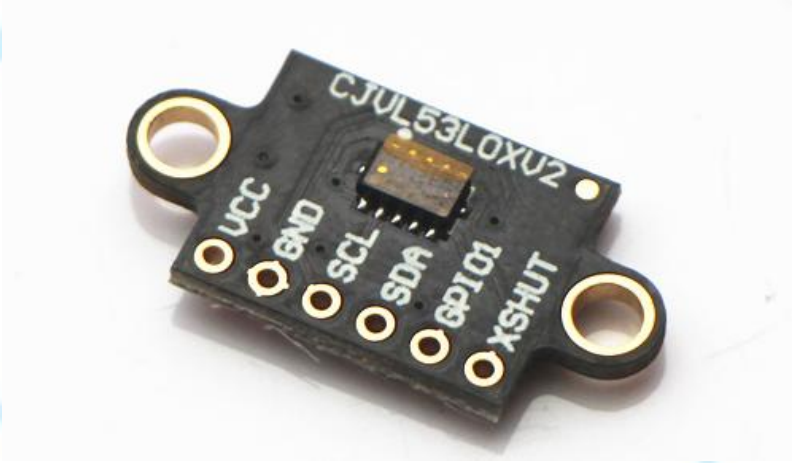
MEDIDOR DE DISTANCIA LÁSER VL53L0X



Erick Barrios Barocio.
Electrónica v.2025

Existen situaciones, en las que es necesario conocer la distancia a la que está un cierto objeto con precisión. Dicho problema se puede resolver de forma muy sencilla y con una relativa gran precisión con ayuda del módulo para Arduino del VL53L0X, el cual es un elemento electro-óptico el cual mide el retraso temporal de pulso láser para calcular distancias.

Contenido

El VL53L0X [1] es un módulo de medición de distancia láser basado en el método de tiempo de vuelo, el cual mide el retraso temporal de pulsos de luz que viajan una cierta distancia respecto de una referencia de tiempo. Este método proporciona una medición de distancia dentro de un rango de hasta 2 m.

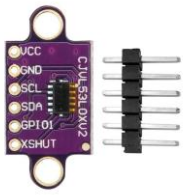


Figura 1. Módulo VL53L0X para Arduino.

El corazón del módulo es una matriz SPAD (Diodos de Avalancha de Fotón Individuales) los cuales son capaces de detectar (recibir) intensidades de luz extremadamente bajas (del orden de miles de fotones). También cuenta con un emisor VCSEL (Láser de Emisión Superficial de Cavidad Vertical) de 940 nm totalmente invisible para el ojo humano y que, junto con filtros infrarrojos físicos internos, permite una gran distancia de alcance e inmunidad a la luz ambiental. El tamaño del módulo es de $2.4\ mm \times 4.4\ mm$, por lo que varias compañías distribuidoras, lo han implementado en un módulo (Figura 1) diseñado para poder conectarse con placas como Arduino de forma más sencilla.

En el método de tiempo de vuelo se envía un pulso láser desde el VCSEL y se mide el tiempo que tarda en regresar al SPAD, después de haberse reflejado en un objetivo. El tiempo es proporcional a la distancia entre el sensor y el objeto.

Una ventaja de este módulo es que incluye un microcontrolador y software necesarios para calcular internamente la distancia recorrida por los pulsos de luz, sin necesidad de que se tenga que calcular externamente, además el software es capaz de comunicarse con lenguajes de programación como Arduino IDE a través de un protocolo I2C.

1 DISPOSICIÓN DE PINES

El módulo VL53L0X para Arduino cuenta con 6 pines (Figura 2). Visto de frente, el pin superior es el de *alimentación* de voltaje (VCC), el cual es de 5V y por lo general proviene del Arduino; el pin inferior es la *tierra* o referencia (GND); el tercero es el pin de *reloj* (SCL) para el control de la comunicación I2C; el cuarto pin es el de *transmisión de datos* (SDA), que utiliza el protocolo I2C. El siguiente pin es el de *interrupción de transmisión de datos* (GPI01), y el último es el pin de *apagado temporal* (XSHUT).

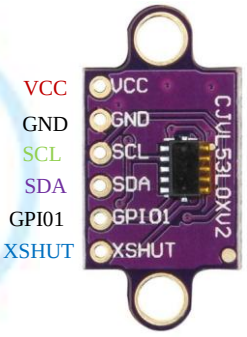


Figura 2. Diagrama de pines del módulo VL53L0X.

Para poder operar el módulo, se necesitan de por lo menos los primeros cuatro pines (VCC, GND, SCL y SDA).

2 CONEXIÓN CON UN ARDUINO

En la Figura 3 se muestran las conexiones de pines Arduino-VL53L0X necesarias para su funcionamiento. Las conexiones son bastante directas, los pines VCC y GND del módulo, se conectan a las salidas 5V y GND del Arduino, mientras que los dos pines de comunicación I2C (SDA y SCL) se conectan a los correspondientes pines del Arduino. En un Arduino UNO R3 existen cuatro de estos pines (dos de SDA y dos de SCL). Un par SDA/SCL se encuentran junto al pin AREF del Arduino cerca de la conexión USB, mientras que el otro par son los pines analógicos A4 y A5 respectivamente.

Algo a tomar en cuenta es que, por defecto, la dirección del módulo, bajo el protocolo de comunicación I2C, es 0x29. En caso de que se requiera cambiar, se puede hacer mediante la función "*setAddress*".

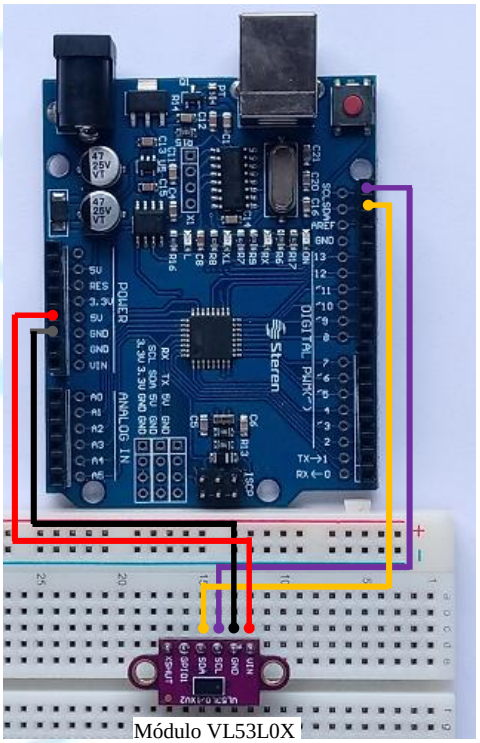


Figura 3. Diagrama de conexión Arduino-VL53L0X.

2.1 LIBRERIA

Para poder manejar el módulo a través del Arduino IDE, es necesario instalar la librería "*VL53L0X*". Existen varias librerías para el módulo, en nuestro caso su utilizará la librería hecha por la compañía Pololu, la cual se encuentra en la referencia [2], y se instala de forma manual, descargando el archivo comprimido y descomprimiéndolo en la carpeta de librerías del Arduino IDE.

2.2 MODOS DE OPERACIÓN

Los dos principales modos de operación del módulo son "Medición simple" y "Medición continua"; además, se tienen sub-modos de operación para grandes distancias, para alta velocidad y para alta precisión. A continuación, se muestran y describen ejemplos de código que ejemplifican estos modos.

A) MODO SIMPLE

En este modo, la medición es realizada solo una vez cada que el comando de medición es invocado, después de lo cual el modulo regresa a modo de espera,

```
#include <Wire.h>          // Incluye la librería para comunicación I2C
#include <VL53L0X.h>       // Incluye la librería del módulo VL53L0X

VL53L0X sensor;            // Se declara la existencia del módulo y se nombra como sensor.

// Quitar el comentario de la siguiente línea para utilizar el sub-modo de distancias grandes.

//#define GRAN_RANGO  //  Este sub modo incrementa la sensibilidad del sensor y extiende su rango de
                      //medición. Sin embargo, aumenta la posibilidad de tener mediciones imprecisas
                      //debido a reflejos provenientes de otros objetos. Se recomienda utilizar este
                      //sub-modo en condiciones de obscuridad.

// Quitar el comentario de una de las siguientes líneas para tener:
```

```
14
15 //#define ALTA_VEL              // - Mayor velocidad de medición a expensas de la precisión
16 #define ALTA_PRECISION        // - Mayor precisión a cambio de menor velocidad de medición
17
18 void setup()
19 {
20   Serial.begin(9600);       // Se inicia la pantalla serial para mostrar las mediciones.
21   Wire.begin();             // Se inicia el protocolo de comunicación I2C con el módulo
22
23   sensor.setTimeout(500);  // Fija un intervalo de tiempo en ms, después del cual los intentos de medir
24                             //terminarán si el sensor no está listo.
25   if (!sensor.init())       //Inicializa y configura el sensor, si hay un problema manda un aviso de error
26     {
27       Serial.println("Error para detector o iniciar el sensor!");
28       while (1) {}          // Mientras continue el error no procederá
29     }
30
31   #if defined GRAN_RANGO          // Si este modo se escogió:
32     sensor.setSignalRateLimit(0.1);   // Baja el límite de la taza de regreso de la señal (por defecto es 0.25 MCPS)
33     sensor.setVcselPulsePeriod(VL53L0X::VcselPeriodPreRange, 18);      // ver texto
34     sensor.setVcselPulsePeriod(VL53L0X::VcselPeriodFinalRange, 14);   // ver texto
35   #endif
36
37   #if defined ALTA_VEL                          // Si este modo se escogió.
38   sensor.setMeasurementTimingBudget(20000);   // ver texto
39   #elif defined ALTA_PRECISION                 // Si este modo se escoge.
40   sensor.setMeasurementTimingBudget(200000);
41   #endif
42 }
43
44 void loop()
45 {
46   Serial.print(sensor.readRangeSingleMillimeters());          // Imprime el resultado. Ver texto
47   if (sensor.timeoutOccurred()) { Serial.print(" Error"); }     // Si el tiempo de la medición se sobrepasa
48   Serial.println();                                            //sin obtener un resultado se imprime error
49 }
```

En la línea 32, la función “sensor.setSignalRateLimit(0.1)”, establece el límite de velocidad de la señal de retorno (en Mega Cuentas Por Segundo), la cual es la amplitud mínima de la señal reflejada por el objetivo, y recibida por el sensor, necesaria para que la lectura sea válida. Establecer un valor bajo aumenta el rango potencial del sensor ya que permite detectar señales más débiles (que viajen mayor distancia), pero también aumenta la probabilidad de obtener una lectura inexacta debido a reflejos de objetos distintos al objetivo previsto.

En las líneas 33 y 34, se usa la función "sensor.setVcselPulsePeriod()", la cual establece el período de pulsos del emisor láser. Esta función tiene dos pasos: "VcselPeriodPreRange", el cual se usa para ajustar la emisión láser durante la calibración del sistema y solo acepta valores pares entre 12 y 18 (por defecto se tiene un valor de 14); la otra función es "VcselPeriodFinalRange", el cual es el periodo final utilizado durante las mediciones y toma valores pares de 8 a 14 (por defecto se tiene el valor de 10). Los períodos más largos aumentan el alcance potencial del sensor ya que permiten que el sistema espere un mayor tiempo para el regreso de los pulsos sin el inconveniente de que se pueda confundir con un pulso posterior.

En las líneas 38 y 40 se utiliza la función "setMeasurementTimingBudget()", la cual establece el tiempo que durará un proceso de medición en microsegundos. Un valor de tiempo largo permite mediciones más precisas. El valor predeterminado es de 33000 microsegundos o 33 ms; el mínimo es 20 ms.

En la línea 46 se tiene la función más importante de este código, "sensor.readRangeSingleMillimeters()". Realiza una medición de distancia una sola vez y devuelve la lectura en milímetros. Si el código se repite, realizará otra medición.

Figura 4. Resultado mostrado en la pantalla serial de la operación de modo sencillo. Se incluye el arreglo experimental.

Este código muestra en la pantalla serial la distancia en milímetros a la cual se encuentra algún objeto, en nuestro caso una caja(Figura 4).

B) MODO COINTINUO

En este modo, la medición es realizada de forma continua, una después de otra a menos que se especifique lo contrario.

```
1   #include <Wire.h>              // Incluye la librería para comunicación I2C
2   #include <VL53L0X.h>           // Incluye la librería del módulo VL53L0X
3
4   VL53L0X sensor;                // Se declara la existencia del módulo y se nombre como sensor.
5
6   void setup()
7   {
8     Serial.begin(9600);          // Esta parte es igual que el código anterior.
9     Wire.begin();

10    sensor.setTimeout(500);
11    if (!sensor.init())
12    {
13      Serial.println("Failed to detect and initialize sensor!");
14      while (1) {}
15    }
16
17    sensor.startContinuous();   //Se activa el modo continuo. Ver texto para detalles.
18  }
```

```
19
20  void loop()
21  {
22    Serial.print(sensor.readRangeContinuousMillimeters());        // Imprime el resultado. Ver texto
23    if (sensor.timeoutOccurred()) { Serial.print(" Error"); }
24    Serial.println();
25  }
```

Este código presenta las diferencias respecto del anterior en las líneas 17 y 22.

En la línea 17 se utiliza la función "sensor.startContinuous()", la cual inicia mediciones de distancia una después de otra. El argumento es un periodo de espera entre mediciones continúas dado en ms (toma el valor predeterminado de cero si no se especifica). Así, el sensor tomará mediciones con la mayor frecuencia posible dependiendo del valor de espera y solo se detendrá hasta que el código termine.

En la línea 22, la función "sensor.readRangeContinuousMillimeters()", realiza la medición de distancia de forma continua y van mostrando los resultados obtenidos.

El resultado en pantalla de este modo es el mismo que el del código anterior, sin embargo, permite una mayor recolección de datos o visto de otra forma es más rápido, lo cual puede ser conveniente en ciertas aplicaciones.

3 CONSIDERACIONES EXTRA PARA EL VL53L0X

Otras consideraciones importantes al momento de operar el módulo VL53L0X son la textura y color del objetivo. De acuerdo con la hoja de datos [1], la precisión en las mediciones de distancia aumentan cuando se trata de objetivos blancos y disminuye cuando el objetivo es obscuro, de igual forma, la mediciones son más precisas en condiciones de interior (ambientes de baja luz infrarroja) que en el exterior (ver Tabla 1).

También se ha encontrado que los materiales con terminación mate dispersan la luz, por lo que incrementan la desviación en las mediciones. Se ha encontrado que los objetivos claros con terminación brillante son los que presentan las mejores mediciones. De igual forma, variaciones en el voltaje de alimentación o en la temperatura (el punto de funcionamiento óptimo es de 23°C) pueden generar desviaciones de las mediciones de hasta 10mm, por lo que se recomienda tener condiciones controladas en caso de requerir mediciones precisas.

Como se mencionó en la sección anterior, la precisión de medición del módulo depende del sub-modo de operación seleccionado. En la Tabla 2 se sugieren los límites de operación del módulo dependiendo del modo de operación.

Tabla 1. Precisión en la Distancia [1].

Nivel de Reflectancia del Objetivo	En Interior (no IR)			Exterior (con IR)		
	Distancia	TimingBudget 33ms	TimingBudget 66ms	Distancia	TimingBudget 33ms	TimingBudget 66ms
Objetivo Blanco al 88%	120 cm	4%	3%	120 cm	7%	6%
Objetivo Gris al 17%	70cm	7%	6%	70cm	12%	9%

Tabla 2. Sub-modos de Medición [1].

Sub-modo	Timing Budget	Error relativo a 1.2m	Aplicación
Precisión Alta	200 ms	≤ ±3%	Mediciones precisas
Gran Rango	33ms	3% a 7%	Distancias grandes (en condiciones de obscuridad)
Alta Velocidad	20ms	±5%	Cuando la precisión no es importante.

3.1 SOBRE LA CALIBRACIÓN

La librería "*VL53L0X*" cuenta con muchas otras funciones [3]; sin embargo, la mayoría están relacionadas con cuestiones de calibración del sensor. De acuerdo a la hoja de datos del módulo, la calibración de fabrica puede ser ajustada mediante programación directa de su software interno; sin embargo, esto requiere de mayor conocimiento en el lenguaje utilizado para manejar el módulo, por lo cual no es tan fácil llevarlo a cabo.

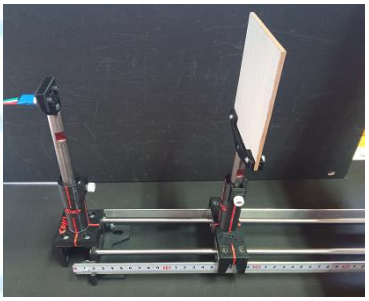


Figura 5. Montaje para calibración del módulo VL53L0. El módulo se encuentra en el encapsulado sobre el poste a la izquierda del riel.

Aun cuando el módulo es nuevo, es muy probable que las mediciones de distancia puedan estar desviadas. Esto requiere de calibrar el módulo a través del IDE del Arduino.

Para esto, se necesita contar con un arreglo experimental que permita medir la distancia entre el módulo y una pantalla blanca con terminación brillante con relativa buena precisión (en nuestro caso un riel graduado). El módulo se coloca en el origen del riel, mientras que la pantalla en un carro móvil, el cual permite medir su posición relativa al origen (Figura 5). Posteriormente se realizan mediciones de la "distancia real" (marcada por el riel) y de la "distancia medida" (dada por el módulo).

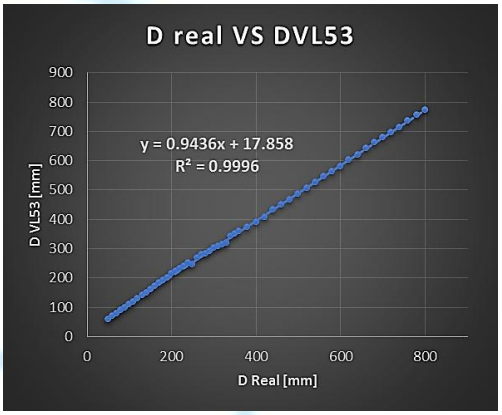


Figura 6. Gráfica de mediciones de distancia Real (dada por el riel) vs distancia dada por el módulo VL53.

Al graficar estas dos cantidades y hacer un ajuste lineal (Figura 6) se observan dos cosas:

- La pendiente debería ser 1 ya que se trata de la misma distancia.
- La ordenada al origen debería ser cero.

Para resolver estos problemas se tiene que aplicar una simple transformación lineal a los datos, en la que cada medición se multiplica por un valor promedio tal que corrija la pendiente (en nuestro caso $m = 1.062/mm$) y se le reste otro valor promedio para corregir la ordenada al origen (en nuestro caso $b = -17.86mm$). Esto se puede realizar en el código de operación del módulo simplemente aplicando una función del tipo $y = mx + b$ a los datos proporcionados directamente por el módulo (la variable x son dichos datos). Después de aplicar dicha transformación, los resultados de mediciones de distancia deben ser más próximos al valor real (Figura 7).

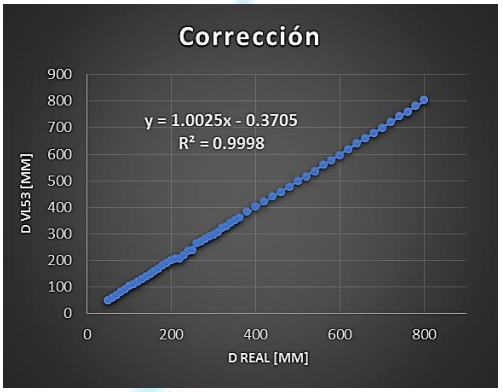


Figura 7. Gráfica de mediciones corregidas.

A pesar de esto, debido a las fluctuaciones inherentes al sensor, la corrección sólo será en promedio, existiendo siempre una incertidumbre en la medición. En nuestro caso se encontró que la incertidumbre en la pendiente era de hasta $\Delta m = 0.002/mm$ y la incertidumbre en la ordenada al origen de hasta $\Delta b = 0.9mm$.

3.2 EJEMPLO DE CODIGO

La calibración anterior se puede llevar a cabo utilizando el siguiente código:

```
1  #include <Wire.h>                 // Librería de comunicación I2C de Arduino
2  #include <LiquidCrystal_I2C.h>  // Librería del módulo I2C/LCD
3  #include <VL53L0X.h>            // Librería del VL53L0X
```

```
4
5   LiquidCrystal_I2C lcd(0x27,16,2);  // Define la dirección y tamaño de la pantalla
6   VL53L0X sensor;                    // Define el VL53L0X como sensor
7
8   double Medicion;     //variable que almacena la medición directa
9   double Distancia;    //variable de distancia real recorrida por el pulso de luz
10  double mI = 1.062;   //variables para calibración
11  double bI = -17.86;
12
13  void setup() {
14    pinMode(13, INPUT);     //Prepara el pin 13, donde está conectado el botón, como entrada
15    Wire.begin();           //Inicia la librería wire.h
16    sensor.init();          //Inicia el sensor (VL53L0X)
17    sensor.setTimeout(500); //Tiempo en ms de espera para que el sensor encienda, si no, se reporta falla
18    if (!sensor.init()) {
19      Serial.println("Fallo al detectar el sensor!");
20      while (1) {}
21     }
22    sensor.setSignalRateLimit(0.1); // Límite de la Taza de retorno de señal (en MCPS)
23    sensor.setVcselPulsePeriod(VL53L0X::VcselPeriodPreRange, 16);    // Periodos del pulso láser
24    sensor.setVcselPulsePeriod(VL53L0X::VcselPeriodFinalRange, 12);
25    sensor.setMeasurementTimingBudget(100000);  // Tiempo (us) de análisis de la medición.
26    lcd.init();          // Inicializa la pantalla
27    lcd.clear();         // Limpia pantalla
28    lcd.backlight(); // Enciende la luz de fondo
29    delay(300);
30  }
31
32  void loop() {
33    if (digitalRead(13) == HIGH){     // Si el botón se presiona
34    lcd.clear();                      // Limpia la pantalla
35    lcd.setCursor(3,0);               //Pone el cursor en la columna 2 de la primera fila
36    lcd.print("Calculando");
37    lcd.setCursor(3,1);
38    lcd.print("Espera 20s");
39    Medicion = 0;                  // La variable de Distancia se inicia en cero (mm)
40    for (int i=1; i<=100; i++){  // Repite la medición 100 veces para incrementar la precisión y promediar
41      Medicion = Medicion + sensor.readRangeSingleMillimeters();  // Acumula las mediciones de distancia
42    }
43    Medicion = Medicion/100;         // Promedio de mediciones
44    Distancia = mI*Medicion + bI;  // Corrección de desviación
45    lcd.clear();
46    lcd.setCursor(2,0);
```

```
47    lcd.print("Distancia:");
48    lcd.setCursor(2,1);
49    lcd.print(Distancia);
50    lcd.print(" mm");
51    delay(6000);
52  }
53
54  else {
55    lcd.clear();
56    lcd.setCursor(2,0);
57    lcd.print("En espera");
58    delay(100);
59    }
60  }
```

NOTA: este código está diseñado para utilizar una pantalla LCD 16×2 con módulo I2C.

Para utilizar el código, primero se deben hacer las mediciones de distancia directas dadas por el módulo VL53L0X, y para esto las variables mI y bI deben igualarse a uno y cero respectivamente (líneas 10 y 11). Posteriormente, ya que se grafican las mediciones y se encuentran los factores de corrección se sustituyen a los nuevos valores.

4 MEDICION DE TIEMPO DE VUELO

A pesar de que el módulo VL53L0X basa su funcionamiento en medir retrasos de tiempo en el regreso de los pulsos de luz que manda, está programado internamente para proporcionar la distancia a la cual se encuentra el objeto que refleja los pulsos (la mitad de la distancia recorrida); por lo que, si lo que requerimos es conocer dicho tiempo de retraso es necesario calcularlo a partir de la distancia recorrida.

De acuerdo con la hoja de datos del módulo, la calibración de fábrica es realizada en condiciones ambientales típicas, es decir, en aire a temperatura de 23°C y baja humedad. De esto se infiere que el módulo calcula la distancia utilizando un valor de velocidad de la luz en aire, la cual podemos aproximar a la del vació (en base a la incertidumbre y dispersión de las mediciones) $v \approx 2.99792 \times 10^8 \frac{m}{s} = 0.299792 \frac{mm}{ps}$.

Así, implementar un código que nos proporciones el tiempo de vuelo (o retraso de los pulsos) se reduce a agregar una línea, en el código anterior, en donde la Distancia proporcionada (corregida) por el VL53 se multiplique por dos (ida y vuelta) y se divida por 0.299792, lo cual nos dará el tiempo de retraso en picosegundos.

5 REFERENCIAS

[1] ST Microelectronics, *VL53L0X-Time-of-Flight (ToF) ranging sensor*, 2024. https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html

[2] Pololu, *VL53L0X*, Arduino Library List. 2024. https://www.arduinolibraries.info/libraries/vl53-l0-x

[3] GitHub, Pololu. *VL53L0X library for Arduino*. 2024. https://github.com/pololu/vl53l0x-arduino