

# Introducing tinyLiDAR

The Maker-Friendly Laser Sensor

Have you heard of LiDAR? Chances are that you probably have. It's fast becoming the new buzzword in the news lately. It's the cool new tech that can be found in all sorts of new products like driverless cars, drones and even smartphones.

So what is it? LiDAR stands for "Light Detection and Ranging" and if you Google it, you'll find a bunch of references to long range airborne surveying systems. That was the old-school LiDAR. Today's LiDAR is much more compact, lightweight and lower power so its finding its way into many areas that were unimaginable in the past.

LiDAR works just like RADAR but instead of sending out high frequency radio bursts, it uses infrared laser light.

Professional LiDAR systems like those used on driverless cars incorporate ultra-fast Analog to Digital converters, rotating platforms and high-end optics so they're generally not very hobbyist friendly. Fortunately, for those that don't need the ultimate in ranging distance and measurement speed, there are simpler and lower cost options.

A company called ST Microelectronics has created some of the world's smallest time-of-flight (ToF) sensors which have been integrated into more than quarter of a billion smartphones and computers to date. You probably have one in your phone now.

Their first gen product was called the VL6180X and it could measure up to about 10cm but their more recent, second gen ToF sensor is called the VL53L0X and it can measure up to 2meters thanks to its built-in Vertical Cavity Surface-Emitting LASER (VCEL) and matrix of over 200 high sensitivity Single Photon Avalanche Diode (SPAD) receivers. This particular ToF sensor is what we used in tiny**LiDAR**.

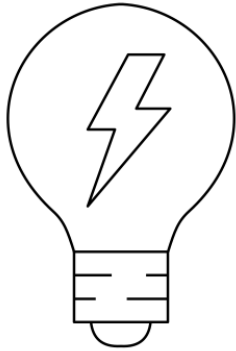
Of course, measuring the speed of light needs some very high performance electronics. Thankfully, all of this is integrated inside of the VL53L0X sensor. In fact, this little sensor can resolve all the way down to a few **pico**seconds which means it's capable of providing mm level accuracies. Pretty impressive for such a small device!

Since these sensors were designed to be used in smartphones, they are typically paired with high performance processors. Unfortunately, the Arduino Uno is not considered high performance. Remarkably, however, there are some bright individuals who have managed to make the Uno work with this sensor. But, as surmised there are some performance/feature limitations and the resulting library eats up quite a bit of Arduino's precious little memory space.

The inner workings of the VL53L0X is not well known publicly. ST has developed this technology which they call FlightSense™ on their own and patented it so its not surprising that they would want to keep the details under wraps/under NDA. But to give them some credit, they have made public, a lengthy 157 page Photonics Abstraction Layer (PAL) API spec for this little sensor. For most non-career programmers, however, it's a bit much to take in.

Like all innovative products, the idea for tiny**LiDAR** came along the way of building something else. Some time ago, we had a need for an accurate, low power and inexpensive proximity sensor.

Ultrasonic sensors were our first choice but we found them to be inaccurate, slow and power hungry. The new ST ToF sensors also seemed to fit the bill nicely. However, after digging a bit deeper, running the demos and immersing ourselves in the ST PAL API, we quickly found that using them was going to be anything but simple! And then it hit us...



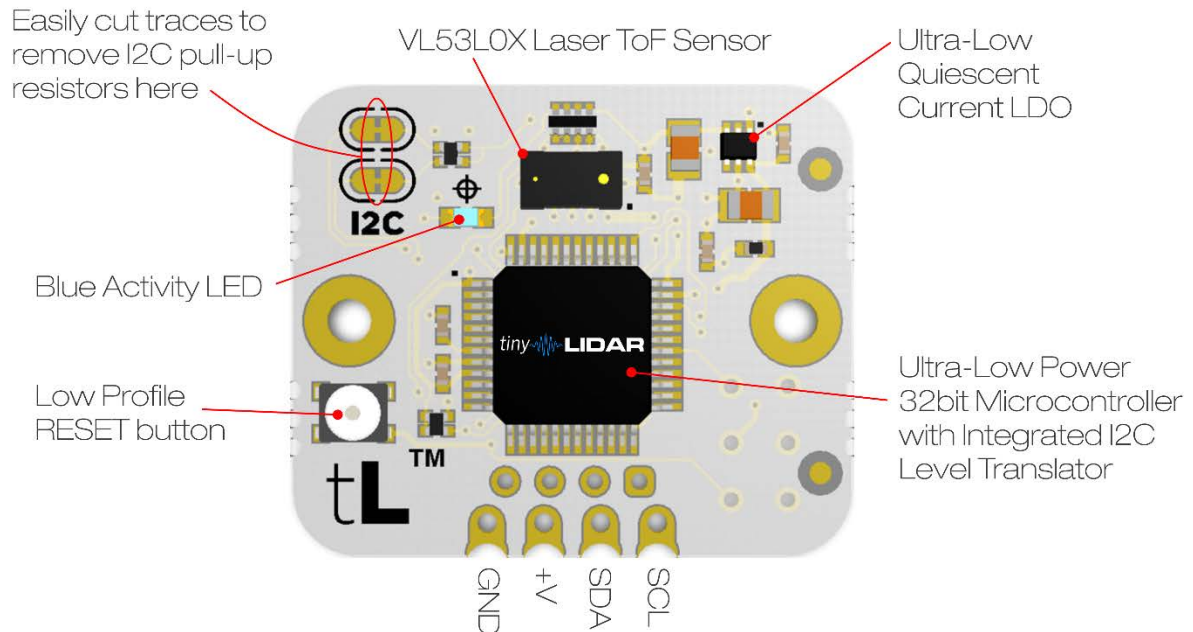
What if we could bolt on a dedicated micro to deal with the complex API and abstract it out to just simple I2C commands? Then even the UNO could run it at full speed... EUREKA!  
The idea for tiny**LIDAR** was born.

---

Well, it took a while to hash out the firmware and optimize the PCB, but we think we **finally** have a robust, nifty little sensor that's perfect for almost any range sensing application up to about 2meters.

tiny**LIDAR** is far easier to use, more accurate, faster and lower power than any of the distance sensors we've seen, and since its I2C based, paralleling them is a breeze!

**Fast** up to 60Hz sample rates  
**Accurate** to 3%  
**Low Power** <3uA stdby in SingleStep ULP mode  
**Long Range** up to 2m+  
**Small Size** 21 x 25 x 8.3mm  
**Autonomous Mode** runs at low power without an Arduino!



GROVE/0.1inch/Castellation Pads for I2C connection

## Features

- Advanced Technology: **Eye safe** VCSEL Class 1 Laser
- **Ultra Fast:** up to 60Hz sample rates and up to approx 930Hz I2C reading rates even with an Arduino UNO.
- Accurate: to +/-3% with mm precision. Every unit is pre-calibrated by us before shipping.
- Ultra Low Power: <3uA typ Quiescent Current in Single Step Ultra Low Power Mode at 2.8v supply.
- Long Range: up to 2 meters.
- Small Size: 21 x 25 x 8.3 mm.
- Light Weight: <1.5 g.
- Easy to Use: control with I2C commands (no sensor library required)
- Easy to Parallel using I2C bus
- RoHS compliant, Lead-Free assembly
- Integrated level shifter for native 3v to 5v operation
- Remotely re-configurable I2C board address and operation modes
- On board non-volatile storage for I2C address, calibration and settings
- Several connector options including GROVE, SMD or standard 0.1 inch (2.54mm) header pins
- Blue activity LED to show measurement activity
- Hardware WatchDog Timer to automatically reboot tinyLiDAR if no I2C bus activity is seen in approx 3 seconds.

## Example Code

The sketch below gives you an idea of the minimal code required to talk to tinyLiDAR from an Arduino UNO. Compiled size is only 2396 bytes!

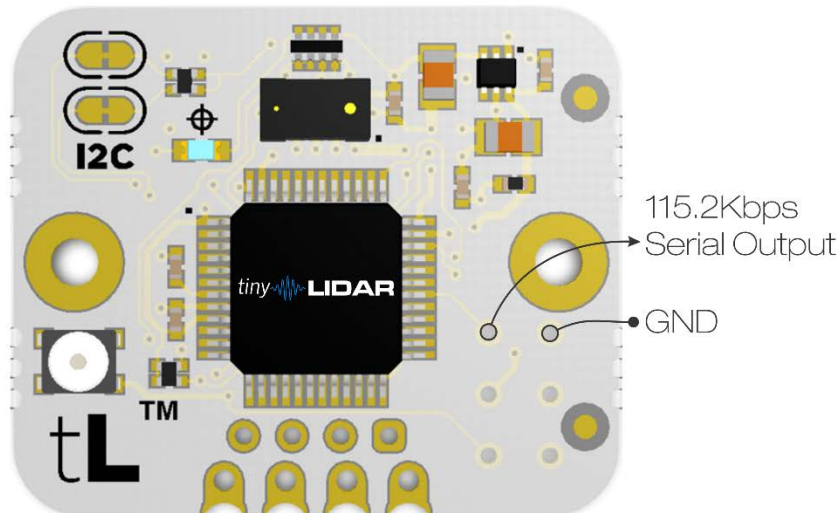
- Note that although this sketch can provide data at up to 75Hz using the default single step mode, new data is available at “only” up to approx a 60Hz scan rate. If desired, even faster reading rates of up to approx 930Hz are possible - see Appendix A for details.

```
void setup()
{
  Serial.begin(115200);    //setup the serial port
  I2c.begin();
}

void loop()
{
  uint16_t i;
  while(1)
  {
    I2c.write(0x10,'D'); //take single measurement
    I2c.read(0x10,2);    // request 2 bytes from tinyLiDAR
    i = I2c.receive();   // receive MSB byte
    i = i<<8 | I2c.receive(); // receive LSB byte and put them together
    Serial.println(i);  // print distance in mm
    delay(100);        // delay as required (13ms or higher in default single step mode)
  }
}
```

## Serial (TTY) Log Output

As an option, you can monitor some of the activity from tinyLiDAR on its serial log output port. This output only serial stream is available from the pin shown in the diagram below.

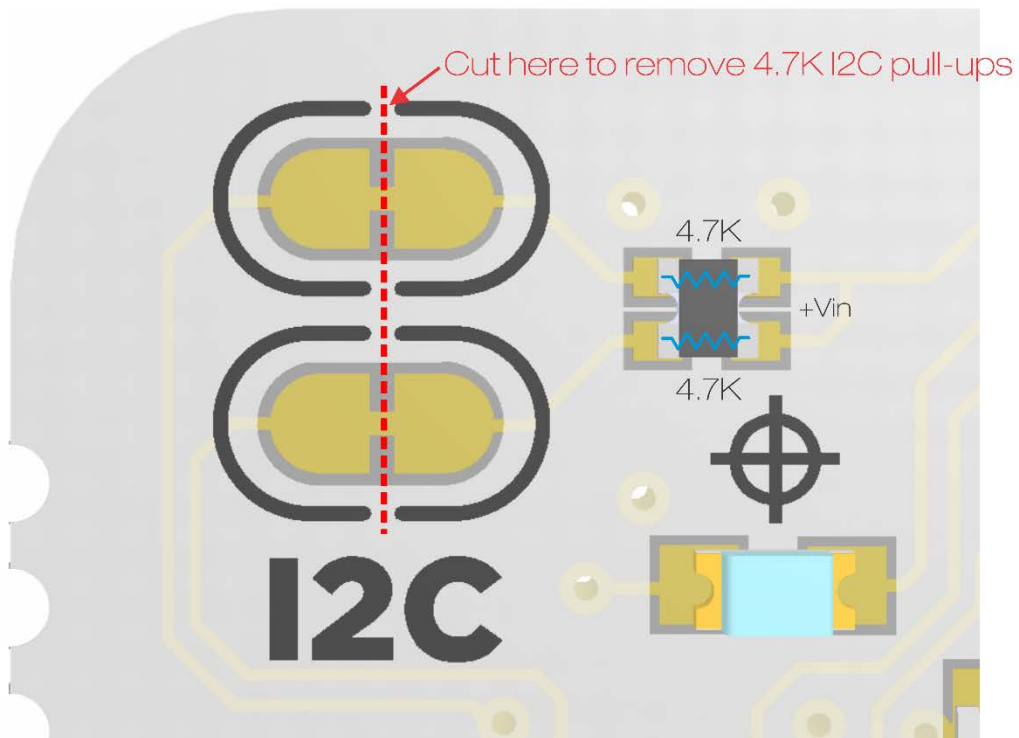


Serial parameters as used on TeraTerm are: 115200 baud, 7data bits, no parity, 1 stop bit, and no flow control.

Sample output is shown below:

```
> tinyLiDAR FW: 1.3.8
> ST PAL API: 1.0.2
> WatchDog: on
> LED: measurement
> I2C Address: 0x10
> Presets: tinyLiDAR
  SignalLimit 0.10 Mcps
  SigmaLimit 60 mm
  TimingBudget 18 ms
  PreRangeVcSELPeriod 18
  FinalRangeVcSELPeriod 14
> Offset Cal: Default = 53 mm
> Xtalk Cal: Default
> Ready
```

**Note:** The I2C pull-up resistors may be removed from circuit by cutting the top layer traces in the area shown below.



## I2C commands for tinyLiDAR

### **D** Read Distance

Read distance in mm from the most recent measurement

### **E** LED Disable

Disable the on-board Blue LED indicator.

### **F** LED Enable

This will make the LED blink for each measurement taken.

### **G** LED On

This will turn on the LED and keep it on regardless of measurements.

### **MC** Continuous mode

Set tinyLiDAR to continuous conversion mode. This is the lowest latency and highest power mode of the module. Note that this command will cause tinyLiDAR to write this mode to non-volatile memory and reboot. Therefore please allow some time for it to reboot.

### **MS** Single Step mode

Set tinyLiDAR to single step conversion mode. This is the lowest power mode of the module where it will take a measurement on demand only (using the D command) and stay in the lowest power state otherwise. Note that this command will cause tinyLiDAR to write this mode to non-volatile memory and reboot. Therefore please allow some time for it to reboot.

### **R** Change I2C address

Change the I2C address for tinyLiDAR. This new address will be written to tinyLiDAR's non-volatile memory and take effect immediately. You can reset the address by using the RESET command.

### **X** Reboot only

Reboot tinyLiDAR immediately without writing any settings to non-volatile storage. Please allow some time for it to reboot.

### **Y** LED mode save with reboot

Same as the X command but also writes the LED indicator mode to non-volatile storage. Note that this command will cause tinyLiDAR to write this mode to non-volatile memory and reboot. Therefore please allow some time for it to reboot.

### **RESET** Reset to factory defaults

Reset tinyLiDAR to the default board address of 0x10 and clear user settings. Please note, this command uses a special broadcast address of 0x00 to talk to all tinyLiDAR modules connected to the I2C bus. After writing the default address to the non-volatile memory and resetting configuration settings to factory defaults, the module will reboot. Therefore please allow some time for it to reboot.

### **W** Write custom VL53L0X config

Write new VL53L0X configuration parameters.

Parameters are written in the following order: Signal Limit, Sigma Limit, Time Budget, VCSEL Period selection. This command requires a sequence of steps which are best understood by following the code from the tinyLiDAR Terminal GUI sketch shown in Appendix A.

## I2C commands for tinyLiDAR (cont'd)

### **P\_** Preset Ranging mode

Set tinyLiDAR to one of 4 Ranging modes: L,S,H,T as described below.

Note that this command will cause tinyLiDAR to write this mode to non-volatile memory and reboot. Therefore please allow some time for it to reboot.

**PL:** long range mode (up to 2m, 33ms)

**PS:** high speed mode (up to 1.2m, 20ms)

**PH:** high accuracy mode (up to 1.2m, 200ms)

**PT:** tinyLiDAR mode (up to 2m, 18ms)

### **CD** Auto-Calibrate Distance Offset

Perform Offset Distance Calibration on tinyLiDAR.

Before using this calibration command, please set tinyLiDAR to **Continuous, High Accuracy** mode by issuing the commands "MC" and "PH". See example code in Appendix A for details.

ST recommends to use a distance of **100mm** to a white target. Therefore, place a white target 100mm away from tinyLiDAR before running this calibration.

Must specify calibration distance in mm.

The new offset correction distance will be placed in non-volatile memory and used for all subsequent operations. This calibration takes about 10 seconds to run and the LED will flash slowly during the calibration. You can reset to our factory defaults by executing the "RESET" command.

### **CX** Auto-Calibrate Crosstalk

Perform Crosstalk Calibration on tinyLiDAR. This correction factor is typically not used as it is meant to try and correct for crosstalk from thin cover windows placed in front of the sensor. The tinyLiDAR module does not ship with a cover window hence it does not require crosstalk correction by default.

Before using this calibration command, please set tinyLiDAR to **Continuous, High Accuracy** mode by issuing the commands "MC" and "PH".

ST recommends to use a distance of 400mm to a gray (17% reflectivity) target. Therefore, place a gray target 400mm away from tinyLiDAR before running this calibration.

Must specify calibration distance in mm.

The crosstalk factor will be placed in non-volatile memory and used for all subsequent operations. This calibration takes about 10seconds to run and the blue LED will flash slowly during the calibration. You can clear this calibration out by running the "RESET" command.

### **WatchDog Timer**

The hardware WatchDog Timer will automatically reboot tinyLiDAR if no I2C bus activity is seen in approx 3 seconds. This feature is essential for long term reliability as the watchdog timer is an independent circuit inside of the microcontroller and can automatically reset itself if the code locks up for any reason. When the watchdog timer triggers it will print a period on the serial log terminal output and reboot the tinyLiDAR sensor quickly. The default mode is with the watchdog timer enabled but you can disable it using the T0 command and re-enable it with the T1 command.

### **T0** Disable WatchDog Timer

This command will turn off the watchdog timer in the non-volatile settings and cause a reboot. Therefore please allow some time for it to reboot.

### **T1** Enable WatchDog Timer

This command will turn on the watchdog timer in the non-volatile settings and cause a reboot. Therefore please allow some time for it to reboot.

## I2C commands for tinyLiDAR (cont'd)

### Q Query Settings

This command will provide the current tinyLiDAR module settings:

- Current Operation Mode: {single step or continuous}
- WatchDog Timer: {on/off}
- LED Indicator: {on/off/measurement}
- Current Preset Configuration: {HighSpeed/LongRange/HighAccuracy/tinyLiDAR/Custom}
- Signal Rate Limit (in MCPS)
- Sigma Estimate Limit (in mm)
- Timing Budget (in ms)
- Pre Range VCSEL Period
- Final Range VCSEL Period
- tinyLiDAR Firmware Version
- ST PAL API Version
- Offset Cal: {custom or default}
- Cal Offset (in mm)
- Crosstalk (in MCPS)

### AR Auto-Set I2C Addresses

This command will set the connected tinyLiDAR(s) in a special mode where it's possible to set their I2C addresses by simply placing a finger in front of them. The sequence for this command is best performed with the Arduino Terminal GUI sketch provided in Appendix A.

### V Verify connected I2C addresses of the connected tinyLiDAR units

This command was designed as a companion to the AR command above. It will search for and find all connected tinyLiDAR modules. It will then make their LEDs blink in sequence showing the addresses on the terminal. Again, this sequence is best performed with the Arduino Terminal GUI sketch provided in Appendix A.



## I2C commands for tinyLiDAR (cont'd)

### **ML** Single Step Ultra Low Power mode (available since FW: 1.3.8)

Set tiny**LiDAR** to single step Ultra Low Power conversion mode. This is a new **lower** power mode of the module where it will take a measurement on demand only (using the D command) and stay in the lowest power state otherwise. This particular mode was designed for use in slower sampling rate, battery powered sensors.

Therefore, when using this mode, please be aware that the minimum time between samples has been relaxed to 30ms vs 13ms as was in the normal SS mode.

This command will also cause tiny**LiDAR** to write the mode to non-volatile memory and reboot. Hence please allow some time for it to reboot.

### **A** Autonomous mode start (available since FW: 1.3.8)

Set tiny**LiDAR** to a host controller-less Single Step Ultra Low Power conversion mode. This is a new **lower** power mode of the module where it will take a measurement continually at a specified cadence and stay in the lowest power state otherwise. A trigger threshold distance must be programmed before this mode can be started. Once an object is closer than this trigger distance, the logic output available on the board as shown in the diagram below will be set to “high” for a programmable duration. No I2C controller board such as an Arduino nor Raspberry Pi is needed for operation once tiny**LiDAR** is in this mode. The I2C bus will be inoperable during this mode, however the serial TTY output port will still be running.

### **AZ** Autonomous mode end (available since FW: 1.3.8)

Stops the tiny**LiDAR** autonomous mode and enables the I2C port again for normal host controller based operation. This command is provided in the Arduino Terminal GUI sketch version 1.1

## D Read Distance

Read distance in mm from the most recent measurement

### Host -> Sensor

Send Command

0x44 {D}

### Sensor -> Host

Read Response

2 bytes with MSB first

### Example:

Read from distance from tiny**LiDAR** at default address of 0x10.  
Value read was 0x0690 or 1680mm.

```
<10:w> 44
```

```
<10:r> 06 90
```

## E LED Disable

Disable the on-board Blue LED indicator.

### Host -> Sensor

Send Command

0x45 {E}

### Example:

Disable the LED indicator on tiny**LiDAR** at default address of 0x10.

```
<10:w> 45
```

### **F** LED Enable

This will make the LED blink for each measurement taken.

#### **Host -> Sensor**

Send Command  
0x46 {F}

#### **Example:**

Enable the LED indicator on tiny**LiDAR** at default address of 0x10.

```
<10:w> 46
```

### **G** LED On

This will turn on the LED and keep it on regardless of measurements.

#### **Host -> Sensor**

Send Command  
0x47 {G}

#### **Example:**

Set the LED to be on.

```
<10:w> 47
```

### **X** Reboot only

Reboot tiny**LiDAR** immediately without writing any settings to non-volatile storage. Please allow some time for it to reboot.

#### **Host -> Sensor**

Send Command  
0x58 {X}

#### **Example:**

Reboot now.

```
<10:w> 58
```

### Y LED mode save with reboot

Same as the X command but also writes the LED indicator mode to non-volatile storage. Note that this command will cause tinyLiDAR to write this mode to non-volatile memory and reboot. Therefore please allow some time for it to reboot.

#### Host -> Sensor

Send Command

0x59 {Y}

#### Example:

Save LED mode and reboot now.

```
<10:w> 59
```

### RESET Reset command

Reset tinyLiDAR to the default board address of 0x10 and clear user settings. Please note, this command uses a special broadcast address of 0x00 to talk to all tinyLiDAR modules connected to the I2C bus. After writing the default address to the non-volatile memory and resetting configuration settings to factory defaults, the module will reboot. Therefore please allow some time for it to reboot.

#### Host -> Sensor

Send Command

0x00, 0x06

#### Example:

Reset tinyLiDAR to factory defaults.

```
<10:w> 00 06
```

## MC Continuous mode

Set tiny**LiDAR** to continuous conversion mode. This is the lowest latency and highest power mode of the module. Note that this command will cause tiny**LiDAR** to write this mode to non-volatile memory and reboot. Therefore please allow some time for it to reboot.

### Host -> Sensor

Send Command

0x4d {M}, 0x43 {C}

### Example:

Set tiny**LiDAR** to continuous mode at default address of 0x10.

```
<10:w> 4d 43
```

### MS Single Step mode

Set tinyLiDAR to single step conversion mode. This is the lowest power mode of the module where it will take a measurement on demand only (using the D command) and stay in the lowest power state otherwise. Note that this command will cause tinyLiDAR to write this mode to non-volatile memory and reboot. Therefore please allow some time for it to reboot.

#### Host -> Sensor

Send Command

0x4d {M}, 0x53 {S}

#### Example:

Set tinyLiDAR to single step mode at default address of 0x10.

```
<10:w> 4d 53
```

### I Change Terminal's I2C address

Change the I2C address for the Arduino based Terminal GUI. This address will be used as the new target address to talk to the connected tiny**LiDAR** module.

#### **Example:**

Set the terminal to address the new target at I2C address of 0x22.

Enter the following on the terminal screen:

```
i 22
```

### R Change I2C address

Change the I2C address for tiny**LiDAR**. This new address will be written to tiny**LiDAR**'s non-volatile memory and take effect immediately. You can reset the address by using the RESET command.

#### **Host -> Sensor**

Send Command

0x52 {R}, Desired New I2C Address

#### **Example:**

Set the tiny**LiDAR** connected at address 0x10 to the new I2C address of 0x22.

```
<10:w> 52 22
```

## W Write custom VL53LOX configuration

Write new VL53LOX configuration parameters. Parameters are written in the following order: Signal Limit, Sigma Limit, Time Budget, VCSEL Period selection. This command requires a sequence of steps which are best understood by following the code from the tinyLiDAR Terminal GUI sketch shown in Appendix A.

### Host -> Sensor

Send Command  
0x57 {W},

Signal Rate Limit in MCPS written as 100x the required value, {required range is 0.00 to 65.00MCPS so have to write 0 to 6500 here}

Sigma Limit in mm,

Timing Budget in milliseconds (ms), {range is 20ms to 2000ms }

VCSEL Period selection, {choose either 18 or 14 here to signify which pair of values you want to choose}

**18 means:**

“Pre-range VCSEL Period = **18** and Final-range VCSEL Period = 14”

**14 means:**

“Pre-range VCSEL Period = **14** and Final-range VCSEL Period = 10”

### Example:

Set tinyLiDAR at the default address of 0x10 to the “High Accuracy” configuration parameters:

Signal Limit = 0.25 MCPS, so write 100x 0.25 = 25 which is 0x0019

Sigma Limit = 18 mm, so write 0x12

Time Budget = 200 ms, so write 0x00c8

Pre-range VCSEL Period = 14, Final-range VCSEL Period = 10, so write 0x0e

```
<10:w> 57 00 19 12 00 c8 0e
```



## CD Auto-Calibrate Distance Offset

Perform Offset Distance Calibration on tinyLiDAR.

Before using this calibration command, please set tinyLiDAR to **Continuous, High Accuracy** mode by issuing the commands “MC” and “PH”.

ST recommends to use a distance of 100mm to a white (88% reflectivity) target. Therefore, place a white target 100mm away from tinyLiDAR before running this calibration.

Must specify calibration distance in mm.

The new offset correction distance will be placed in non-volatile memory and used for all subsequent operations. This calibration takes about 10 seconds to run and the blue LED will flash slowly during the calibration. You can reset to our factory defaults by executing the “RESET” command.

---

Before using this calibration command, please set tinyLiDAR to **High Accuracy, Continuous** mode by issuing the commands “PH” and “MC”.

---

### Host -> Sensor

Send Command

0x43 {C}, 0x44 {D}, Distance to Cal Target in mm

### Example:

Run the offset distance calibration for the tinyLiDAR connected at the default address of 0x10 with a target exactly 100mm away:

```
<10:w> 43 44 00 64
```

## CX Auto-Calibrate Crosstalk

Perform Crosstalk Calibration on tiny**LiDAR**. This correction factor is typically not used as it is meant to try and correct for crosstalk from thin cover windows placed in front of the sensor. The tiny**LiDAR** module does not ship with a cover window hence it does not require crosstalk correction by default.

Before using this calibration command, please set tiny**LiDAR** to **Continuous, High Accuracy** mode by issuing the commands “MC” and “PH”.

ST recommends to use a distance of 400mm to a gray (17% reflectivity) target. Therefore, place a gray target 400mm away from tiny**LiDAR** before running this calibration.

Must specify calibration distance in mm.

The crosstalk factor will be placed in non-volatile memory and used for all subsequent operations. This calibration takes about 10seconds to run and the blue LED will flash slowly during the calibration. You can clear this calibration out by running the “RESET” command.

---

Before using this calibration command, please set tiny**LiDAR** to **High Accuracy, Continuous** mode by issuing the commands “PH” and “MC”.

---

### Host -> Sensor

Send Command

0x43 {C}, 0x58 {X}, Distance to Cal Target in mm

### Example:

Run Crosstalk calibration for the tiny**LiDAR** connected at the default address of 0x10 with a target exactly 400mm away:

```
<10:w> 43 58 01 90
```

## P\_ Preset Ranging mode

Set tiny**LiDAR** to one of 4 Ranging modes: L,S,H,T as described below.  
Note that this command will cause tiny**LiDAR** to write this mode to non-volatile memory and reboot. Therefore please allow some time for it to reboot.

You can also set to custom formats using the “W” command.

PL: long range mode (up to 2m, 33ms)  
PS: high speed mode (up to 1.2m, 20ms)  
PH: high accuracy mode (up to 1.2m, 200ms)  
PT: tiny**LiDAR** mode (up to 2m, 18ms)

### Host -> Sensor

Send Command

0x50 {P},

And then 0x4c {L}, 0x53 {S}, 0x48 {H}, or 0x54 {T},

### Example:

Set the tiny**LiDAR** connected at the default address of 0x10 to “High Accuracy” mode by writing P H:

```
<10:w> 50 48
```

## Q Query Settings

This command will provide the current tiny**LiDAR** module settings:

- Current Operation Mode: {single step or continuous}
- WatchDog Timer: {on/off}
- LED Indicator: {on/off/measurement}
- Current Preset Configuration: {HighSpeed/LongRange/HighAccuracy/tiny**LiDAR** /Custom}
- Signal Rate Limit (in MCPS)
- Sigma Estimate Limit (in mm)
- Timing Budget (in ms)
- Pre Range VCSEL Period
- Final Range VCSEL Period
- tiny**LiDAR** Firmware Version
- ST PAL API Version
- Offset Cal: {custom or default}
- Cal Offset (in mm)
- Crosstalk (in MCPS)

### Host -> Sensor

Send Command

0x51 {Q}

### Sensor -> Host

Read Response

[0] 0x43 for continuous or 0x53 for single step mode

[1] 0x53 for High Speed, 0x52 for Long Range, 0x41 for High Accuracy, 0x43 for Custom or 0x54 for tiny**LiDAR**, else is Unknown preset configuration.

[2][3] for Signal Rate Limit. Scale down by 65536 to get proper value in MCPS. Note that we are limited to a readout max value of 0.99MCPS due to the 16bit read but the serial log terminal output is not limited.

[4] for Sigma Estimate Limit. In mm.

[5][6] for Timing Budget in ms.

[7] for 0x0e or 0x12 for 14/10 or 18/14 VCSEL period settings.

[8][9][10] for the tiny**LiDAR** firmware version number.

[11][12][13] for the ST PAL API version number.

[14] bit 3 for Offset Cal flag. Set means “custom” else “default” settings being used currently by tiny**LiDAR**.

[14] bits 2 and 1 for LED Indicator mode.

- 00 means 'OFF'.
- 01 means 'ON'.
- 10 means 'Measurement' mode.

[14] bit 0 for WatchDog Timer flag. Set means “ON” else “OFF”.

[15][16][17][18] for Offset Cal value. Scale down by 1000 for proper value in mm.

[19][20][21][22] for Crosstalk Cal value. Scale down by 65536 for proper value in MCPS.

**Example:**

Query tiny**LiDAR** connected at the default address of 0x10.

For this example, we have

0x53 = SingleStep mode

0x54 = tiny**LiDAR** preset

0x1999 = Signal Limit read as 6553 so that is  $6553/65536 = 0.1$  MCPS

0x3c = Sigma Limit is 60mm

0x0012 = Timing Budget which is 18ms

0x12 = Pre Range VCSEL Period of 18 and so Final Range VCSEL Period = 14

0x01 0x03 0x07 = tiny**LiDAR** f/w version 1.3.7

0x01 0x00 0x02 = ST PAL API version = 1.0.2

0x05 = 0101 so this means “default” offset cal being used,

Measurement mode for LED setting, and WDT is ON.

0x00 0x00 0xcF 0x08 = 53000 so that is  $53000/1000 = 53$ mm offset

0x00 0x00 0x00 0x00 = 0 for the Crosstalk cal.

```
<10:w> 51
```

```
<10:r> 53 54 19 99 3c 00 12 12 01 03 07 01 00
      02 05 00 00 CF 08 00 00 00 00
```

### **T0** Disable WatchDog Timer

This command will turn off the watchdog timer in the non-volatile settings and cause a reboot. Therefore please allow some time for it to reboot.

#### **Host -> Sensor**

Send Command  
0x54 {T} 0x30 {0}

#### **Example:**

Turn off the WDT.

```
<10:w> 54 30
```

### **T1** Enable WatchDog Timer

This command will turn on the watchdog timer in the non-volatile settings and cause a reboot. Therefore please allow some time for it to reboot.

#### **Host -> Sensor**

Send Command  
0x54 {T} 0x31 {1}

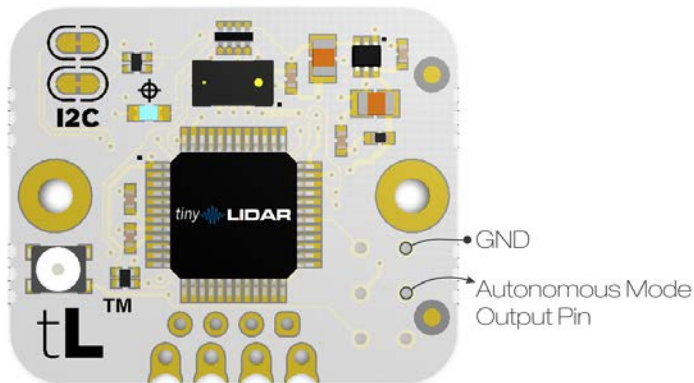
#### **Example:**

Turn off the WDT.

```
<10:w> 54 31
```

### A Autonomous mode start (available since FW: 1.3.8)

Set tinyLiDAR to a host controller-less Single Step Ultra Low Power conversion mode. This is a new **lower** power mode of the module where it will take a measurement continually at a specified cadence and stay in the lowest power state otherwise. A trigger threshold distance must be programmed before this mode can be started. Once an object is closer than this trigger distance, the logic output available on the board as shown in the diagram below will be set to “high” for a programmable duration. No I2C controller board such as an Arduino nor Raspberry Pi is needed for operation once tinyLiDAR is in this mode. The I2C bus will be inoperable during this mode, however the serial TTY output port will still be running.



*This command requires a sequence of steps which are best applied from the tinyLiDAR Terminal GUI sketch shown in Appendix A.*

**Limits:** 10 to 2000mm for low and high side limits

**Interval:** 1 to 100 for 0.1sec to 10sec max

**LED Indicator:** 1 for ON, 0 for OFF

#### **Example:**

Set to detect object in 10 to 200mm range and output a pulse width of 1sec with LED indicator showing when measurements are taken and when it is triggered. Take measurements every 3 seconds. Note: measurement type should be setup prior to this (i.e. high accuracy mode, etc.)

Enter command on Terminal GUI as:

```
A 10 200 30 10 1
```

Command 'A': Autonomous Mode.

```
The values you entered were -  
  Low side limit = 10mm  
  High side limit = 200mm  
  Repetition interval = 30 for 3000ms  
  Pulse Width = 10 for 1000ms  
  LED indicator = ON
```

### AZ Autonomous mode end (available since FW: 1.3.8)

Stops the tiny**LiDAR** autonomous mode and enables the I2C port again for normal host controller based operation. This command is provided in the Arduino GUI (version 1.1)

*This command requires a sequence of steps which are best applied from the tiny**LiDAR** Terminal GUI sketch shown in Appendix A.*

#### Additional Steps:

To bring the tiny**LiDAR** board out of this autonomous mode you need to issue the AZ command (via the I2C bus) while also pressing the RESET button on the tiny board. The proper sequence is as follows:

- 1) Type AZ on the Terminal GUI but do not press enter yet.
- 2) Click and hold the RESET button on the tiny**LiDAR** board.
- 3) Press enter for the GUI Terminal to execute the command while releasing the RESET button on the tiny**LiDAR** board. A counter is shown counting down from 10 to 0. You must release the RESET button on the board before 0 is reached on this counter.
- 4) If successful, you will see the LED on tiny**LiDAR** turn on solid for about 3 seconds to show it understood your AZ command. If it didn't work then please repeat from step 1.
- 5) At this point the board is functioning as normal and can be controlled by the I2C bus again.

### ML Single Step Ultra Low Power mode (available since FW: 1.3.8)

Set tiny**LiDAR** to single step Ultra Low Power conversion mode. This is a new **lower** power mode of the module where it will take a measurement on demand only (using the D command) and stay in the lowest power state otherwise. This particular mode was designed for use in slower sampling rate, battery powered sensors. Therefore, when using this mode, please be aware that the minimum time between samples has been relaxed to 30ms vs 13ms as was in the normal SS mode.

This command will also cause tiny**LiDAR** to write the mode to non-volatile memory and reboot. Hence please allow some time for it to reboot.

#### Host -> Sensor

Send Command

0x4d{M} 0x4c {L}

#### Example:

Set to ML ULP mode.

```
<10:w> 4d 4c
```



## Technical Specifications

### Physical

Size: 21mm x 25mm x 8.3mm

Weight: approximately <1.5grams

Mounting: 2x plated holes which are 2.0mm in diameter and spaced 20mm apart (both are connected to the circuit ground). Designed for mounting with M2 type screws.

### Operating Temperature

-20 to +70C

### Power Consumption

Operation Voltage: 3v to 5v DC (2.8v operation is possible with a regulated voltage source)

Quiescent Current: <3uA measured at 2.8v supply in ULP single step, tiny**LiDAR** mode

Average Current: 24mA measured at 3.3v supply in continuous, tiny**LiDAR** mode

### Bus Interface

I2C Rate: 100Kbps fixed

I2C Levels: 3v to 5v (i.e. same as supply voltage)

I2C Pull-up resistors: 2x 4.7K SMD (user accessible PCB traces on top layer can be cut if desired)

I2C Default Address: 0x10 (can change with I2C command)

### Bus Connectors

1x Right-Angled 4pin GROVE type universal connector

4x PCB edge "castellated" half-pads at 0.1inch centers

4x 0.1inch pads for header pins

### Optical

Field of View: 25degrees

940nm (IR) VCSEL Class 1 Laser {as per ST's datasheet: "The laser output is designed to remain within Class 1 laser safety limits under all reasonably foreseeable conditions including single faults in compliance with IEC 60825-1:2014 (third edition).\*" }

Activity Indicator: Blue LED (can be disabled with I2C command)

### Performance

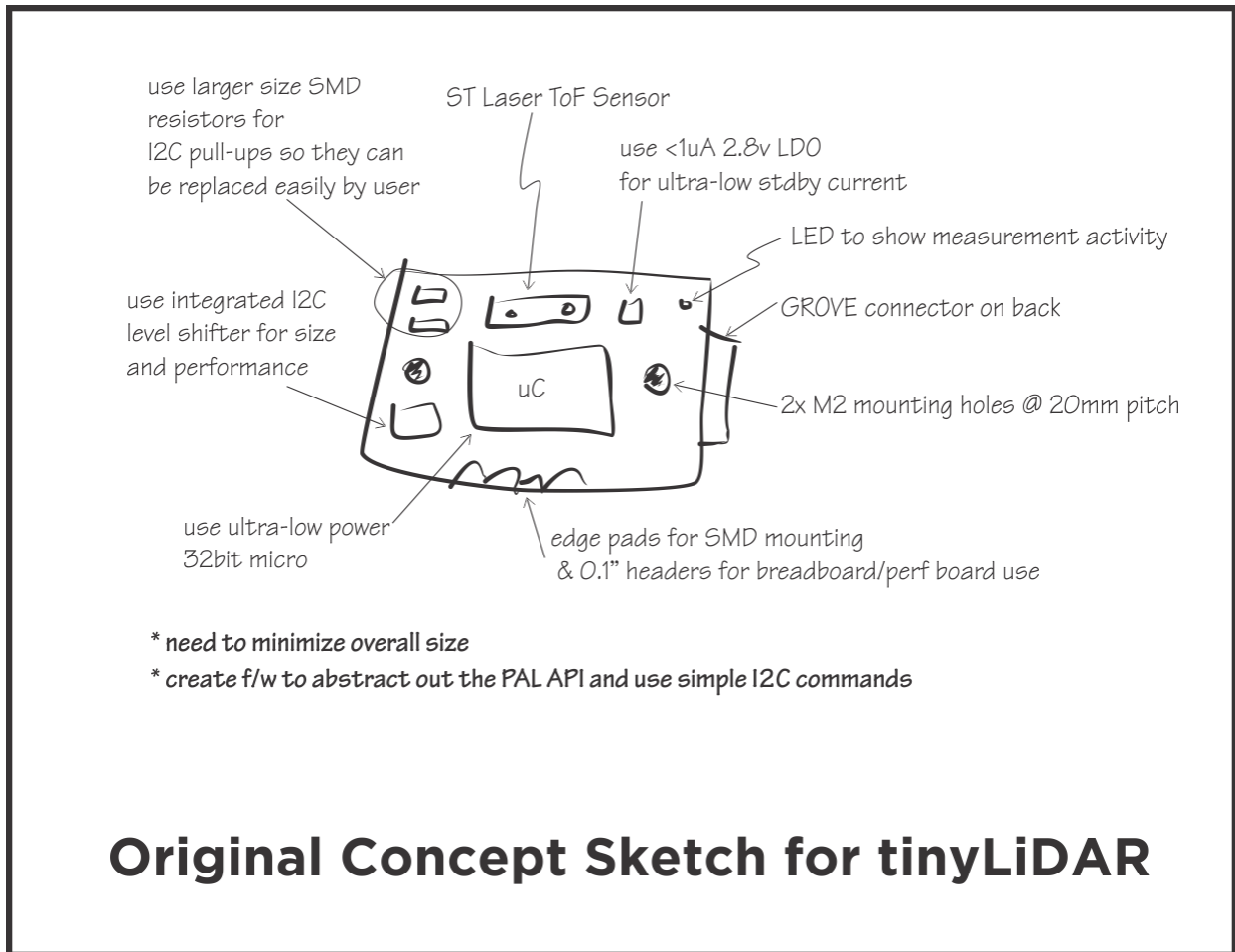
Accuracy: +/-3% typical up to 1.2m in High Accuracy mode\*

Sensing Distance: approx. 3cm to 2meters

Sample Rate: 60Hz max in continuous mode

*\* please see VL53L0X datasheet for further details.*

## Original Concept



## Original Concept Sketch for tinyLiDAR

## **Appendix A**

### Sample Arduino UNO Sketches for tiny**LiDAR**

Links to download Arduino Sketches:

[https://microelectronicdesign.s3.amazonaws.com/tinyLiDAR\\_Terminal\\_GUI\\_1\\_1.ino](https://microelectronicdesign.s3.amazonaws.com/tinyLiDAR_Terminal_GUI_1_1.ino)

<https://microelectronicdesign.s3.amazonaws.com/responseRate.ino>

<https://microelectronicdesign.s3.amazonaws.com/minimalRead.ino>

```

/*
Arduino UNO sketch to check response rate for tinyLiDAR
It will run a number of readings as fast as possible and give the effective response rate in Hz.
(showed over 930Hz in our lab testing)

Last Edit: Oct 23, 2017
Copyright (c) 2017 by Dinesh Bhatia

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>

Notes:
This code requires the "Arduino I2C Master Library rev5" library to allow the
standard I2C stretch feature to work properly on the UNO.
** Please install it before running this sketch. **
The library can be downloaded from here:
http://dsscircuits.com/articles/arduino-i2c-master-library

*/

#define samples 1000 //number of samples for each loop
#define SCL_PORT PORTC
#define SDA_PORT PORTC
#define SCL_PIN 5 //std SCL pin
#define SDA_PIN 4 //std SDA pin
#include <I2C.h>

void setup() {

  Serial.begin(115200); //setup the serial port
  I2c.begin();
  I2c.write(0x10,'M', 'C'); //send MC command for continuous mode
  delay(1000); //give time to reboot

} //setup

void loop() {

  uint16_t oldTime = 0;
  uint16_t timeNow = 0;
  uint16_t duration = 0.0;
  uint16_t i,j;

  while(1)
  {

    for (j = 0; j<samples;j++)
    {

      I2c.write(0x10,'D'); //take single reading
      I2c.read(0x10,2); // request 2 bytes from tinyLiDAR
      i = I2c.receive(); // receive 1st byte
      i = i<<8 | I2c.receive(); // receive 2nd byte and put them together
    }

    timeNow = millis(); //read the current time in milliseconds
    duration = timeNow - oldTime;
    oldTime = timeNow;
    Serial.print(F("Effective Response Rate is "));
    Serial.print( samples/(duration*.001));
    Serial.println(F("Hz "));

  } //while

} //loop
*/

```

### Arduino UNO sketch to show minimal coding required to read distance from tinyLiDAR

This program will continually print the measured distance from tinyLiDAR which is operating in its default single step mode.

Last Edit: Oct 23, 2017

Copyright (c) 2017 by Dinesh Bhatia

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>

#### Notes:

This code requires the "Arduino I2C Master Library rev5" library to allow the standard I2C stretch feature to work properly on the UNO.

\*\* Please install it before running this sketch. \*\*

The library can be downloaded from here:

<http://dsscircuits.com/articles/arduino-i2c-master-library>

\*/

```
#define SCL_PORT PORTC
#define SDA_PORT PORTC
#define SCL_PIN 5      //std SCL pin
#define SDA_PIN 4     //std SDA pin
#include <I2C.h>

void setup() {

    Serial.begin(115200);    //setup the serial port
    I2c.begin();

} //setup

void loop() {

    uint16_t i;

    while(1)
    {

        I2c.write(0x10,'D'); //take single measurement
        I2c.read(0x10,2);    // request 2 bytes from tinyLiDAR
        i = I2c.receive();   // receive MSB byte
        i = i<<8 | I2c.receive(); // receive LSB byte and put them together
        Serial.println(i);   // print distance in mm
        delay(100);         // delay as required (13ms or higher in default single step mode)

    } //while

} //loop
```