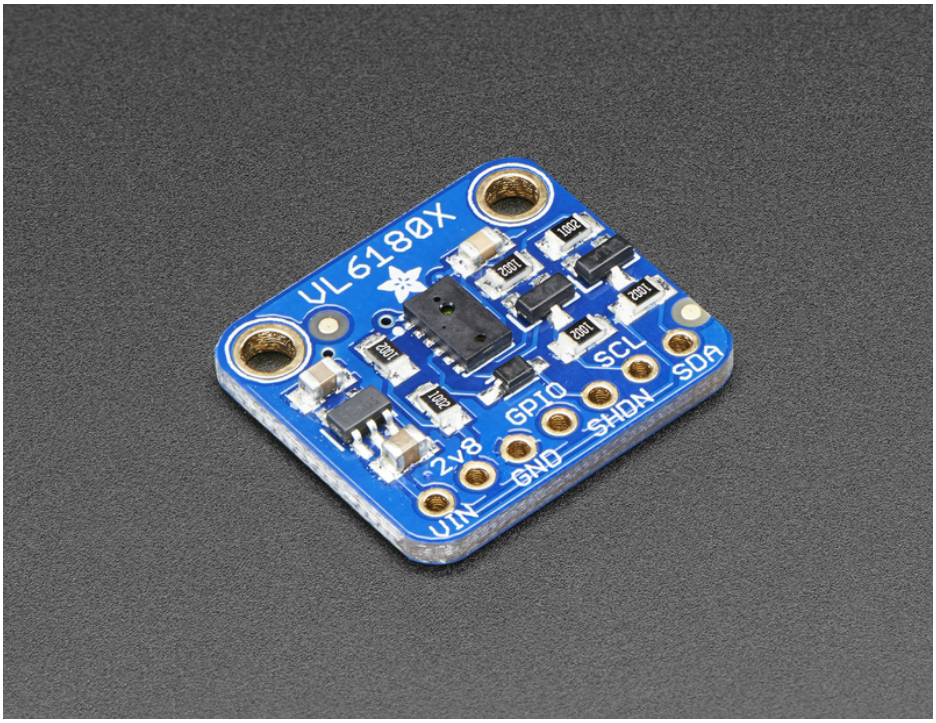
Overview



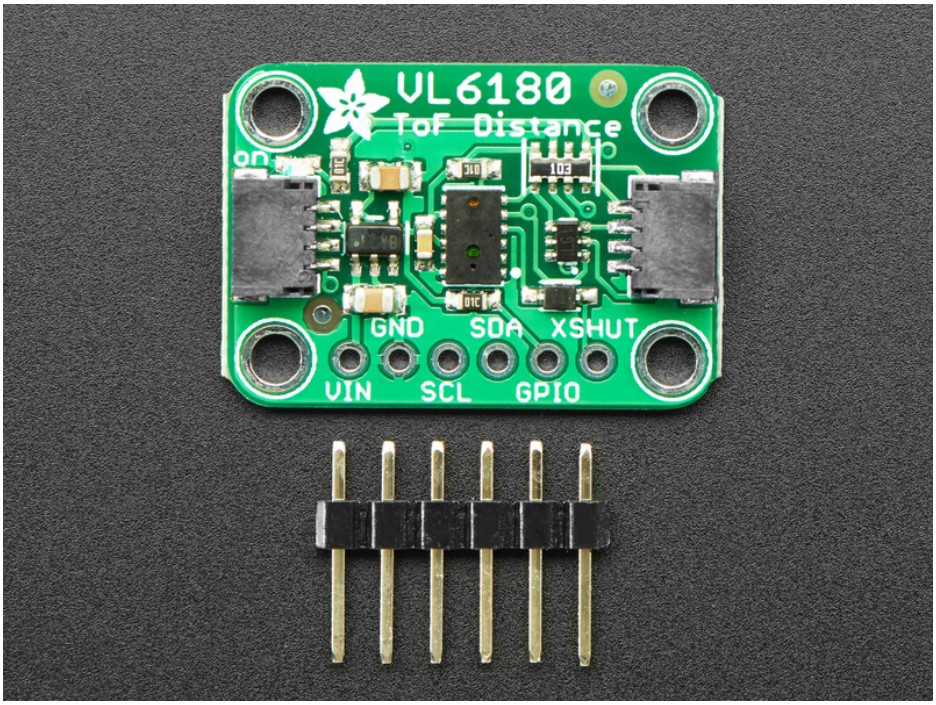
The VL6180X is a Time of Flight distance sensor like no other you've used! The sensor contains a very tiny invisible laser source, and a matching sensor. The VL6180X can detect the "time of flight", or how long the light has taken to bounce back to the sensor. Since it uses a very narrow light source, 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 VL6180X 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.



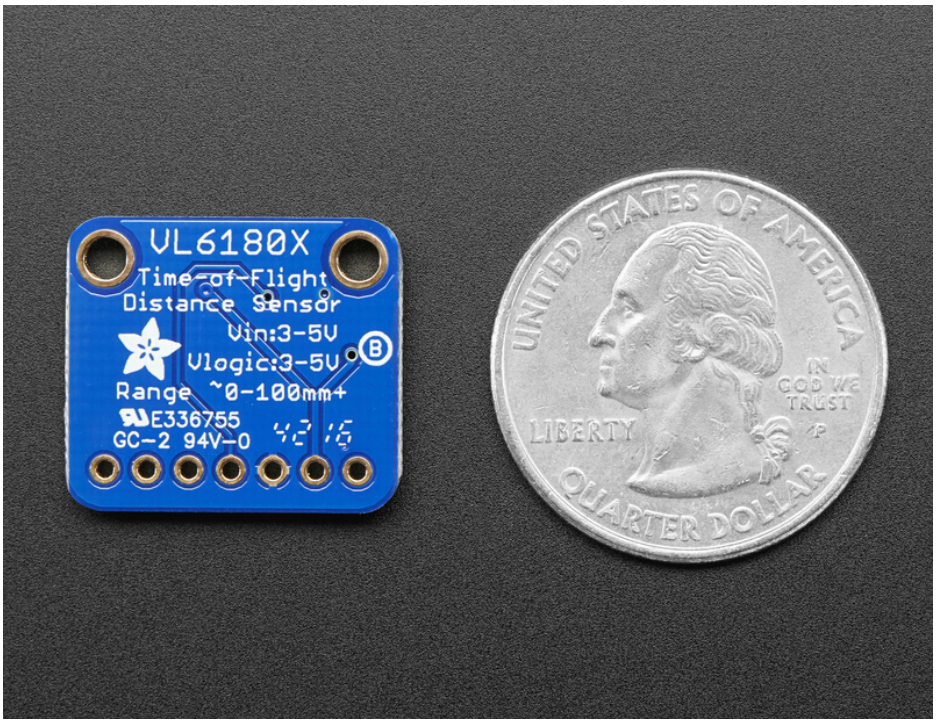
As if that weren't enough, we've also added SparkFun qwiic (https://adafru.it/Fpw) compatible **STEMMA QT (https://adafru.it/Ft4)** connectors for the I2C bus **so you don't even need to solder.** Just wire up to your favorite micro with a plug-and-play cable to get 6-DoF data ASAP. For a no-solder experience, just wire up to your favorite micro, like the STM32F405 Feather (https://adafru.it/Iqc) using a STEMMA QT adapter cable. (https://adafru.it/JnB) The Stemma QT connectors also mean the VL6180 can be used with our various associated accessories. (https://adafru.it/Ft6)



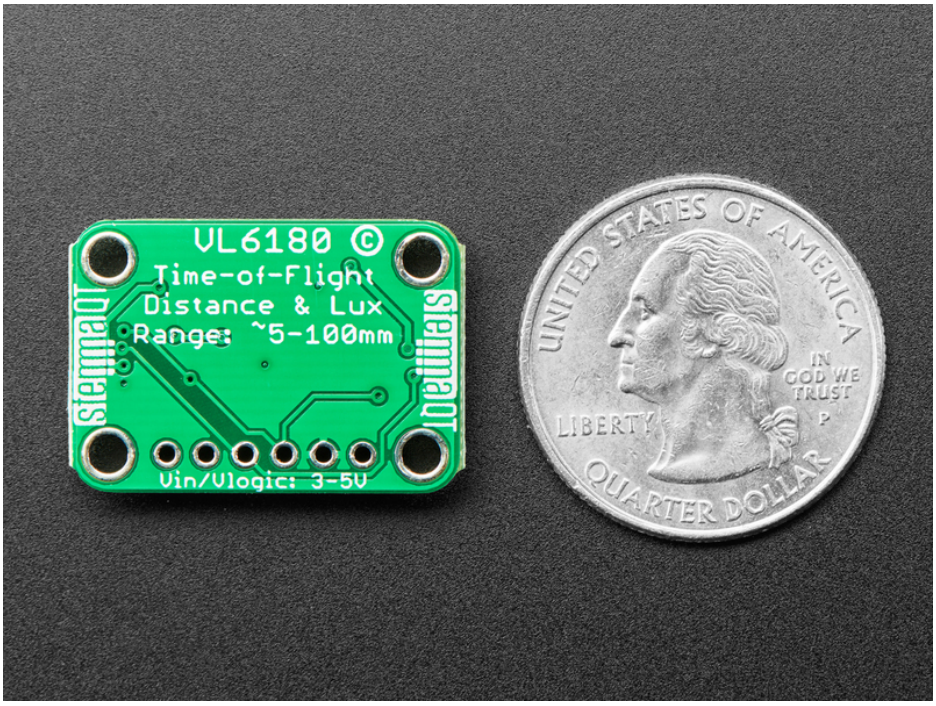
This is the 'little sister' of the VL53L0X ToF sensor, and can handle about 5mm to 200mm of range distance. It also includes a lux sensor. If you need a larger range, check out the VL53L0X which can measure 50 - 1200 mm.



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!

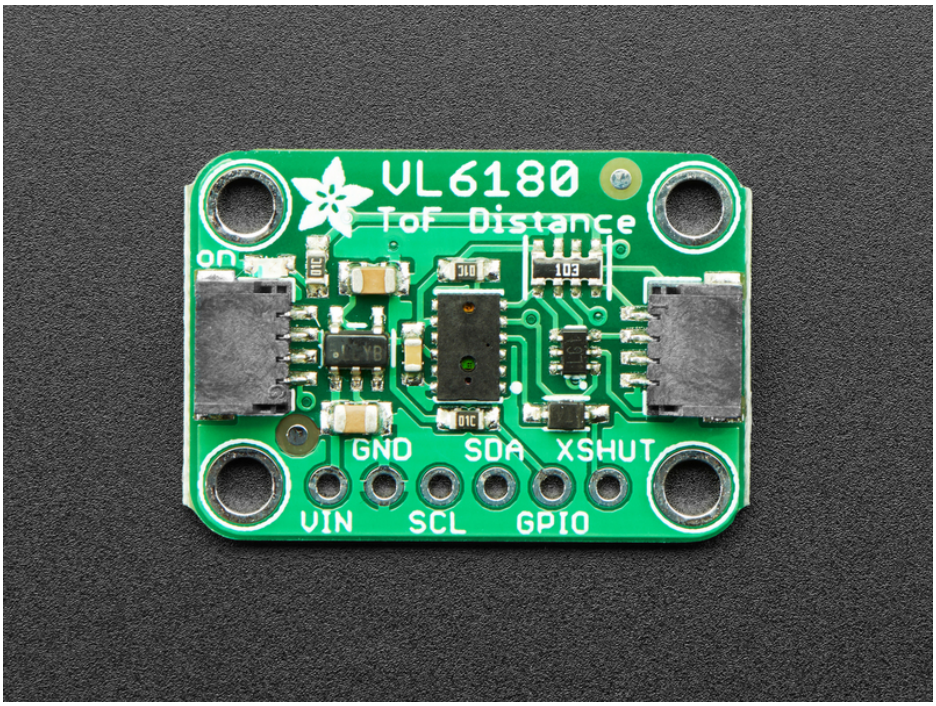


Communicating to the sensor is done over I2C with some simple commands. Most of the work is handled inside the sensor itself, so its very easy to port our Arduino library to another microcontroller.



Pinouts

The VL6180X 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 I2C address is **0x29** and you *can't* change it!



Power Pins:

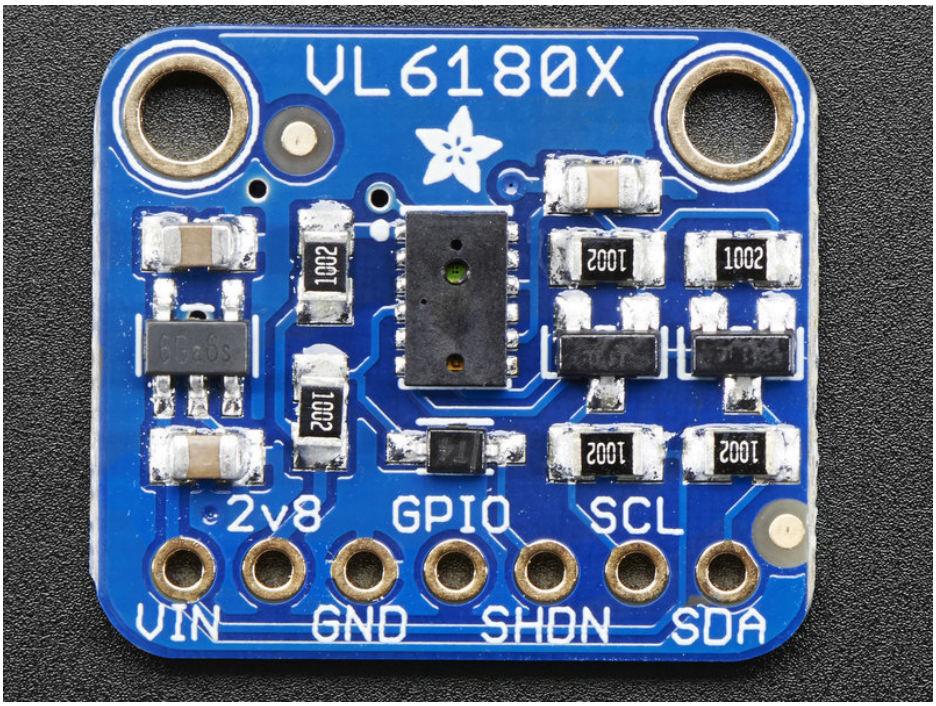
- **Vin** - this is the power pin. Since the chip uses 2.8 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- **2v8** - 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.
- **STEMMA QT (https://adafru.it/Ft4) -** These connectors allow you to connectors to dev boards with **STEMMA QT** connectors or to other things with various associated accessories (https://adafru.it/Ft6)

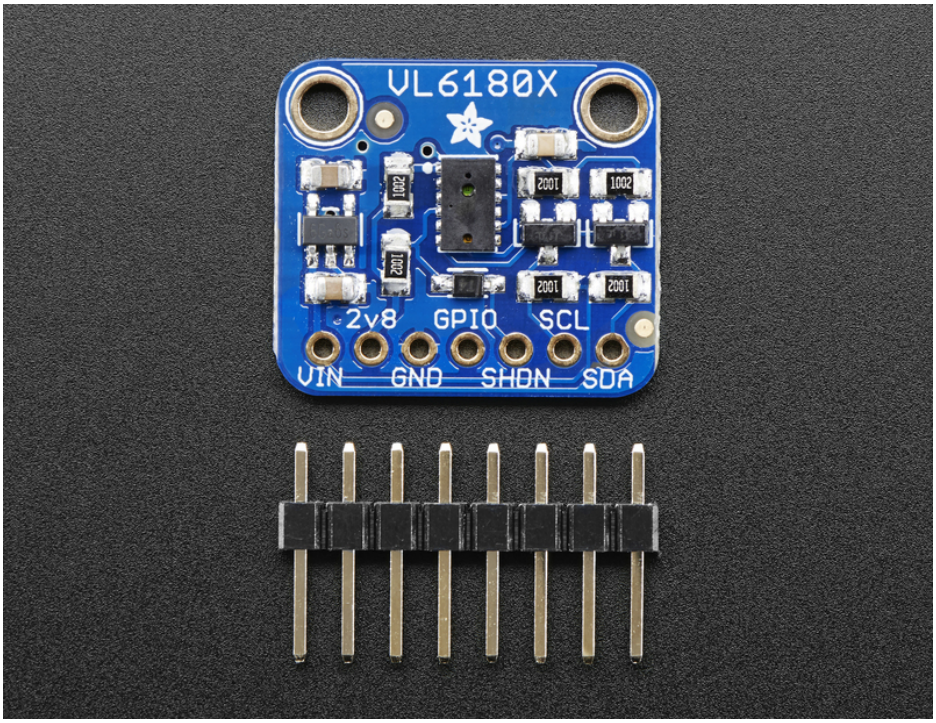
Control Pins:

- **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!
- **XSHUT/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.

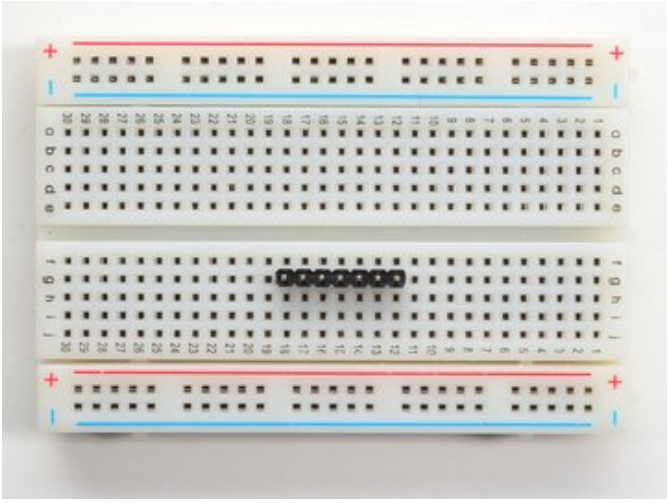

VL6180X
2v8
GPIO
SCL
VIN
GND
SHDN
SDA

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

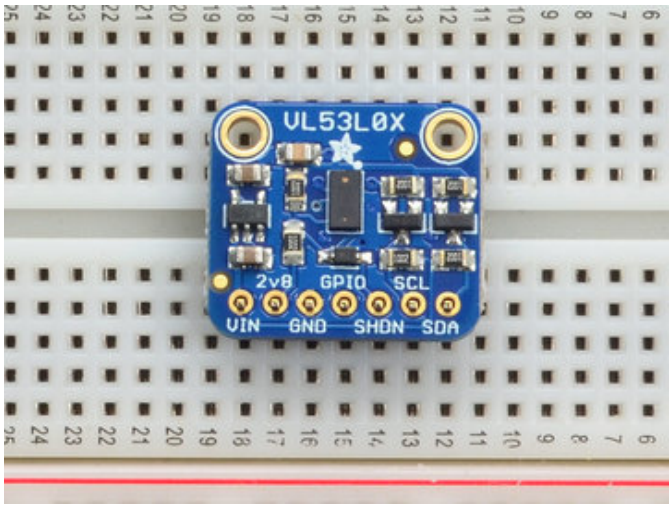


This page shows the VL53L0X or VL6180X sensor - the procedure is identical!

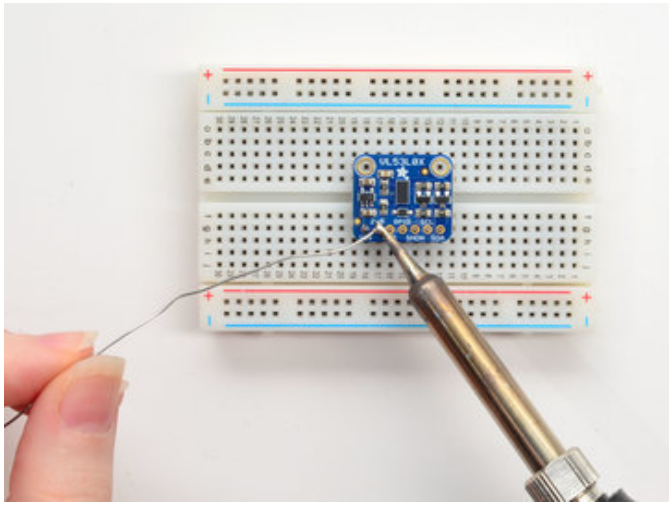


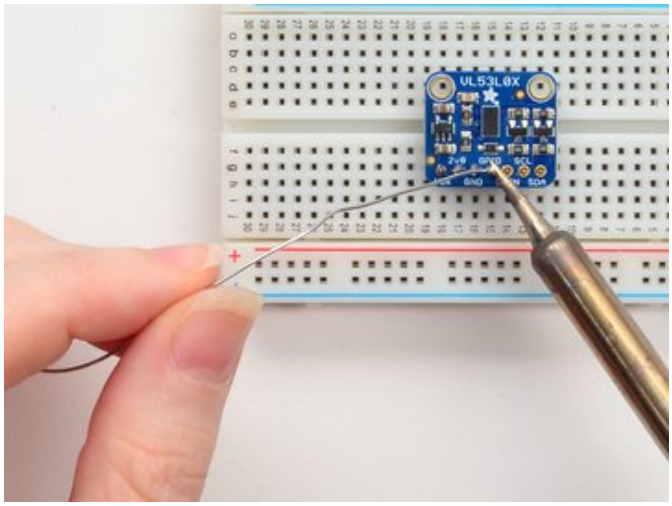
Prepare the header strip:

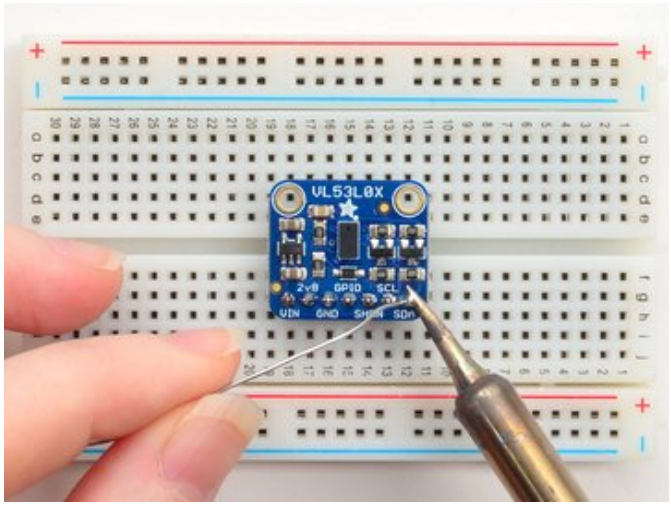
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**



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



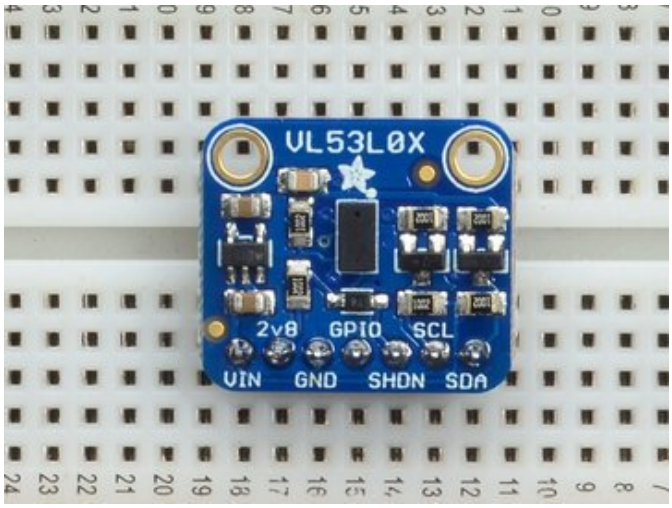




Solder!

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).



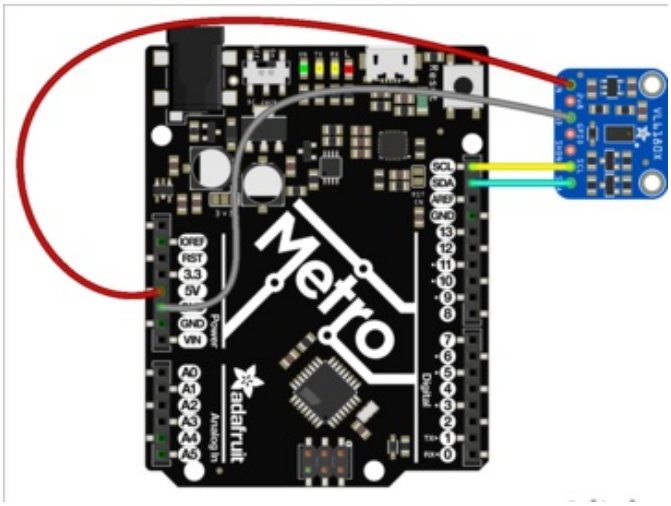
OK You're done! Check your work over and continue on to the next steps

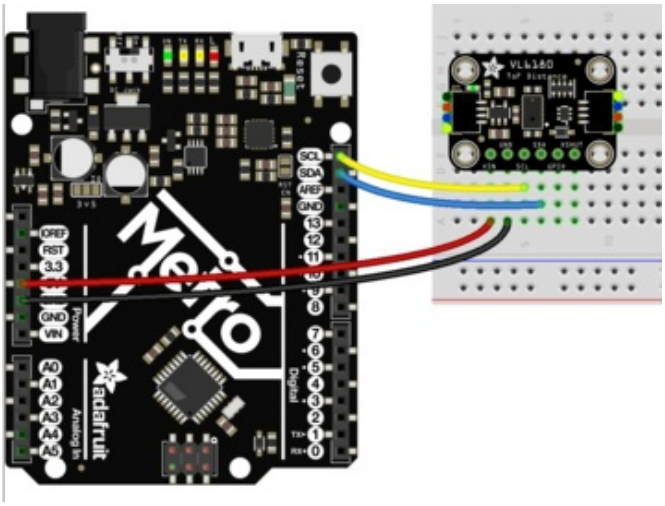
Arduino Code

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!

Wiring

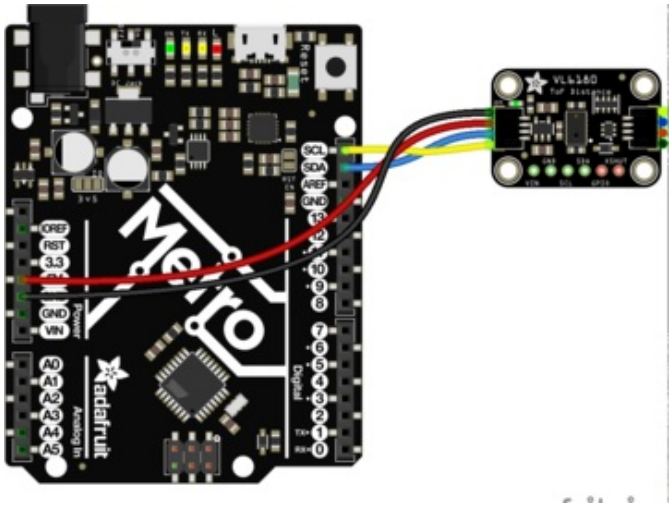
There's two versions of this sensor, and **the pin orders vary slightly so be sure to read the text on the breakout to match the pins to the wiring diagrams**.





- 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 **(red wire on STEMMA)**
- Connect **GND** to common power/data ground **(black wire on STEMMA)**
- 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 (yellow wire on STEMMA)**
- 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 (blue wire on STEMMA)**

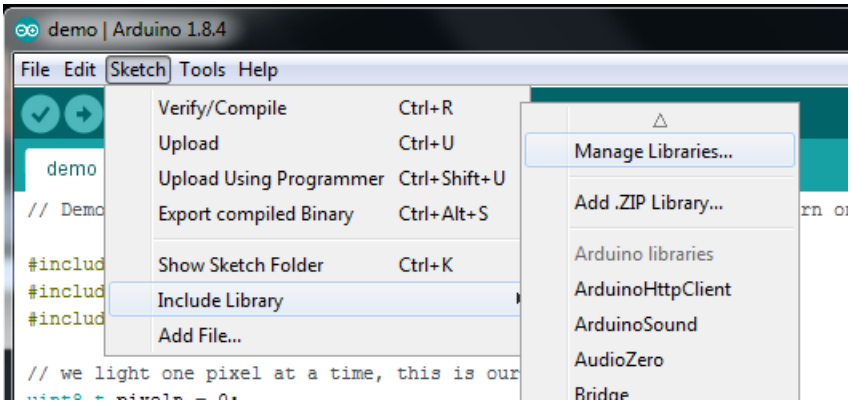
The VL6180X has a default I2C address of 0x29!



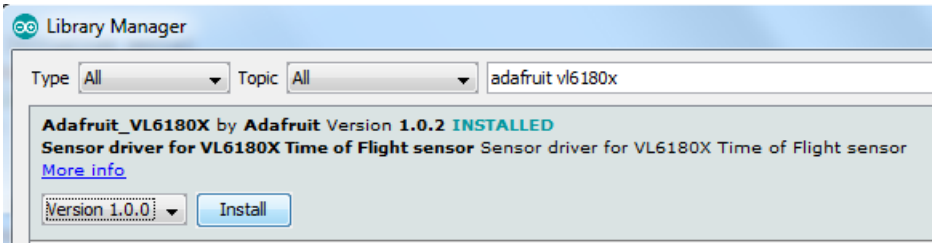
Install Adafruit_VL6180X

To begin reading sensor data, you will need to install Adafruit_VL6180X Library from our github repository (https://adafru.it/sld). It is available from the Arduino library manager so we recommend using that.

From the IDE open up the library manager...



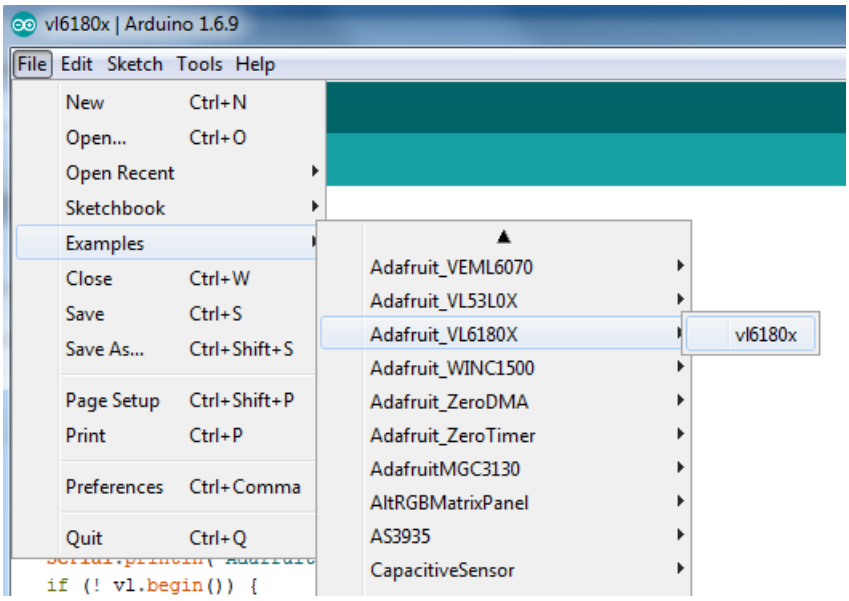
And type in **adafruit vl6180** to locate the library. Click **Install**



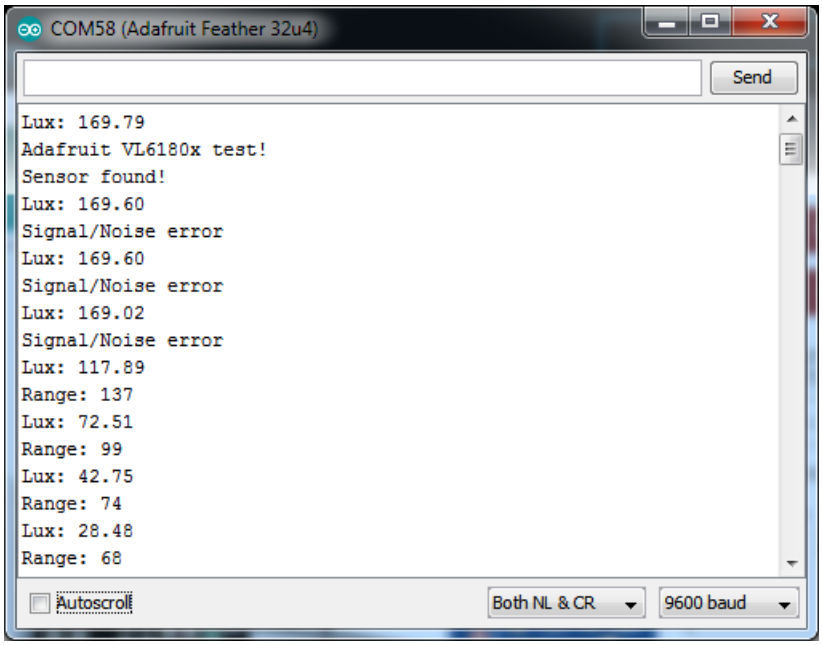
We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (https://adafru.it/aYM)

Load Demo

Open up **File->Examples->Adafruit_VL6180X->vl6180x** 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, the range readings are in millimeters and the light sensors in Lux. Note that when nothing is detected, it will print out the error which may vary.

Library Reference

You can start out by creating a Adafruit_VL6180X object using the default I2C:

```
Adafruit_VL6180X vl = Adafruit_VL6180X();
```

Once started, you can initialize the sensor which will also do some simple initializations and settings using **begin()**. Note that this will return True if the sensor was found and initialized. If not, it will return False.

```
vl.begin()
```

Reading Distance

Reading the range/distance is easy, just call **readRange()**

```
vl.readRange()
```

which will return the range in millimeters. If you get 0 or a value over 200 there's likely an error. Either way, before you trust that reading make sure to ask the sensor if the last reading had an error:

```
uint8_t status = vl.readRangeStatus()
```

The status will be **0** on no error, anything else is an error. Here's a simple code chunk that will decode the error!

```
  if  ((status >= VL6180X_ERROR_SYSERR_1) && (status <= VL6180X_ERROR_SYSERR_5)) {
    Serial.println("System error");
  }
  else if (status == VL6180X_ERROR_ECEFAIL) {
    Serial.println("ECE failure");
  }
  else if (status == VL6180X_ERROR_NOCONVERGE) {
    Serial.println("No convergence");
  }
  else if (status == VL6180X_ERROR_RANGEIGNORE) {
    Serial.println("Ignoring range");
  }
  else if (status == VL6180X_ERROR_SNR) {
    Serial.println("Signal/Noise error");
  }
  else if (status == VL6180X_ERROR_RAWUFLOW) {
    Serial.println("Raw reading underflow");
  }
  else if (status == VL6180X_ERROR_RAWOFLOW) {
    Serial.println("Raw reading overflow");
  }
  else if (status == VL6180X_ERROR_RANGEUFLOW) {
    Serial.println("Range reading underflow");
  }
  else if (status == VL6180X_ERROR_RANGEOFLOW) {
    Serial.println("Range reading overflow");
  }
```

Reading Light / Lux

You can also read a light reading from the sensor with

```
vl.readLux(GAIN)
```

which will return a semi-calibrated Lux reading. You can use different Gain settings to get a different range. For better results at low light, use higher gain. For better results at high light, use a lower gain.

Here's the gain's available:

- **VL6180X_ALS_GAIN_1** - gain of 1x
- **VL6180X_ALS_GAIN_1_25** - gain of 1.25x
- **VL6180X_ALS_GAIN_1_67** - gain of 1.67x
- **VL6180X_ALS_GAIN_2_5** - gain of 2.5x
- **VL6180X_ALS_GAIN_5** - gain of 5x
- **VL6180X_ALS_GAIN_10** - gain of 10x
- **VL6180X_ALS_GAIN_20** - gain of 20x
- **VL6180X_ALS_GAIN_40** - gain of 40x

We suggest starting with a gain of 5x and then adjusting up or down as necessary

Arduino Library Docs

Arduino Library Docs (https://adafru.it/Avp)

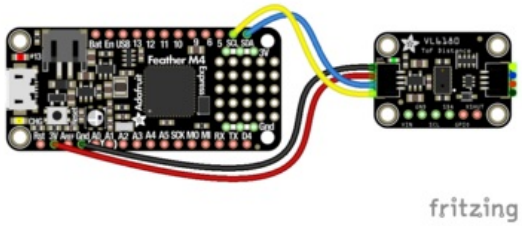
Python & CircuitPython

It's easy to use the VL6180X sensor with Python and CircuitPython, and the Adafruit CircuitPython VL6180X (https://adafru.it/C66) module. This module allows you to easily write Python code that reads the light and proximity readings from the sensor.

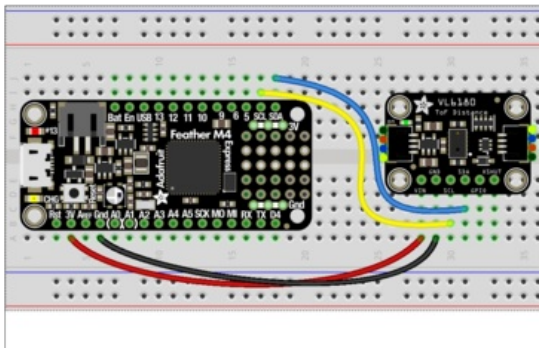
You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

First wire up a VL6180 breakout to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C:

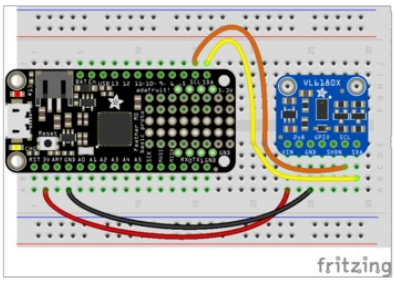


- **Board GND** to **sensor GND (black wire)**
- **Board 3V** to **sensor VIN (red wire)**
- **Board SDA** to **sensor SDA (blue wire)**
- **Board SCL** to **sensor SCL (yellow wire)**



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

Here's an example of wiring a Feather M0 to the sensor with I2C:

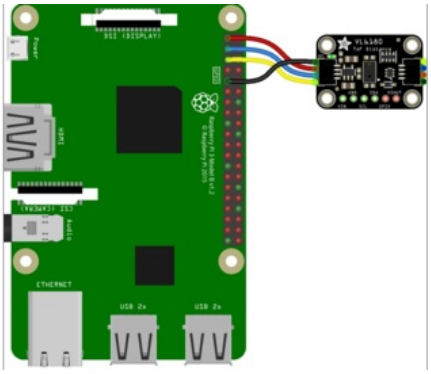


- **Board 3V** to **sensor VIN**
- **Board GND** to **sensor GND**
- **Board SCL** to **sensor SCL**
- **Board SDA** to **sensor SDA**

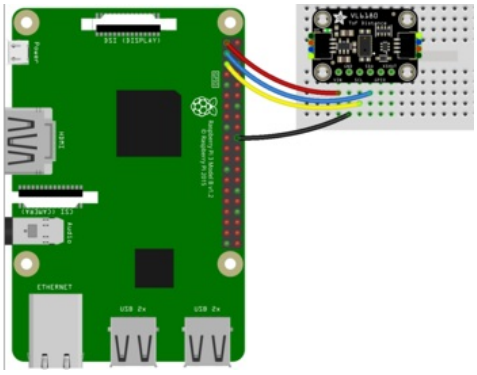
Python Computer Wiring

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).

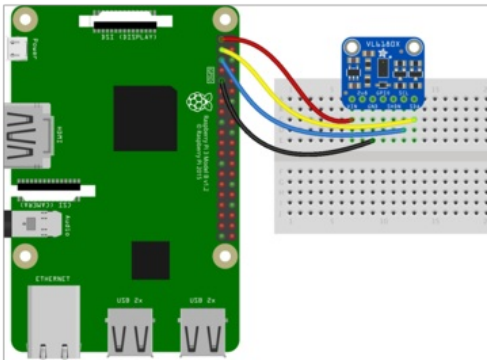
Here's the Raspberry Pi wired with I2C:



- **Pi GND** to **sensor GND (black wire)**
- **Pi 3V3** to **sensor VIN (red wire)**
- **Pi SDA** to **sensor SDA (blue wire)**
- **Pi SCL** to **sensor SCL (yellow wire)**



- **Pi GND** to **sensor GND (black wire)**
- **Pi 3V3** to **sensor VIN (red wire)**
- **Pi SDA** to **sensor SDA (blue wire)**
- **Pi SCL** to **sensor SCL (yellow wire)**



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

CircuitPython Installation of VL6180X Library

Next you'll need to install the Adafruit CircuitPython VL6180X (https://adafru.it/C66) 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_vl6180x.mpy**
- **adafruit_bus_device**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_vl6180x.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 VL6180X 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-vl6180x`

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 distance and more from the board's Python REPL. Run the following code to import the necessary modules to initialize the I2C bus and sensor:

```
import board
import busio
import adafruit_vl6180x
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_vl6180x.VL6180X(i2c)
```

Now you're ready to read the range and other values from the sensor. You can do so with a few properties and functions:

- **range** Property - This property returns the range as read from the sensor in millimeters.
- **range_status** Property - This property returns any error or issues that was encountered while reading the range. You can compare against a few global values to see what might have occured. Check the datasheet for more details about these error cases and how to resolve them:
    - `adafruit_vl6180x.ERROR_NONE`
    - `adafruit_vl6180x.ERROR_SYSERR_1`
    - `adafruit_vl6180x.ERROR_SYSERR_5`
    - `adafruit_vl6180x.ERROR_ECEFAIL`
    - `adafruit_vl6180x.ERROR_NOCONVERGE`
    - `adafruit_vl6180x.ERROR_RANGEIGNORE`
    - `adafruit_vl6180x.ERROR_SNR`
    - `adafruit_vl6180x.ERROR_RAWUFLOW`
    - `adafruit_vl6180x.ERROR_RAWOFLOW`
    - `adafruit_vl6180x.ERROR_RANGEUFLOW`
    - `adafruit_vl6180x.ERROR_RANGEOFLOW`
- **read_lux** Function - This function can be called to read the light read from the sensor and return it in lux. In order to call this function you must specify a gain that will be used during the reading. Check the datasheet for more details on how gain impacts the reading and which value you might want to use for your application. You can specify gain as one of these values:
    - `adafruit_vl6180x.ALS_GAIN_1` - 1x gain
    - `adafruit_vl6180x.ALS_GAIN_1_25` - 1.25x gain
    - `adafruit_vl6180x.ALS_GAIN_1_67` - 1.67x gain
    - `adafruit_vl6180x.ALS_GAIN_2_5` - 2.5x gain
    - `adafruit_vl6180x.ALS_GAIN_5` - 5x gain
    - `adafruit_vl6180x.ALS_GAIN_10` - 10x gain
    - `adafruit_vl6180x.ALS_GAIN_20` - 20x gain
    - `adafruit_vl6180x.ALS_GAIN_40` - 40x gain

```
print('Range: {0}mm'.format(sensor.range))
print('Range status: {0}'.format(sensor.range_status))
print('Light (1x gain): {0}lux'.format(sensor.read_lux(adafruit_vl6180x.ALS_GAIN_1)))
```

That's all there is to using the VL6180X distance sensor with CircuitPython!

Here's a complete example that will read the range and lux (using 1x gain) each second and print it out. Save this as **code.py** on your board and open the REPL to see the output.

Full Example Code

```
# Demo of reading the range and lux from the VL6180x distance sensor and
# printing it every second.
# Author: Tony DiCola
import time

import board
import busio

import adafruit_vl6180x


# Create I2C bus.
i2c = busio.I2C(board.SCL, board.SDA)

# Create sensor instance.
sensor = adafruit_vl6180x.VL6180X(i2c)

# Main loop prints the range and lux every second:
while True:
    # Read the range in millimeters and print it.
    range_mm = sensor.range
    print("Range: {0}mm".format(range_mm))
    # Read the light, note this requires specifying a gain value:
    # - adafruit_vl6180x.ALS_GAIN_1 = 1x
    # - adafruit_vl6180x.ALS_GAIN_1_25 = 1.25x
    # - adafruit_vl6180x.ALS_GAIN_1_67 = 1.67x
    # - adafruit_vl6180x.ALS_GAIN_2_5 = 2.5x
    # - adafruit_vl6180x.ALS_GAIN_5 = 5x
    # - adafruit_vl6180x.ALS_GAIN_10 = 10x
    # - adafruit_vl6180x.ALS_GAIN_20 = 20x
    # - adafruit_vl6180x.ALS_GAIN_40 = 40x
    light_lux = sensor.read_lux(adafruit_vl6180x.ALS_GAIN_1)
    print("Light (1x gain): {0}lux".format(light_lux))
    # Delay for a second.
    time.sleep(1.0)
```

Python Docs

Python Docs (https://adafru.it/C7w)

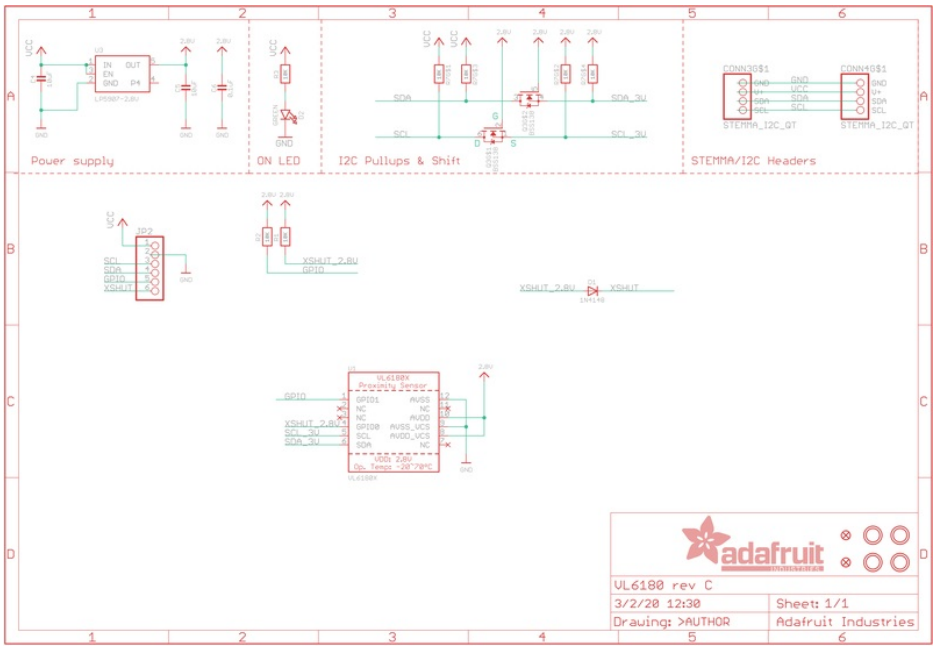
Downloads

Files & Datasheets

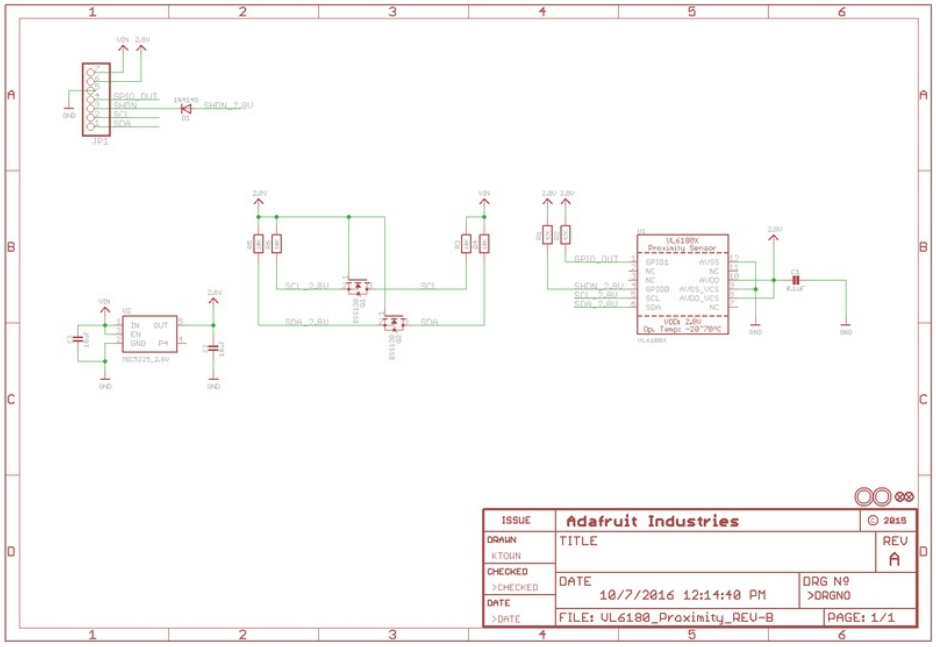
- Datasheet for the VL6180X (https://adafru.it/sIa)
- ST product page (https://adafru.it/sIb) with more details including API downloads
- Fritzing object in Adafruit Fritzing library (https://adafru.it/aP3)
- STEMMA revision Fritzing object in Adafruit Fritzing library (https://adafru.it/JRD)
- EagleCAD PCB files in GitHub (https://adafru.it/sIc)

Schematic & Fabrication Print

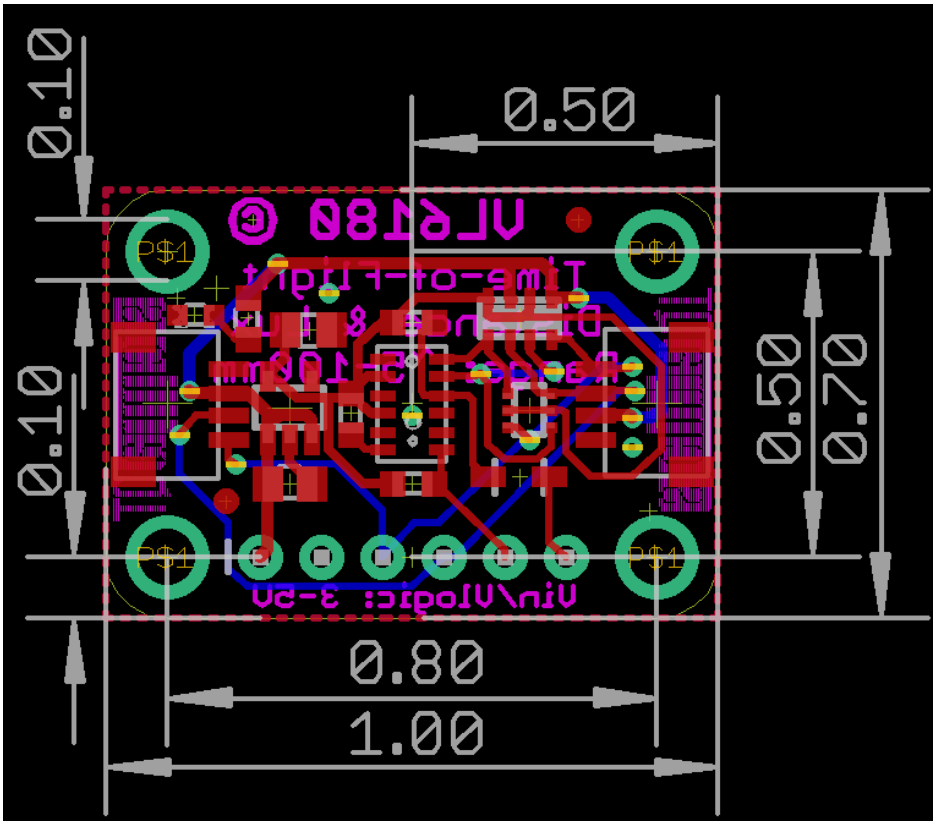
STEMMA revision:



Original:

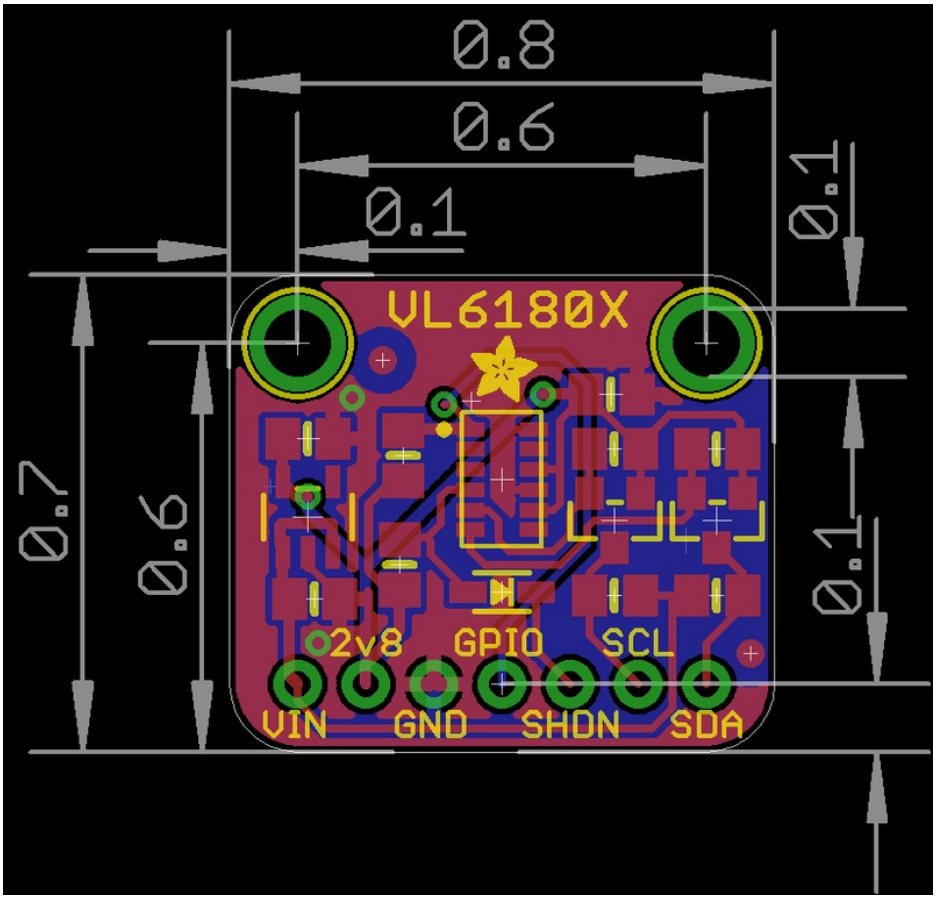


STEMMA revision:



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

Original:


0.8
0.6
0.1
0.1
VL6180X
0.7
0.6
0.1
2v8
GPIO
SCL
VIN
GND
SHDN
SDA