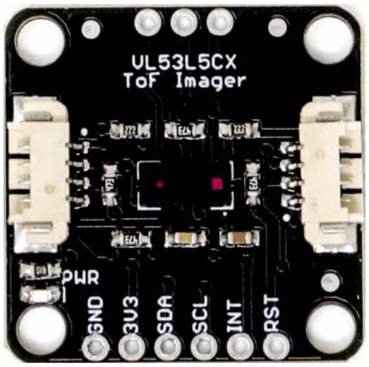
SmartElex ToF Imager - VL53L5CX



The ToF Imager - VL53L5CX breakout board IS built around ST Electronics' VL53L5CX; a state of the art, Time-of-Flight (ToF), multizone ranging sensor enhancing the ST FlightSense product family. This chip integrates a SPAD array, physical infrared filters, and diffractive optical elements (DOE) to achieve the best ranging performance in various ambient lighting conditions with a range of cover glass materials.

Multizone distance measurements are possible up to 8x8 zones with a wide 63° diagonal FoV which can be reduced by software. Thanks to ST Histogram patented algorithms, the VL53L5CX is able to detect different objects within the FoV. The Histogram also provides immunity to cover glass crosstalk beyond 60 cm.

Ideal for 3D room mapping, obstacle detection for robotics, gesture recognition, IoT, laser-assisted autofocus, and AR/VR enhancement, the connector on this sensor makes integration easy.

Hardware Overview

VL53L5CX

The ToF Imager is state of the art, 64 pixel Time-of-Flight (ToF) 4 meter ranging sensors built around the VL53L5CX from ST. To see more details, refer to the datasheet.The 7-bit unshifted I^{2}C address (most commonly used with Arduino) is **0x29**. The 8-bit I^{2}C address of the board is **0x52** for writing and **0x53** for reading.

Power

Ideally power will be supplied by the connector, but if you wish to supply your own power, pins have been broken out along the bottom side of the board labeled 3V3 and GND. The input voltage range should be between **2.7**-**3.3V**.

I^{2}C

The I^{2}C pins break out the functionality of the connectors. Depending on your application, you can connect to these pins via the plated through holes for SDA and SCL.

INT and RST

The interrupt pin is the interrupt output and defaults to an open-drain output. A 47 kΩ pull-up resistor to IOVDD is included.

The reset pin is the I^{2}C interface reset pin and is active high. It is pulled to ground with a 47 kΩ resistor.

LP, VDDIO, & VDDA

The pins in this section are specific to the 1"x1" board. LP is a *low power* enable pin. Drive this pin to logic 0 to disable the I^{2}C comms to reduce power consumption. Drive this pin to logic 1 to enable I^{2}C comms. This pin is typically only needed when you need to change the I2C address in multidevice systems. A 47 kΩ pull-up resistor to IOVDD is included so it can be left unconnected.

VDDIO/VDDA: These pins are used as an alternate power supply. By default, VDDIO and VDDA are tied together but by opening the PSU jumper they can be isolated. A user must then provide separate VDDIO and VDDA supplies. This is most applicable for users who want to use IO voltages (1.8, 2.8, or 3.3V) separate from AVDD voltages (2.8 or 3.3V) for maximum power reduction.

Jumpers

INT

Cut the **INT** jumper to remove the 47 kΩ pull-up resistor from the INT pin.

I^{2}C

The ToF Imager Sensor has two 2.2 kΩ pull-up resistors attached to the I^{2}C bus by default. If multiple sensors are connected to the bus with the pull-up resistors enabled the parallel equivalent resistance may create too strong of a pull-up for the bus to operate correctly. As a general rule of thumb, disable all but one pair of pull-up resistors if multiple devices are connected to the bus. If you need to disconnect the pull-up resistors they can be removed by cutting the traces on the corresponding jumper highlighted below.

PSU

This jumper is related to the pins specific to the ToF board. By default, VDDIO and VDDA are tied together. Cutting the **PSU** jumper will isolate the power rails. A user must then provide separate VDDIO and VDDA supplies. This is most applicable for users who want to use IO voltages (1.8, 2.8, or 3.3V) separate from AVDD voltages (2.8 or 3.3V) for maximum power reduction.

LED

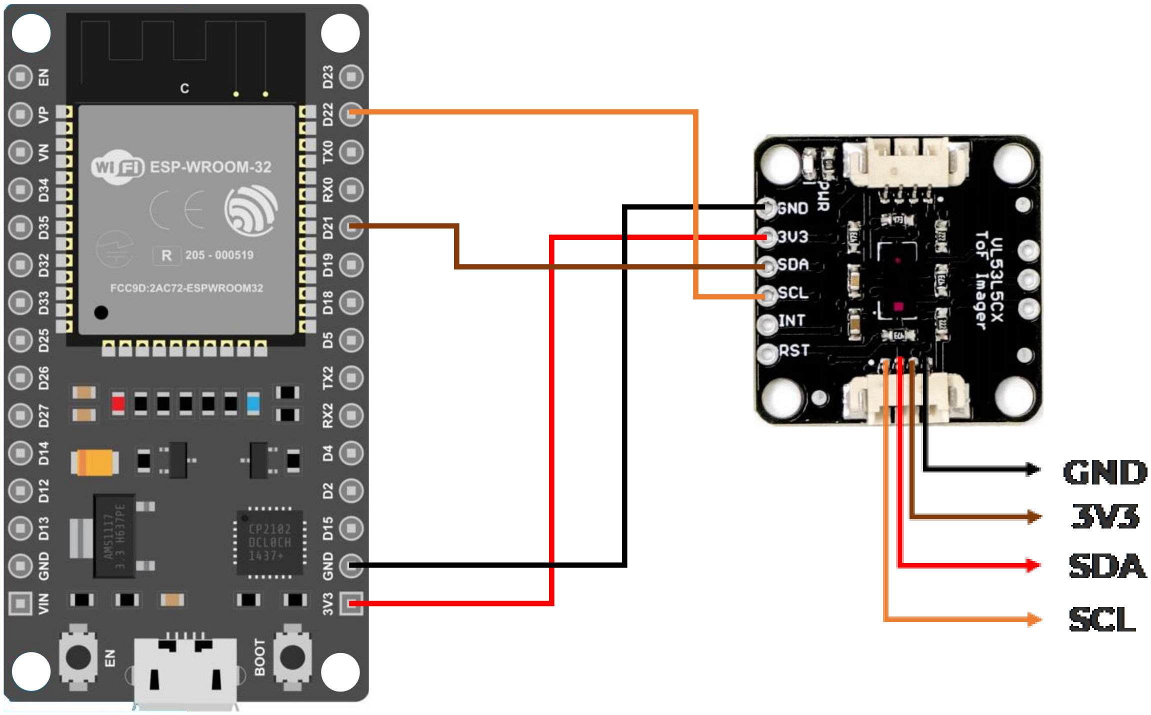
If minimal power consumption is a concern, or you just don't want that Power LED on the front of the board to light up, go ahead and cut this jumper.

A note on choosing a board: The VL53L5CX is unique in that it requires its firmware to be loaded at power-on over the I2C bus. Because this firmware is ~90k bytes, we recommend a microcontroller with enough flash to store VL53L5CX's firmware as well as your program code.

Software Setup and Programming

Sparkfun has written a simple Arduino library to quickly get started reading data from the ToF Imager. Install the library through the Arduino Library Manager tool by searching for **"SparkFun VL53L5CX"**.

Wiring:



ESP32 Devkit V1	VL53L5CX
D22(SCL)	SCL
D21(SDA)	SDA
3.3V	3V3
GND	GND

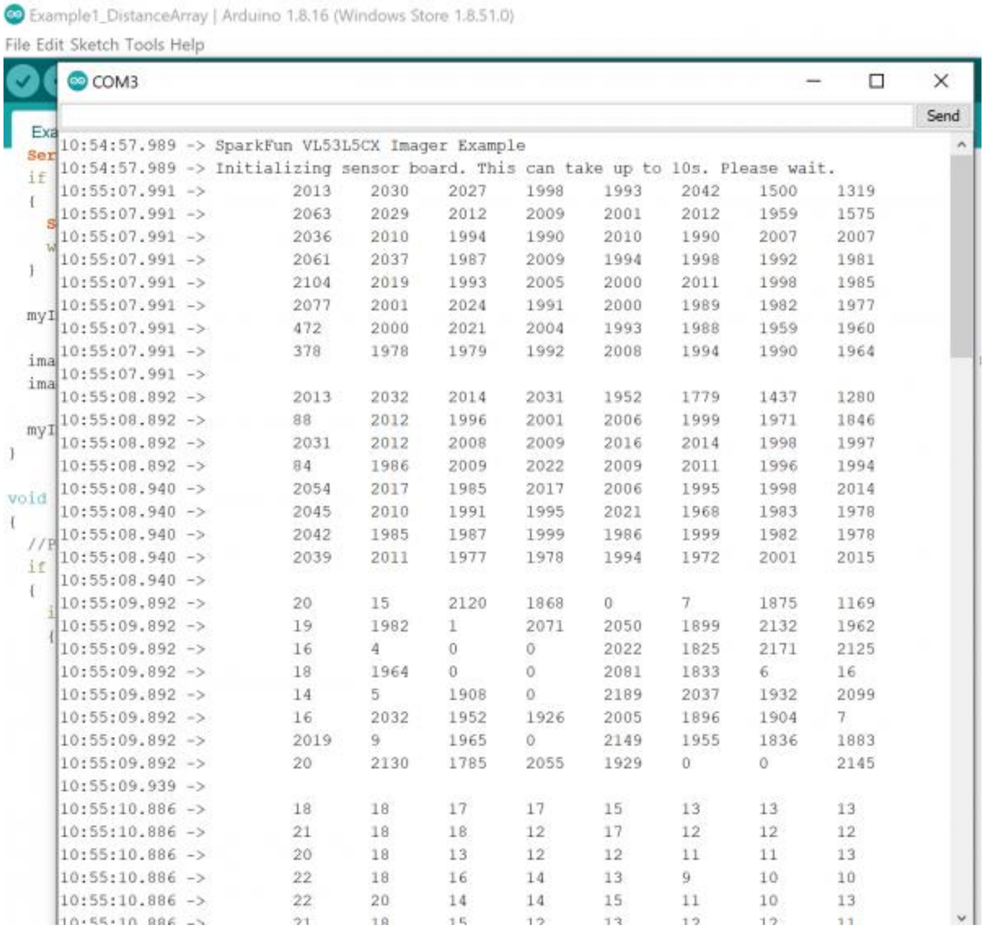
Example1_DistanceArray

Hook up your ToF imager to your Artemis Thing Plus via the cables, and click "**File** > **Examples** > **SparkFun VL53L5CX Arduino Library** > **Example1_DistanceArray**".

```
#include <Wire.h>

#include <SparkFun_VL53L5CX_Library.h> //http://librarymanager/All#SparkFun_VL53L5CX

SparkFun_VL53L5CX myImager;
VL53L5CX_ResultsData measurementData; // Result data class structure, 1356 byes of RAM

int imageResolution = 0; //Used to pretty print output
int imageWidth = 0; //Used to pretty print output

void setup()
{
  Serial.begin(115200);
  delay(1000);
  Serial.println("SparkFun VL53L5CX Imager Example");

  Wire.begin(); //This resets to 100kHz I2C
  Wire.setClock(400000); //Sensor has max I2C freq of 400kHz

  Serial.println("Initializing sensor board. This can take up to 10s. Please wait.");
  if (myImager.begin() == false)
  {
    Serial.println(F("Sensor not found - check your wiring. Freezing"));
    while (1) ;
  }

  myImager.setResolution(8*8); //Enable all 64 pads

  imageResolution = myImager.getResolution(); //Query sensor for current resolution -
either 4x4 or 8x8
  imageWidth = sqrt(imageResolution); //Calculate printing width

  myImager.startRanging();
}

void loop()
{
  //Poll sensor for new data
  if (myImager.isDataReady() == true)
  {
    if (myImager.getRangingData(&measurementData)) //Read distance data into array
    {
      //The ST library returns the data transposed from zone mapping shown in
datasheet
      //Pretty-print data with increasing y, decreasing x to reflect reality
      for (int y = 0 ; y <= imageWidth * (imageWidth - 1) ; y += imageWidth)
      {
```

```
    for (int x = imageWidth - 1 ; x >= 0 ; x--)
    {
     Serial.print("\t");
     Serial.print(measurementData.distance_mm[x + y]);
    }
    Serial.println();
   }
   Serial.println();
  }
 }

 delay(5); //Small delay between polling
}
////////////////////////////////////////END/////////////////////////////////////////
```

Open up your Serial Monitor, make sure the baud rate is set appropriately, and you should see something like the following:


Example1_DistanceArray | Arduino 1.8.16 (Windows Store 1.8.51.0)
File Edit Sketch Tools Help
COM3
Send

```
10:54:57.989 -> SparkFun VL53L5CX Imager Example
10:54:57.989 -> Initializing sensor board. This can take up to 10s. Please wait.
10:55:07.991 ->         2013    2030    2027    1998    1993    2042    1500    1319
10:55:07.991 ->         2063    2029    2012    2009    2001    2012    1959    1575
10:55:07.991 ->         2036    2010    1994    1990    2010    1990    2007    2007
10:55:07.991 ->         2061    2037    1987    2009    1994    1998    1992    1981
10:55:07.991 ->         2104    2019    1993    2005    2000    2011    1998    1985
10:55:07.991 ->         2077    2001    2024    1991    2000    1989    1982    1977
10:55:07.991 ->         472     2000    2021    2004    1993    1988    1959    1960
10:55:07.991 ->         378     1978    1979    1992    2008    1994    1990    1964
10:55:07.991 ->
10:55:08.892 ->         2013    2032    2014    2031    1952    1779    1437    1280
10:55:08.892 ->         88      2012    1996    2001    2006    1999    1971    1846
10:55:08.892 ->         2031    2012    2008    2009    2016    2014    1998    1997
10:55:08.892 ->         84      1986    2009    2022    2009    2011    1996    1994
10:55:08.940 ->         2054    2017    1985    2017    2006    1995    1998    2014
10:55:08.940 ->         2045    2010    1991    1995    2021    1968    1983    1978
10:55:08.940 ->         2042    1985    1987    1999    1986    1999    1982    1978
10:55:08.940 ->         2039    2011    1977    1978    1994    1972    2001    2015
10:55:08.940 ->
10:55:09.892 ->         20      15      2120    1868    0       7       1875    1169
10:55:09.892 ->         19      1982    1       2071    2050    1899    2132    1962
10:55:09.892 ->         16      4       0       0       2022    1825    2171    2125
10:55:09.892 ->         18      1964    0       0       2081    1833    6       16
10:55:09.892 ->         14      5       1908    0       2189    2037    1932    2099
10:55:09.892 ->         16      2032    1952    1926    2005    1896    1904    7
10:55:09.892 ->         2019    9       1965    0       2149    1955    1836    1883
10:55:09.892 ->         20      2130    1785    2055    1929    0       0       2145
10:55:09.939 ->
10:55:10.886 ->         18      18      17      17      15      13      13      13
10:55:10.886 ->         21      18      18      12      17      12      12      12
10:55:10.886 ->         20      18      13      12      12      11      11      13
10:55:10.886 ->         22      18      16      14      13      9       10      10
10:55:10.886 ->         22      20      14      14      15      11      10      13
10:55:10.886 ->         21      18      15      12      13      12      12      11
```