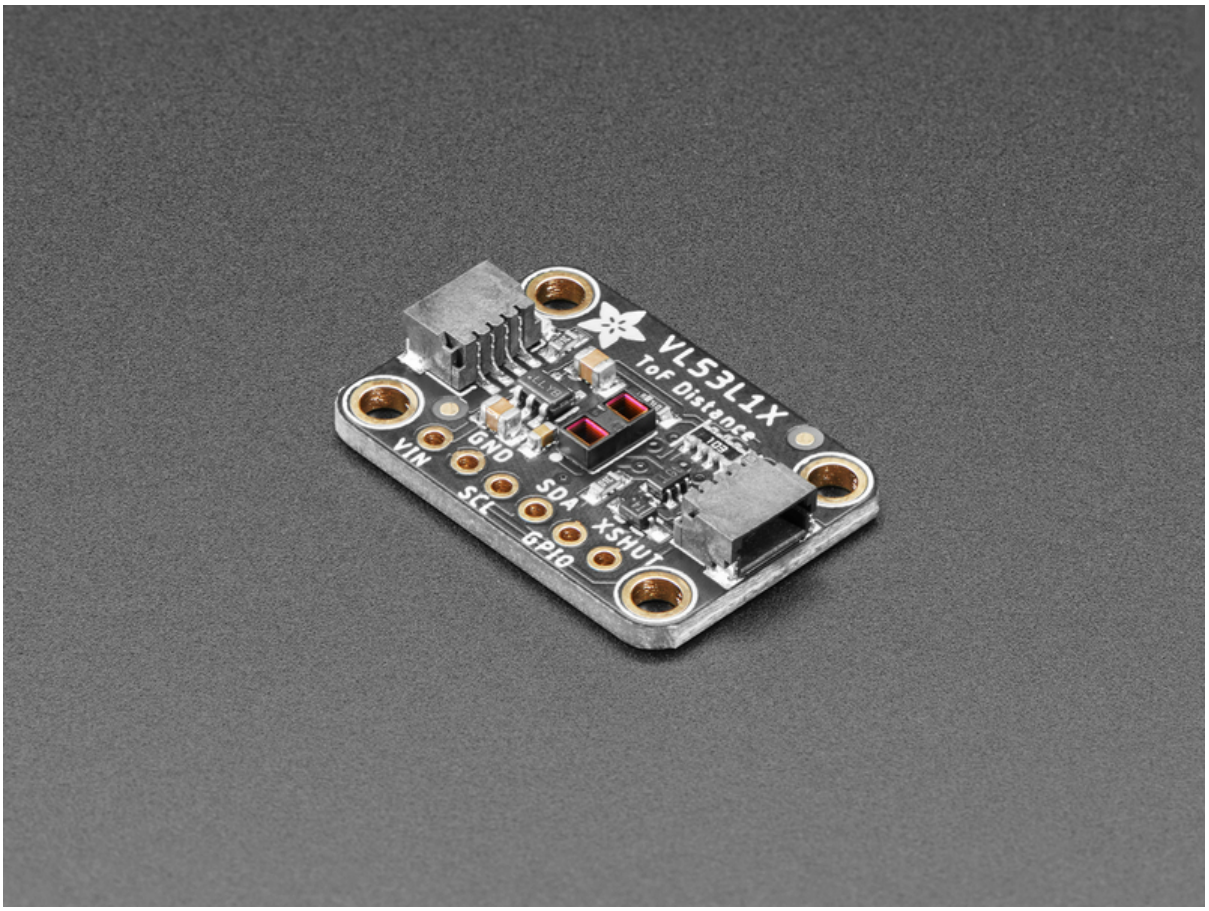
Adafruit VL53L1X Time of Flight Distance Sensor

Created by Kattni Rembor

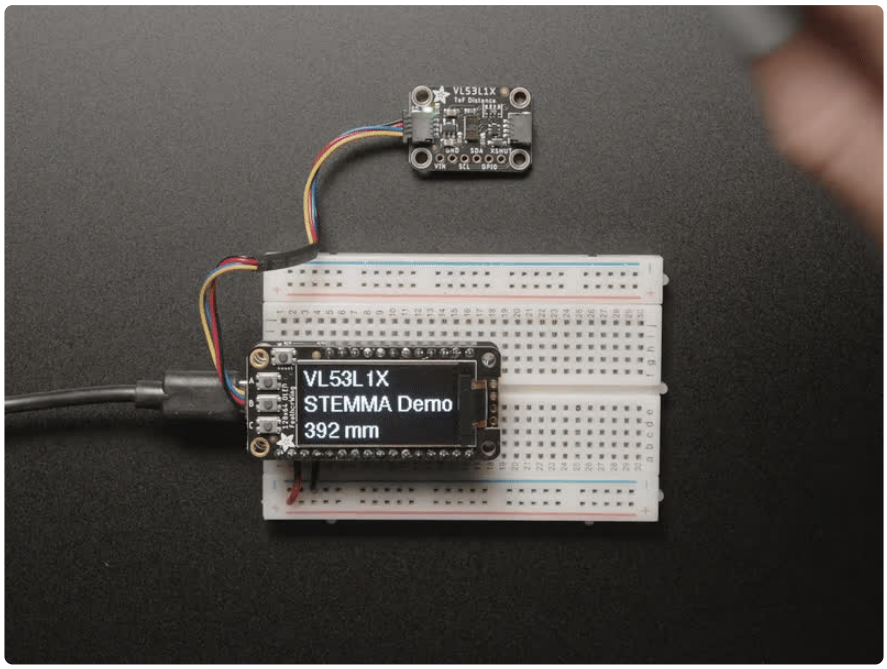


https://learn.adafruit.com/adafruit-vl53l1x

Last updated on 2024-06-03 03:29:49 PM EDT

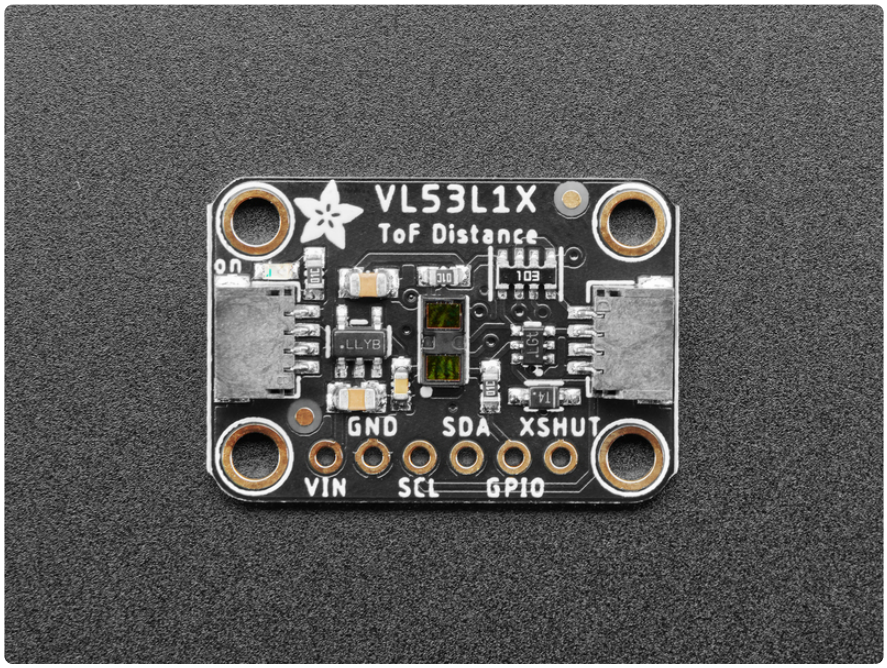
Table of Contents

Overview

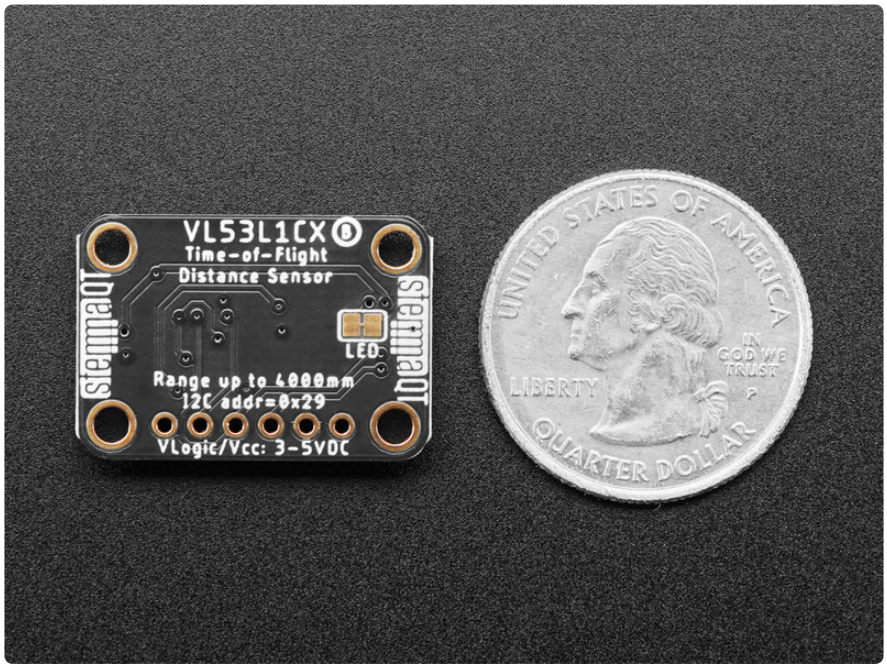


The **Adafruit VL53L1X Time of Flight Distance Sensor** (also known as VL53L1CX) is a Time of Flight distance sensor that has a massive 4 meter range and LIDAR-like precision. The sensor contains a very tiny invisible laser source and a matching sensor. The VL53L1X can detect the "time of flight", or how long the light has taken to bounce back to the sensor.

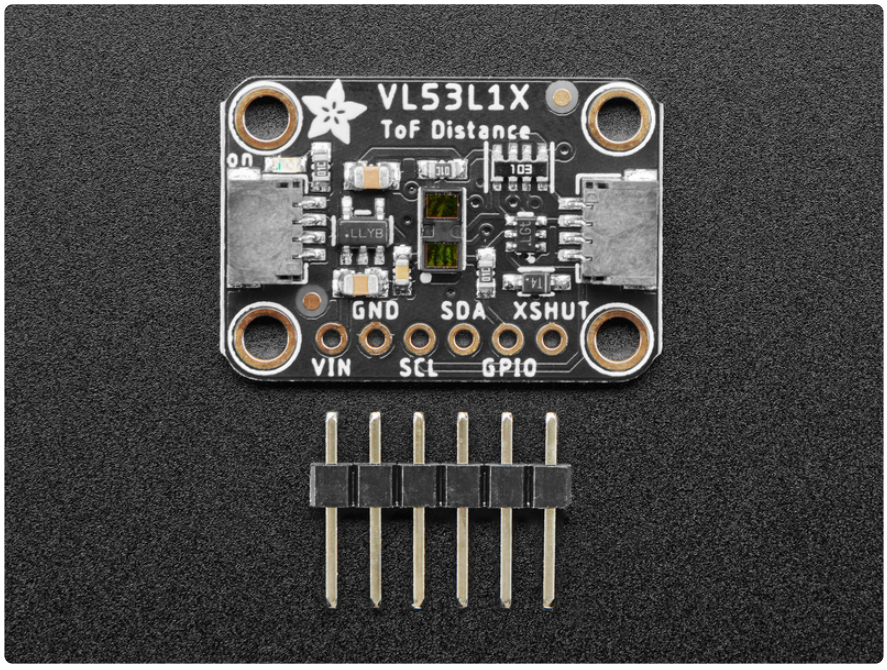
This breakout ships with a protector over the sensor. It must be removed before use! See details below.



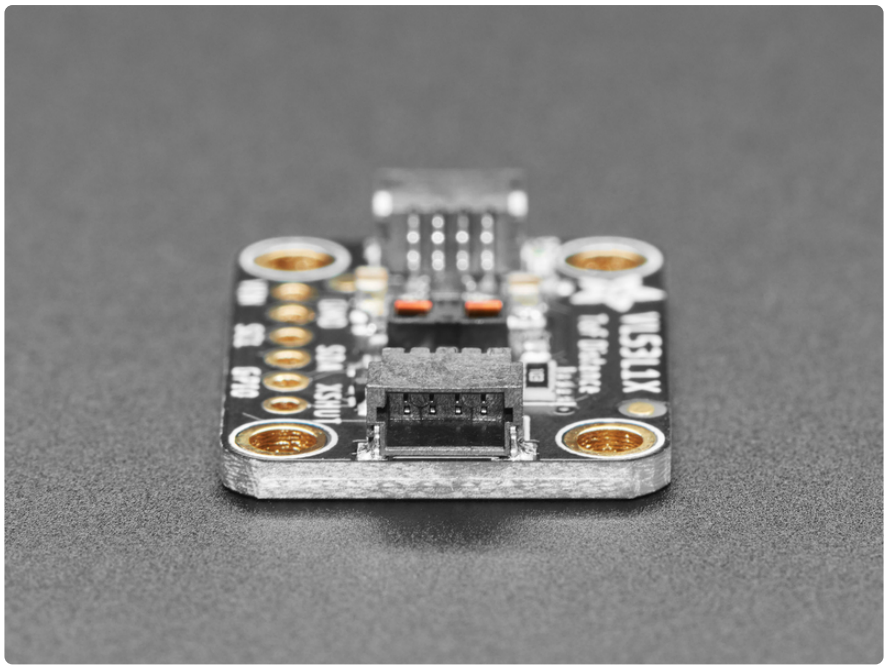
Since the sensor uses a very narrow light source, it is good for determining the distance of only the surface directly in front of it. Unlike sonar that bounces ultrasonic waves, the 'cone' of sensing is very narrow. Unlike IR distance sensors that try to measure the amount of light bounced, the VL53L1X 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 'next generation' of the VL53L0X ToF sensor (http://adafru.it/3317) and can handle about **~30 to 4000mm** of range distance, with up to 50Hz update rate. If you need an even smaller/closer range, check out the VL6180X (http://adafru.it/3316) which can measure 5mm to 200mm and also contains a light sensor.



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. Works great with the **3.3V logic level of a Feather or Raspberry Pi, or the 5V level of a Metro 328 or Arduino Uno**, this breakout is ready to work with most common microcontrollers or SBCs. and since it speaks I2C, you can easily connect it up with two data wires plus power and ground.



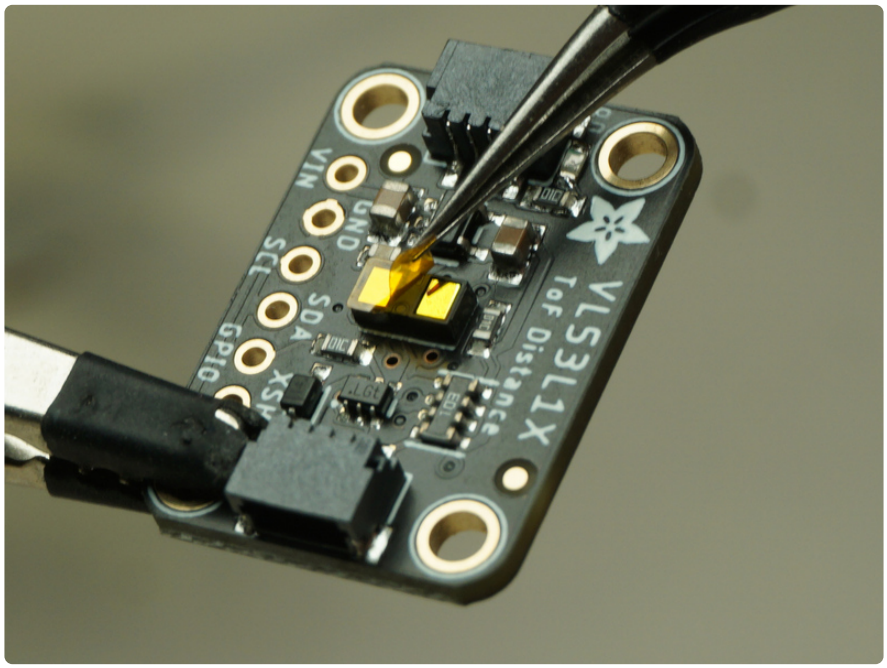
As if that weren't enough, we've also added SparkFun qwiic (https://adafru.it/Fpw) compatible STEMMA QT **(https://adafru.it/Ft4)** connectors for the I2C bus **so you don't even need to solder.** Just wire up to your favorite micro with a plug-and-play cable to get ToF data ASAP. For a no-solder experience, just wire up to your favorite micro, like the STM32F405 Feather (http://adafru.it/4382) using a STEMMA QT adapter cable. (https://adafru.it/JnB) The Stemma QT connectors also mean the VL53L1X can be used with our various associated accessories. (https://adafru.it/Ft6) QT Cable is not included, but we have a variety in the shop (https://adafru.it/17VE)

Communicating to the sensor is done over I2C with an API written by ST, so it's 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 (https://adafru.it/VA4).

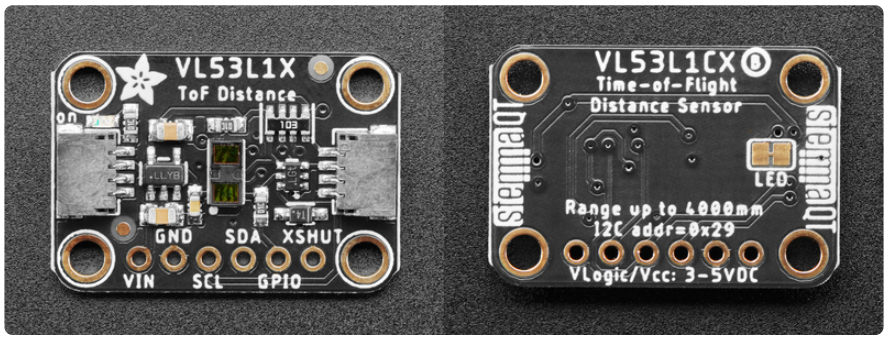
Removing the Protective Tape

Be careful when removing the tape! You don't want to damage the sensor.

Using tweezers (or some other appropriate tool), CAREFULLY remove the protective tape over the sensor, as seen in the image below. There is a small tab on the side of the tape that you can use to remove it.



Pinouts



This breakout ships with a protector over the sensor. It must be removed before use! See details at the bottom of the Overview page.

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 3V microcontroller like a Feather M4, use 3V, or for a 5V microcontroller like Arduino, use 5V.
- **GND** - This is common ground for power and logic.

I2C Logic Pins

The default I2C address for the VL53L1X is **0x29**.

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. There's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller I2C data line. There's a **10K pullup** on this pin.
- **STEMMA QT** (https://adafru.it/Ft4) - These connectors allow you to connect to development boards with **STEMMA QT** connectors or to other things with various associated accessories (https://adafru.it/JRA).

Other Pins

- **GPIO** - This is the interrupt output pin, it is 2.8V logic level output - it can be read by 3.3V and most 5V logic microcontrollers
- **XSHUT** - This is the shutdown pin. It is active low, and is logic-level shifted so you can use 3V or 5V logic.

LED Jumper

- **LED jumper** - This jumper is located on the back of the board. Cut the trace on this jumper to cut power to the "on" LED.

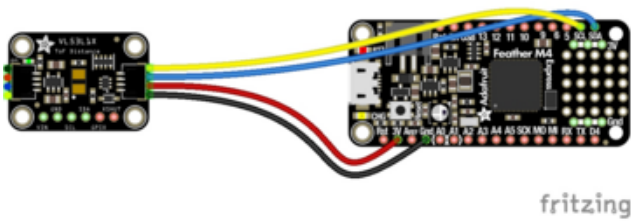
Python & CircuitPython

It's easy to use the **VL53L1X** with Python or CircuitPython, and the Adafruit CircuitPython VL53L1X (https://adafru.it/VA6) module. This module allows you to easily write Python code that reads the distance from the **VL53L1X** 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).

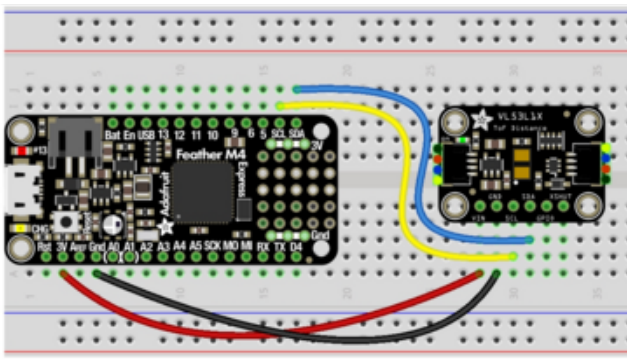
CircuitPython Microcontroller Wiring

First wire up a VL53L1X to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using one of the handy **STEMMA QT** (https://adafru.it/Ft4) 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 use the standard **0.100" pitch** headers to wire it up on a breadboard:

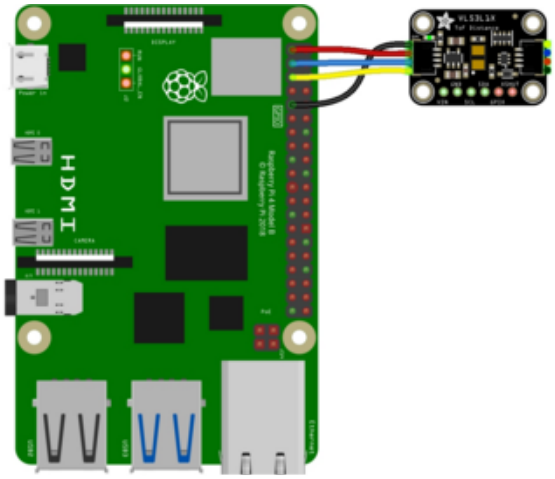


Board 3V to **sensor VIN (red wire)**
Board GND to **sensor GND (black wire)**
Board SCL to **sensor SCL (yellow wire)**
Board SDA to **sensor SDA (blue wire)**

Python Computer Wiring

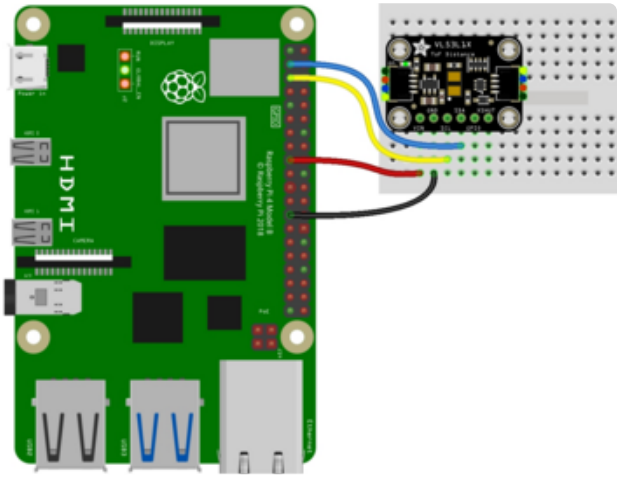
Since there's dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired to the sensor using I2C and a **STEMMA QT** (https://adafru.it/Ft4) 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)**

Finally here is an example of how to wire up a Raspberry Pi to the sensor 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 VL53L1X 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:

- `pip3 install adafruit-circuitpython-vl53l1x`

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!

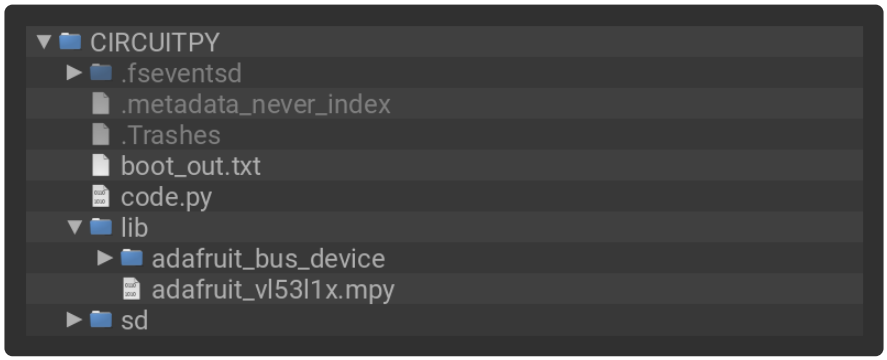
CircuitPython Usage

To use with CircuitPython, you need to first install the VL53L1X 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 folder and file:

- **adafruit_bus_device/**
- **adafruit_vl53l1x.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:

```
python3 code.py
```

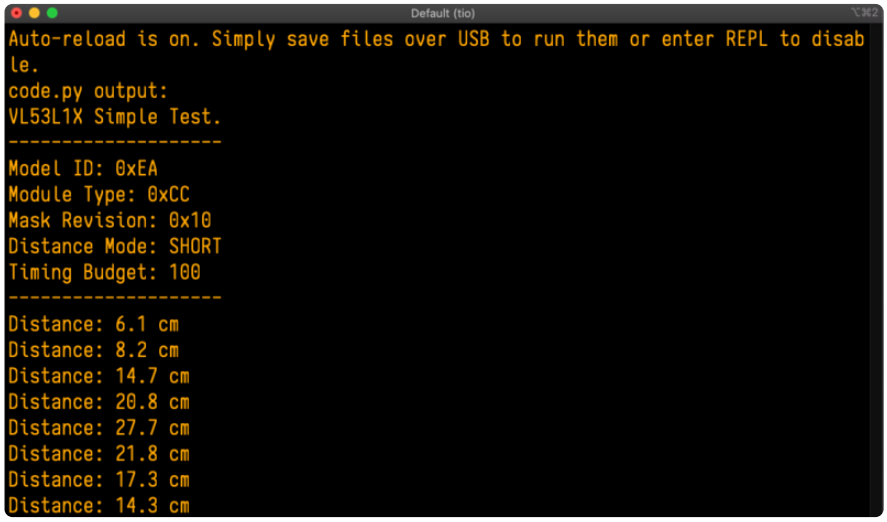
Example Code

```
# SPDX-FileCopyrightText: 2017 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Carter Nelson for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

# Simple demo of the VL53L1X distance sensor.
# Will print the sensed range/distance every second.
```

```
import time
import board
import adafruit_vl53l1x

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

vl53 = adafruit_vl53l1x.VL53L1X(i2c)

# OPTIONAL: can set non-default values
vl53.distance_mode = 1
vl53.timing_budget = 100

print("VL53L1X Simple Test.")
print("--------------------")
model_id, module_type, mask_rev = vl53.model_info
print("Model ID: 0x{:0X}".format(model_id))
print("Module Type: 0x{:0X}".format(module_type))
print("Mask Revision: 0x{:0X}".format(mask_rev))
print("Distance Mode: ", end="")
if vl53.distance_mode == 1:
    print("SHORT")
elif vl53.distance_mode == 2:
    print("LONG")
else:
    print("UNKNOWN")
print("Timing Budget: {}".format(vl53.timing_budget))
print("--------------------")

vl53.start_ranging()

while True:
    if vl53.data_ready:
        print("Distance: {} cm".format(vl53.distance))
        vl53.clear_interrupt()
        time.sleep(1.0)
```

If running CircuitPython: Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!

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



Now try holding your hand in front of the sensor, and moving it closer and further away to see the values change!

First you import the necessary modules and libraries. Then you instantiate the sensor on I2C.

Then you're ready to read data from the sensor, including the initial information printed to the serial console.

Finally, inside the loop, you check the distance every second.

That's all there is to using the VL53L1X with CircuitPython!

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 VL53L1X is **0x29** but you can change this in software.

To set the new address, you need to use `set_address`. 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.

The following example shows how to use `set_address` to change the address on additional sensors plugged into the same I2C bus. It sets the addresses, prints the addresses in use, and then displays the distance data to the serial console. It is written for two sensors, but is easily modifiable to accommodate more.

```
# SPDX-FileCopyrightText: 2022 wrdaigle for Adafruit Industries
# SPDX-FileCopyrightText: 2022 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT
"""
VL53L1X multiple sensor I2C set_address demo.

This example is written for two sensors, but it can easily be modified to include
more.

NOTE: A multitude of sensors may require more current than the on-board 3V
regulator can output.
The typical current consumption during active range readings is about 19 mA per
sensor.
"""

import time
import board
import digitalio
import adafruit_vl53l1x

# Define the I2C pins.
i2c = board.I2C()  # uses board.SCL and board.SDA
```

```
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller

xshut = [
    # Update the D6 and D5 pins to match the pins to which you wired your sensor
XSHUT pins.
    digitalio.DigitalInOut(board.D6),
    digitalio.DigitalInOut(board.D5),
    # Add more VL53L1X sensors by defining their XSHUT pins here.
]

for shutdown_pin in xshut:
    # Set the shutdown pins to output, and pull them low.
    shutdown_pin.switch_to_output(value=False)
    # These pins are active when Low, meaning:
    #   If the output signal is LOW, then the VL53L1X sensor is off.
    #   If the output signal is HIGH, then the VL53L1X sensor is on.
# All VL53L1X sensors are now off.

# Create a list to be used for the array of VL53L1X sensors.
vl53l1x = []

# Change the address of the additional VL53L1X sensors.
for pin_number, shutdown_pin in enumerate(xshut):
    # Turn on the VL53L1X sensors to allow hardware check.
    shutdown_pin.value = True
    # Instantiate the VL53L1X I2C object and insert it into the vl53l1x list.
    # This also performs VL53L1X hardware check.
    sensor_i2c = adafruit_vl53l1x.VL53L1X(i2c)
    vl53l1x.append(sensor_i2c)
    # This ensures no address change on one sensor board, specifically the last one
in the series.
    if pin_number < len(xshut) - 1:
        # The default address is 0x29. Update it to an address that is not already
in use.
        sensor_i2c.set_address(pin_number + 0x30)

# Print the various sensor I2C addresses to the serial console.
if i2c.try_lock():
    print("Sensor I2C addresses:", [hex(x) for x in i2c.scan()])
    i2c.unlock()

# Start ranging for sensor data collection.
for sensor in vl53l1x:
    sensor.start_ranging()
while True:
    # Extract the appropriate data from the current list, and print
    # the sensor distance readings for all available sensors.
    for sensor_number, sensor in enumerate(vl53l1x):
        if sensor.data_ready:
            print("Sensor {}: {}".format(sensor_number + 1, sensor.distance))
            sensor.clear_interrupt()
    time.sleep(0.5)
```

That's all there is to setting the address of an additional sensor using CircuitPython!

Python Docs

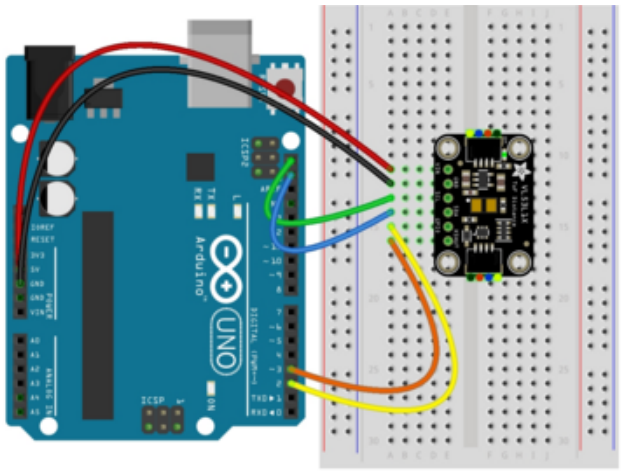
Python Docs (https://adafru.it/VA7)

Arduino

Using the VL53L1X with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit VL53L1X (https://adafru.it/VA4) library and running the provided example code.

Wiring

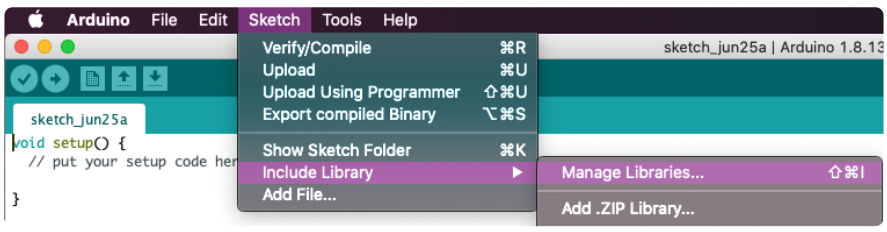
Wire as shown for a **5V** board like an UNO. If you are using a **3V** board, like an Adafruit Metro, wire the board's 3V pin to the VL53L1X Vin.



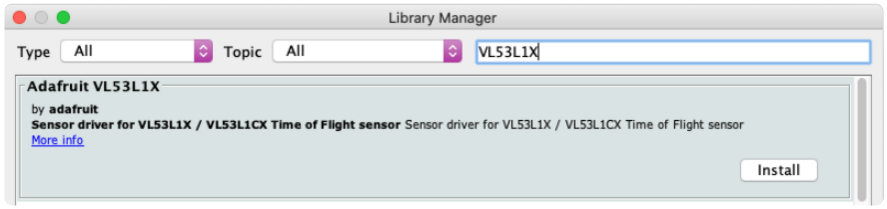
Board 5V to **sensor Vin**
Board GND to **sensor GND**
Board SCL to **sensor SCL**
Board SDA to **sensor SDA**
Board 2 to **sensor GPIO**
Board 3 to **sensor XSHUT**

Library Installation

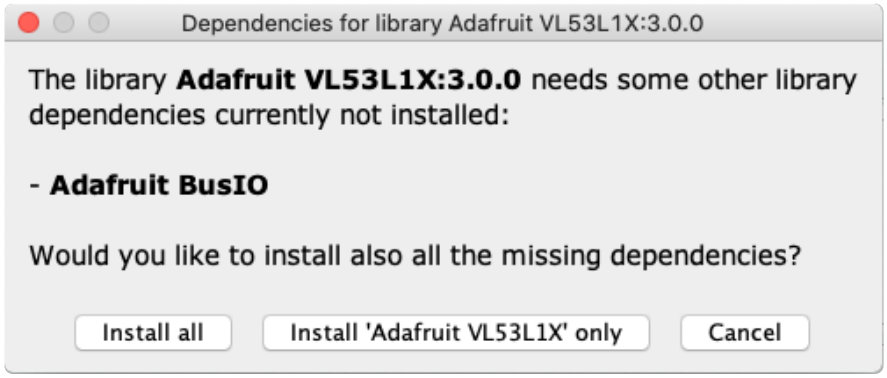
You can install the **VL53L1X** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **VL53L1X** , and select the **Adafruit VL53L1X** library:



If asked about dependencies, click "Install all".



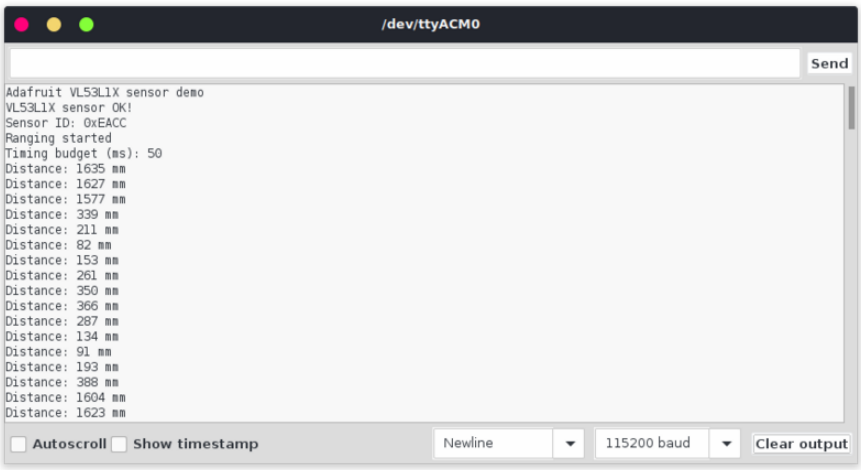
Load Example

Open up **File -> Examples -> Adafruit VL53L1X -> VL53L1X_simpletest** and upload to your Arduino wired to the sensor.

```
#include "Adafruit_VL53L1X.h"

#define IRQ_PIN 2
#define XSHUT_PIN 3

Adafruit_VL53L1X vl53 = Adafruit_VL53L1X(XSHUT_PIN, IRQ_PIN);

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10);

  Serial.println(F("Adafruit VL53L1X sensor demo"));

  Wire.begin();
  if (! vl53.begin(0x29, &Wire)) {
    Serial.print(F("Error on init of VL sensor: "));
    Serial.println(vl53.vl_status);
    while (1)       delay(10);
  }
  Serial.println(F("VL53L1X sensor OK!"));

  Serial.print(F("Sensor ID: 0x"));
  Serial.println(vl53.sensorID(), HEX);

  if (! vl53.startRanging()) {
    Serial.print(F("Couldn't start ranging: "));
    Serial.println(vl53.vl_status);
    while (1)       delay(10);
  }
  Serial.println(F("Ranging started"));

  // Valid timing budgets: 15, 20, 33, 50, 100, 200 and 500ms!
  vl53.setTimingBudget(50);
  Serial.print(F("Timing budget (ms): "));
  Serial.println(vl53.getTimingBudget());

  /*
  vl.VL53L1X_SetDistanceThreshold(100, 300, 3, 1);
  vl.VL53L1X_SetInterruptPolarity(0);
  */
}
```

```
void loop() {
  int16_t distance;

  if (vl53.dataReady()) {
    // new measurement for the taking!
    distance = vl53.distance();
    if (distance == -1) {
      // something went wrong!
      Serial.print(F("Couldn't get distance: "));
      Serial.println(vl53.vl_status);
      return;
    }
    Serial.print(F("Distance: "));
    Serial.print(distance);
    Serial.println(" mm");

    // data is read out, time for another reading!
    vl53.clearInterrupt();
  }
}
```

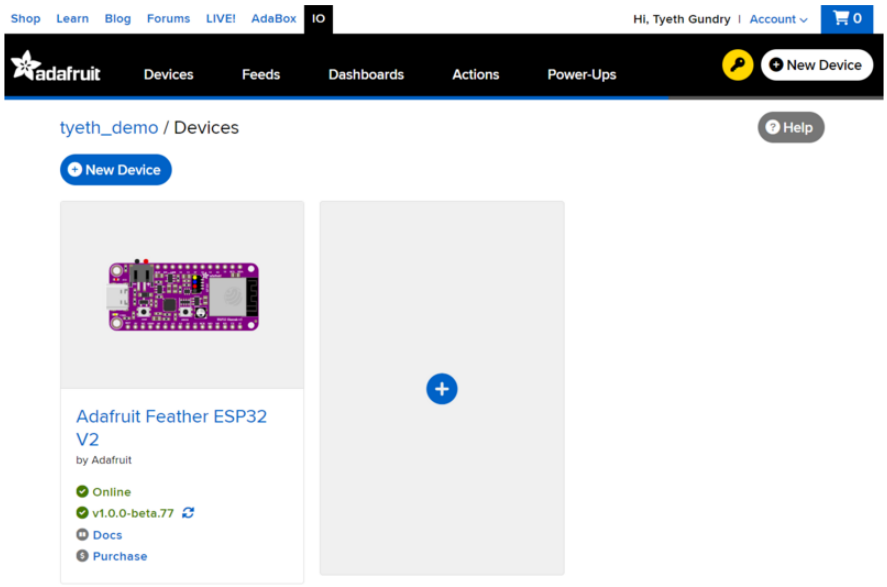
Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You should see the the values from the senor being printed out.



Arduino Docs

Arduino Docs (https://adafru.it/VBt)

WipperSnapper



What is WipperSnapper

WipperSnapper is a firmware designed to turn any WiFi-capable board into an Internet-of-Things device without programming a single line of code. WipperSnapper connects to Adafruit IO (https://adafru.it/fsU), a web platform designed (by Adafruit! (https://adafru.it/Bo5)) to display, respond, and interact with your project's data.

Simply load the WipperSnapper firmware onto your board, add credentials, and plug it into power. Your board will automatically register itself with your Adafruit IO account.

From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

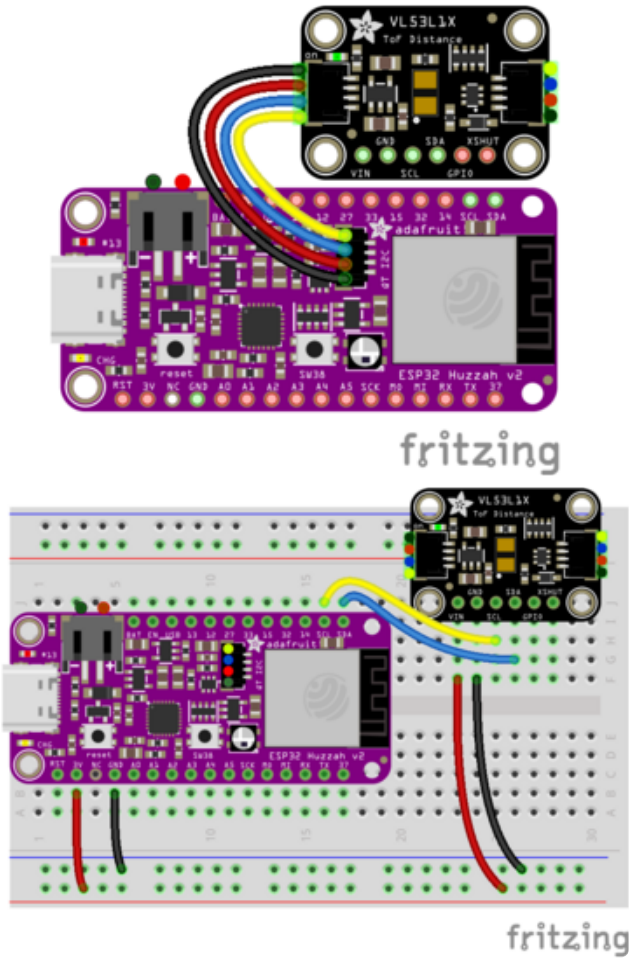
If you've never used WipperSnapper, click below to read through the quick start guide before continuing.

Quickstart: Adafruit IO WipperSnapper

https://adafru.it/Vfd

Wiring

First, wire up an VL53L1X to your board exactly as follows. Here is an example of the VL53L1X wired to an Adafruit ESP32 Feather V2 (http://adafru.it/5400) using I2C with a STEMMA QT cable (no soldering required) (http://adafru.it/4210)

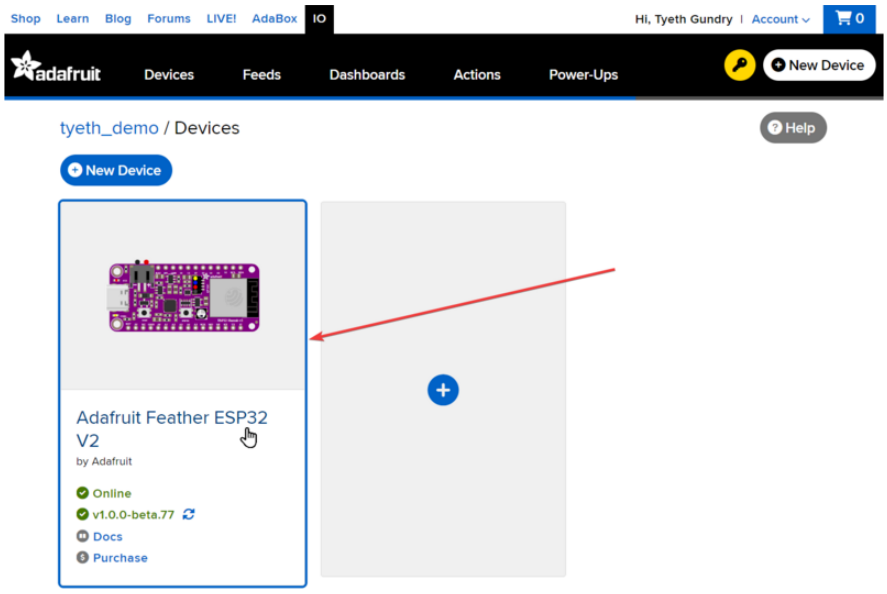


Board 3V to **sensor VIN (red wire on STEMMA QT)**
Board GND to **sensor GND (black wire on STEMMA QT)**
Board SCL to **sensor SCL (yellow wire on STEMMA QT)**
Board SDA to **sensor SDA (blue wire on STEMMA QT)**

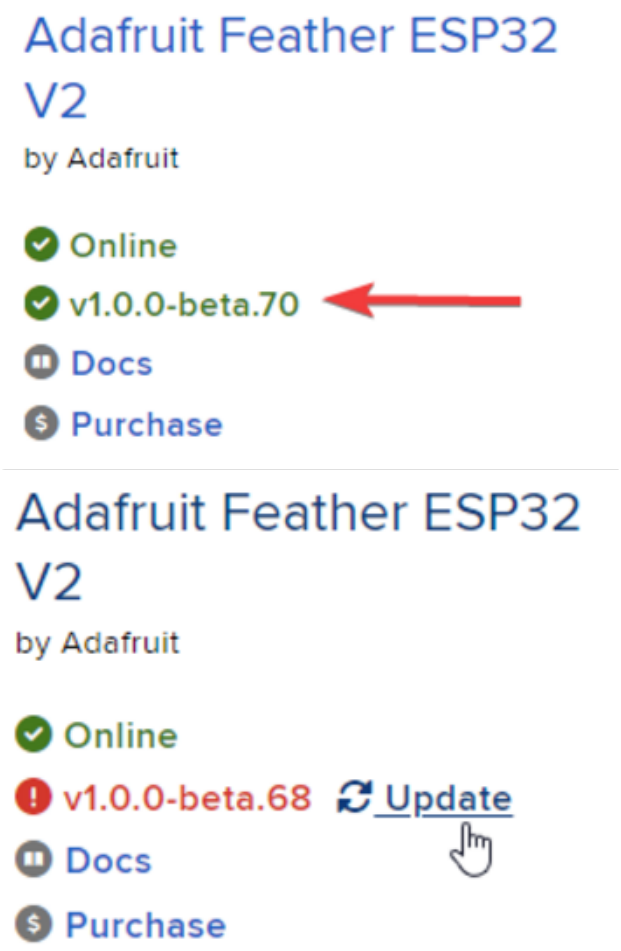
Usage

Connect your board to Adafruit IO Wippersnapper and navigate to the WipperSnapper board list **(https://adafru.it/TAu)**.

On this page, **select the WipperSnapper board you're using** to be brought to the board's interface page.



If you do not see your board listed here - you need to connect your board to Adafruit IO (https://adafru.it/Vfd) first.

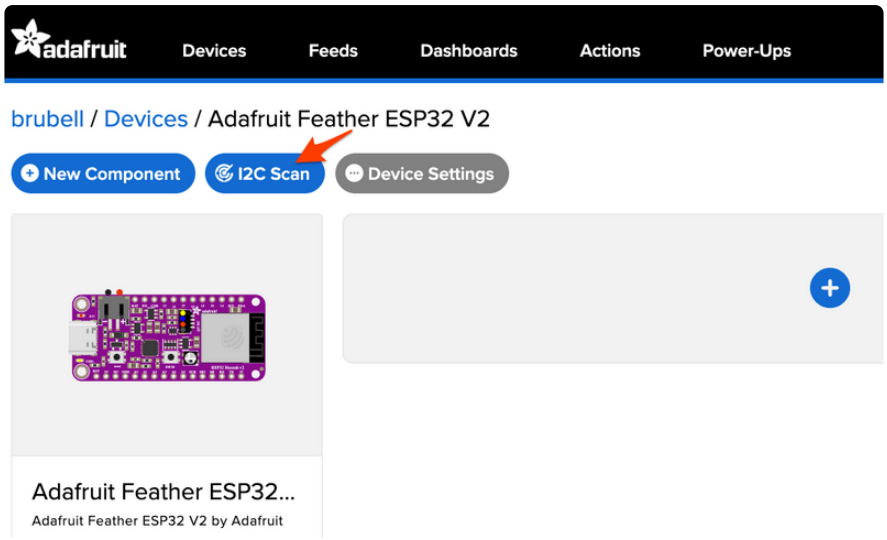


On the device page, quickly **check that you're running the latest version of the WipperSnapper firmware**.

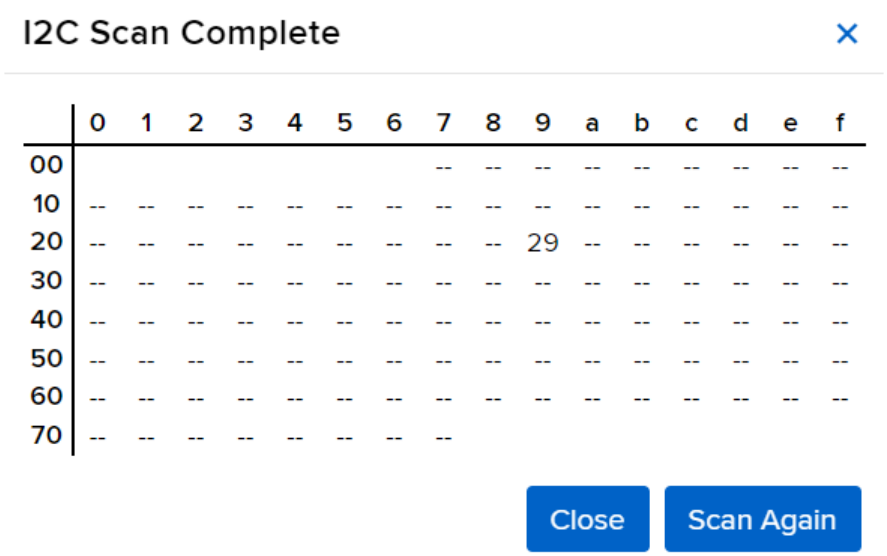
The device tile on the left indicates the version number of the firmware running on the connected board.

If the firmware version is green with a checkmark - continue with this guide.
If the firmware version is red with an exclamation mark "!" - update to the latest WipperSnapper firmware (https://adafru.it/Vfd) on your board before continuing.

Next, make sure the sensor is plugged into your board and click the **I2C Scan** button.



You should see the VL53L1X's default I2C address of `0x29` pop-up in the I2C scan list.



I don't see the sensor's I2C address listed!

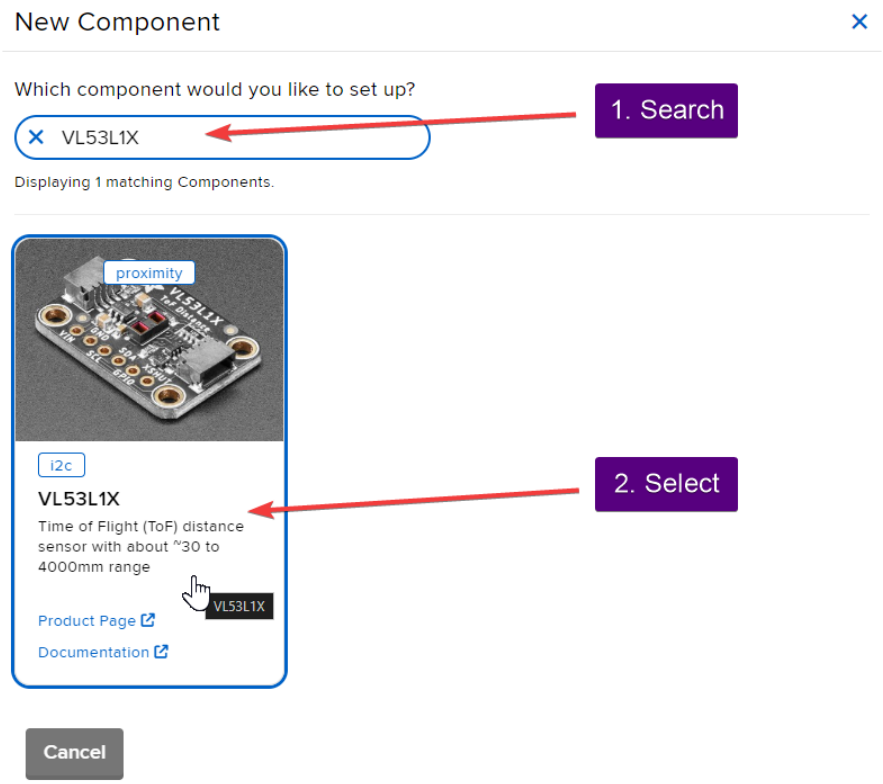
First, double-check the connection and/or wiring between the sensor and the board.

Then, reset the board and let it re-connect to Adafruit IO WipperSnapper.

With the sensor detected in an I2C scan, you're ready to add the sensor to your board.

Click the New Component button or the + button to bring up the component picker.

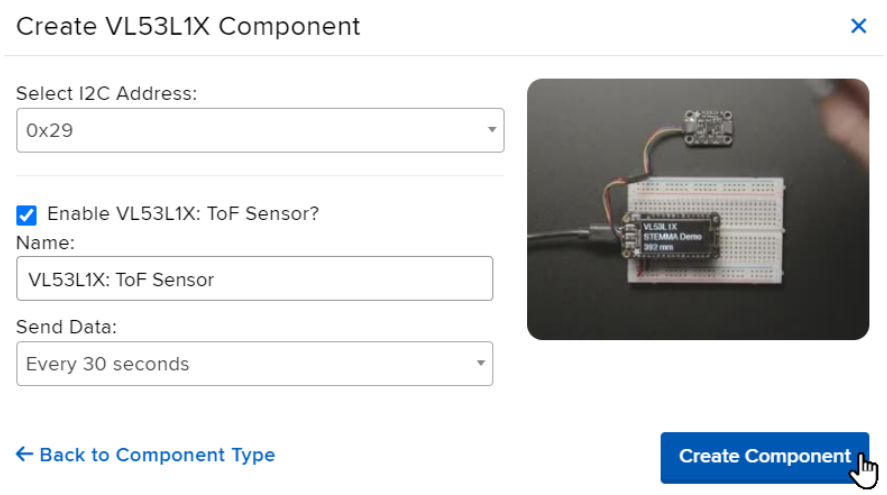
Adafruit IO supports a large amount of components. To quickly find your sensor, type `VL53L1X` into the search bar, then select the **VL53L1X** component.



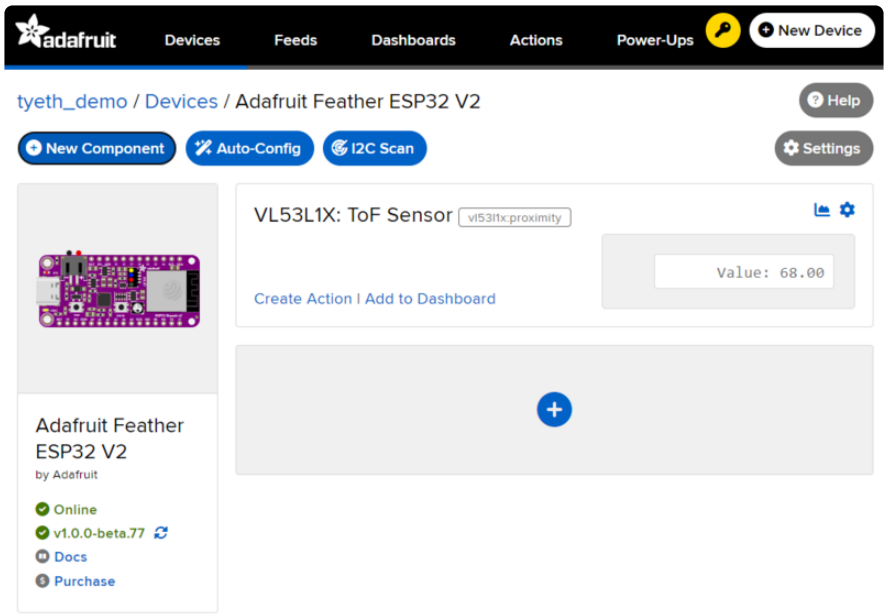
On the component configuration page, the VL53L1X's sensor address should be listed along with the sensor's settings.

The **Send Every** option is specific to each sensor's measurements. This option will tell the Feather how often it should read from the VL53L1X sensor and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

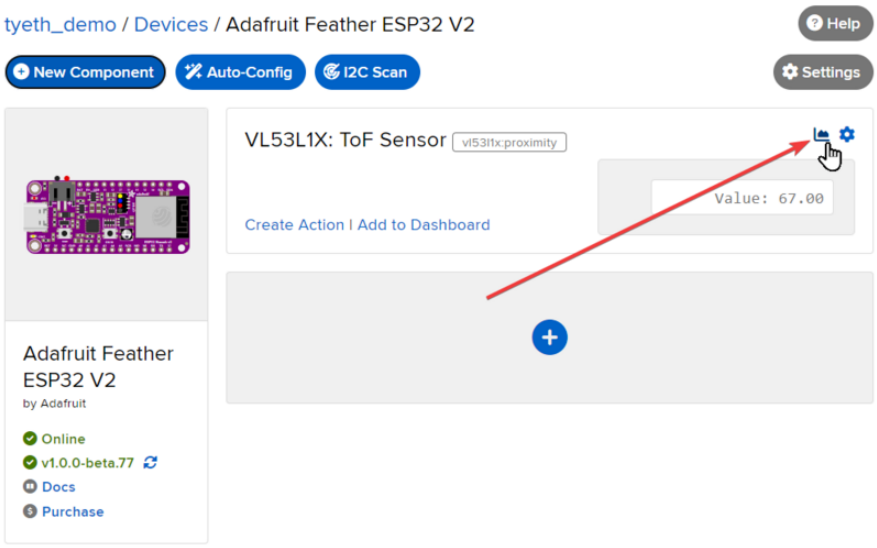
For this example, set the **Send Every** interval to every 30 seconds.



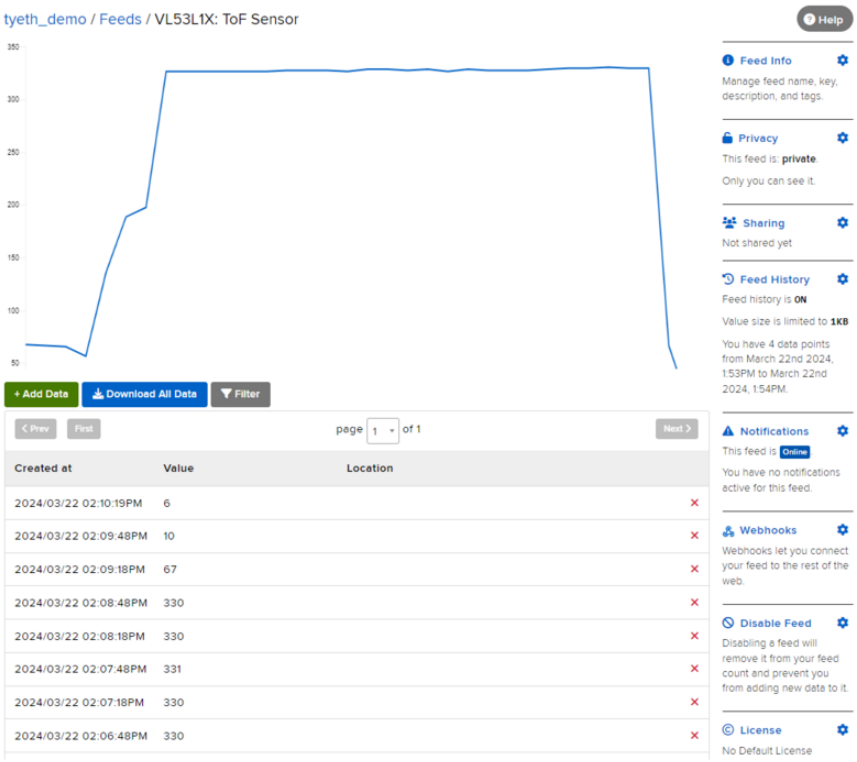
Your device interface should now show the sensor components you created. After the interval you configured elapses, WipperSnapper will automatically read values from the sensor(s) and send them to Adafruit IO.



To view the data that has been logged from the sensor, click on the graph next to the sensor name.



Here you can see the feed history and edit things about the feed such as the name, privacy, webhooks associated with the feed and more. If you want to learn more about how feeds work, check out this page (https://adafru.it/10aZ).



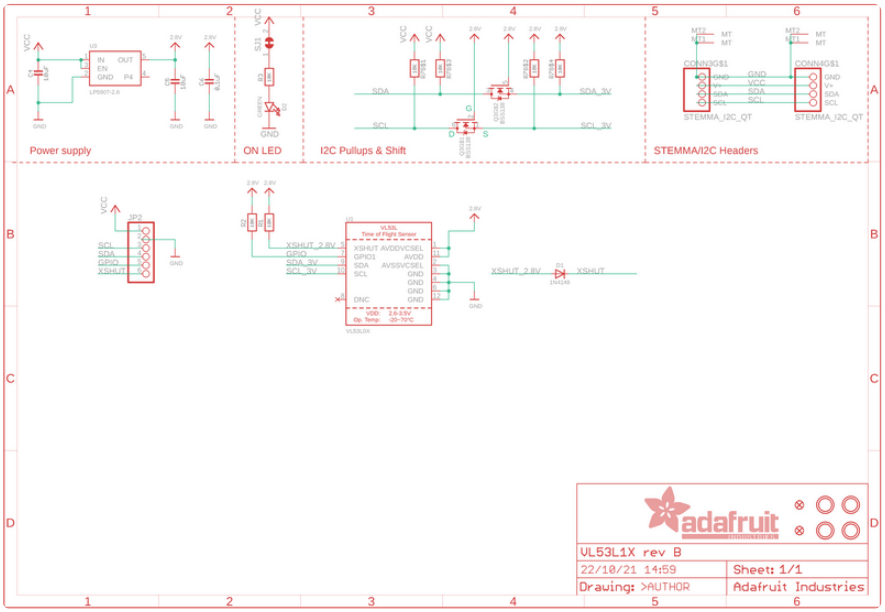
Downloads

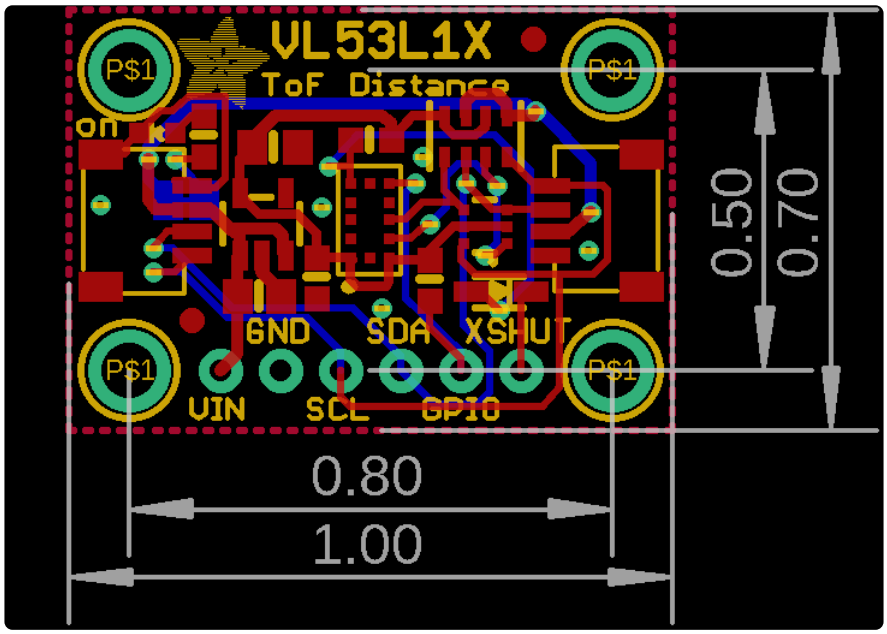
Files:

- VL53L1X datasheet (https://adafru.it/VB8)
- EagleCAD PCB files on GitHub (https://adafru.it/VB9)
- 3D models on GitHub (https://adafru.it/19Fk)

- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/VBa)

Schematic and Fab Print





3D Model