Application note

# Water and liquid level monitoring using VL53L4CD Time-of-Flight high accuracy proximity sensor

## Introduction

This document describes how a user can design a system to measure the level of a liquid in a container with ST Time-of-Flight (ToF) sensors.

The ST ToF sensors work by emitting a pulsed cone of light, which hits the target and bounces back. The sensor detects the returning signal and measures the photon travel time. The travel time divided by two and multiplied by the speed of light gives the distance. In most cases, this is a very robust system.

In this application note, the VL53L4CD ToF sensor is used to measure the level of water in a container. All ST single zone Time-of-Flight sensors (including VL53L3CX and VL53L0X) can be used in the same manner.

**Background**

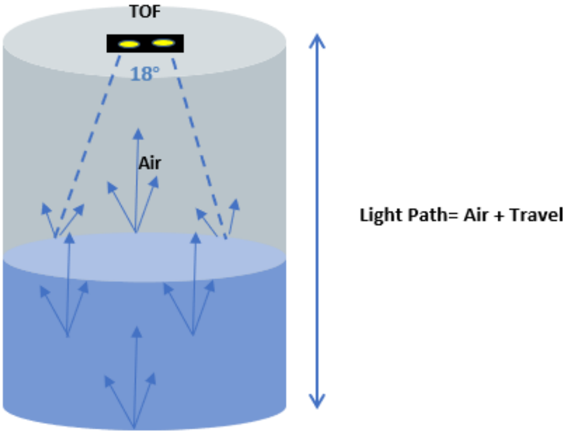
The VL53L4CD is a ToF sensor especially designed to provide very accurate distance measurements from only 1 mm up to 1300 mm. A new generation laser emitter with 18° diagonal FoV improves performances under ambient light, with ranging speed up to 100 Hz.

# 1 Observations on water

The basic principle of a ToF sensor is to measure the distance to the target based on the time for emitted photons to be reflected. In most applications, photons travel through the air. But, to measure the level of a liquid, photons travel through air and water. As a result, the light path is impacted by the different refractive indices.

In other words, there is part of the signal that is reflected from the surface, and another part returned from the container's base (traveling through the liquid), and another part bounces from the liquid itself. Thus, the final ranging distance is impacted with respect to accuracy (see Figure 1. Different light paths when ToF signal aimed at liquid). However, this document describes some techniques and methods to minimize the ranging error, using the nonlinearity correction algorithm.

Figure 1. **Different light paths when ToF signal aimed at liquid**



# 2 ST recommendations

## 2.1 Container properties

The position of the VL53L4CD sensor and the container size impact the accuracy in ranging because the sensor's field of view (FoV) is cone shaped (18° for VL53L4CD). The ToF should be mounted so that the signals are reflected only from the water surface, and not from the edge of the container.

If the container is narrow, with a diameter less than the ToF's FoV (as illustrated in Figure 2. FoV bigger than container diameter) then the signals are reflected from the edge, which induces noise. This impacts the ranging accuracy.

Therefore, ideally, the container diameter should be greater than the ToF's FoV (as illustrated in Figure 3. FoV smaller than container diameter).

Figure 2. FoV bigger than container diameter

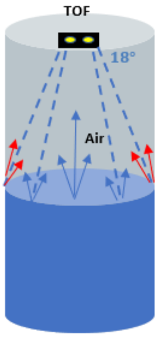
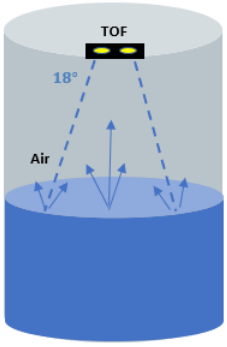


Figure 3. FoV smaller than container diameter



In addition, there should be a reasonable gap (minimum 30 mm) between the sensor and the maximum water level. This avoids any obstruction of the ToF's light path, in the case water drops bounce while the water is pouring, or if the container is shaken.

## 2.2 Container reflectivity

It is recommended to use containers with a base made of low reflective material (see Figure 4. Low reflective base) rather than fully transparent or high reflective (see Figure 5. Transparent or high reflective base). This avoids receiving mixed signals from base and liquid surface when the liquid level is low. Figure 19. Water level measurement with different colored base and no compensation illustrates the ranging behavior with different background base.

Figure 4. Low reflective base

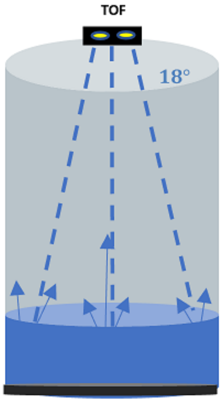
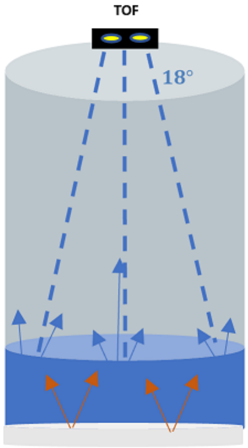


Figure 5. Transparent or high reflective base



# 3 ST solution

## 3.1 Nonlinearity correction algorithm

There are many containers with different shapes and sizes available on the market, and it is not possible to test them all. Therefore, if the ranging result is nonlinear, then the nonlinearity correction algorithm can be used to improve the ranging accuracy.

Figure 7. Nonlinear ranging results, shows that the ranging result is higher than expected when the water level is at the minimum, is steady in the middle range, and decreases when the water level is at the maximum. Hence, the offset mechanism can help reduce the nonlinearity of the results.

As the algorithm works in a defined setup, the input to the algorithm is the total distance of the container, which is stored in the host memory. This algorithm maintains a range of distance in %. The reason the distance is calculated in percentage is that liquid level application shows level of indicators that can be easily correlated with % value. For example if a container's height is 1000 mm, and there are five levels of indicators, the indicator is incremental of 200 mm.

At the start of the program, it calculates the distance based on a list of percentages. As the water level starts increasing, the program gets the ranging distance, and compares it to the actual distance it has stored. By iterating this experiment with different water levels, the error rate that needs to be offset later is identified. Finally, a list of error rates corresponding to the water level is established. These error rates are compensated by the offset values that need to be programmed in an algorithm to improve the ranging accuracy. Figure 6. Level indicators based on water level explains how the zones are divided based on water level and apply the offset values to reduce nonlinear impact on distance.

Figure 6. **Level indicators based on water level**

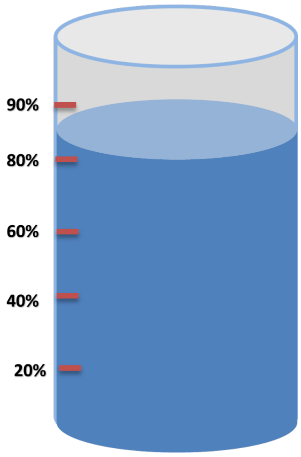
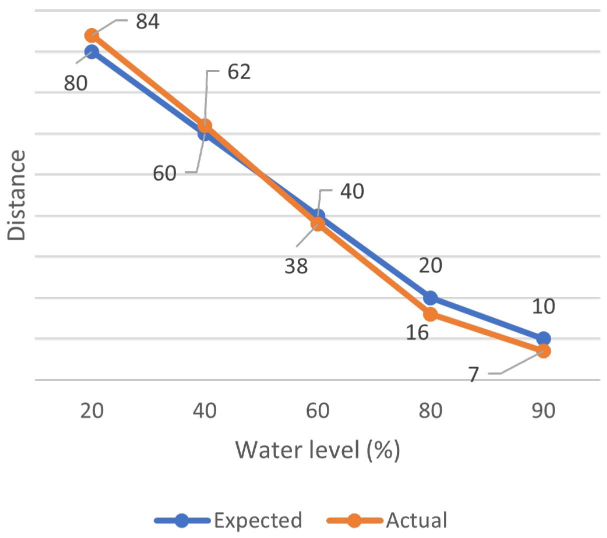


Figure 7. Nonlinear ranging results



### Example

If a container’s height is 1000 mm and it has five indicator levels. The distance corresponds to the indicators: 200 mm, 400 mm, 600 mm, 800 mm, and 900 mm. These correspond to 20%, 40%, 60%, 80%, and 90%.

When the water level starts pouring, note the ranging distance at each level.

Table 1. Ranging results example

| Water level in % | Expected (mm) | Actual (mm) | Error in % |
|---|---|---|---|
| 20 | 800 | 840 | 5 |
| 40 | 600 | 620 | 3 |
| 60 | 400 | 380 | 5 |
| 80 | 200 | 160 | 20 |
| 90 | 100 | 70 | 30 |

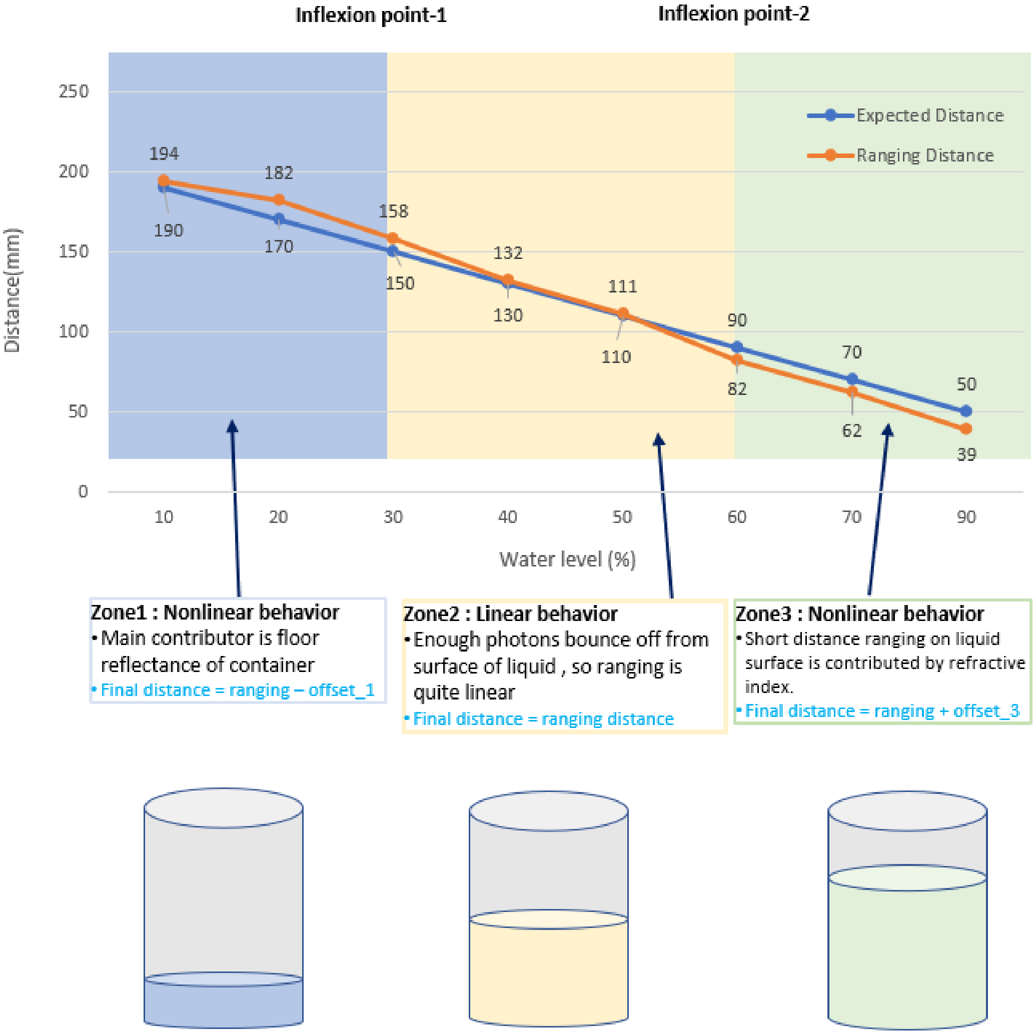
You can reiterate the test to get a clearer view of the deviation in ranging with regard to the water level.

Apply these ranging errors to the algorithm to improve the ranging accuracy. For example, if the program gets 840 mm in 20% water level, then apply 5% offset to the algorithm to minimize the error.

## 3.2 Ranging result characterization

Figure 8. Nonlinear impact on ranging result illustrates the characterization of the ranging result on the different liquid levels. This experiment assumes a transparent plastic jar, with a width of 100 mm and height of 189 mm. The measurement points are broadly divided into three zones (low, medium, and high) to define the ranging behavior within the different zone. The ranging curve changes at the inflexion points.

Figure 8. Nonlinear impact on ranging result



Zone1 : Nonlinear behavior
- Main contributor is floor reflectance of container
- Final distance = ranging – offset_1

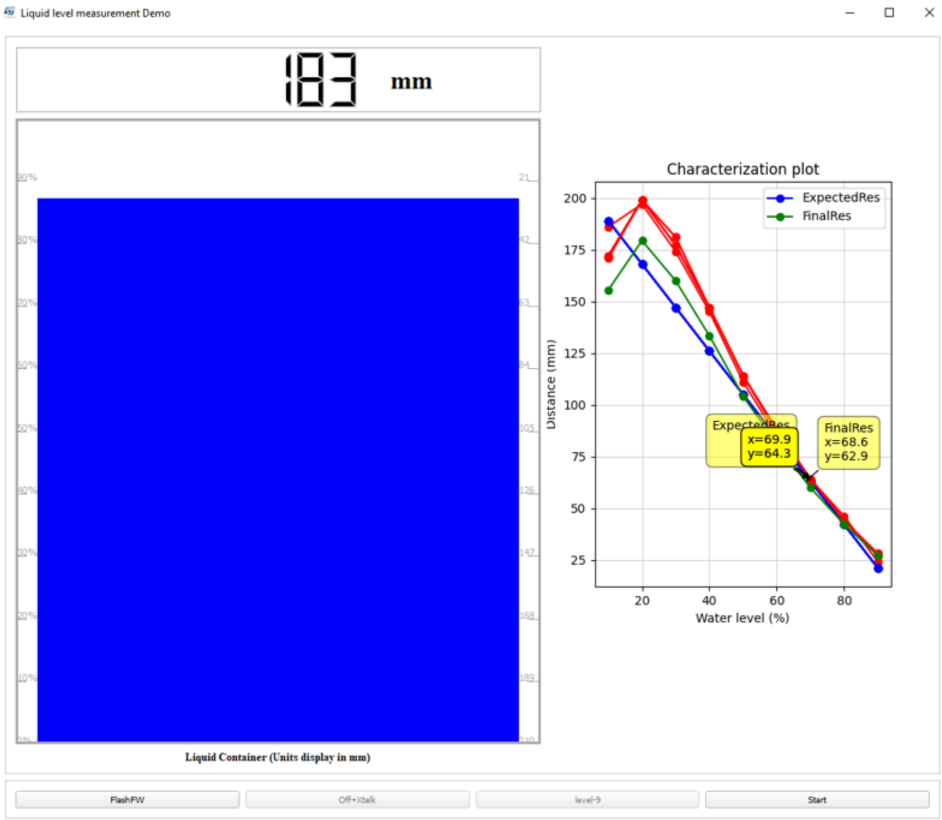
Zone2 : Linear behavior
- Enough photons bounce off from surface of liquid , so ranging is quite linear
- Final distance = ranging distance

Zone3 : Nonlinear behavior
- Short distance ranging on liquid surface is contributed by refractive index.
- Final distance = ranging + offset_3

## 3.3 System characterization using ST tool

ST provides a characterization tool (see Figure 9. Characterization tool) to better understand the signal reflectivity of the container and liquid. The tool can be found in the STSW39_L4CD package. This improves the ranging accuracy in deviation level.

Figure 9. **Characterization tool**



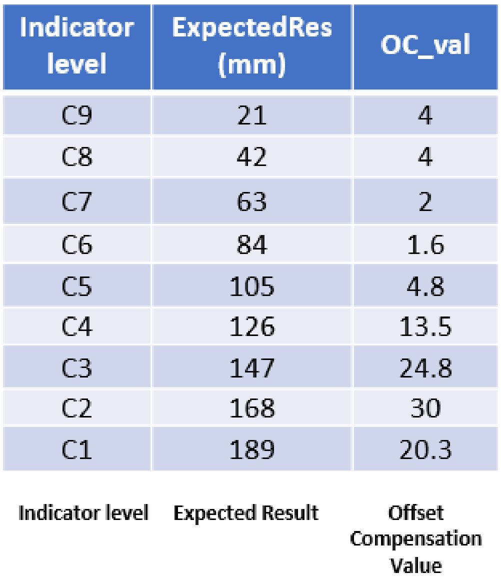
It is recommended to use a dust free cover glass on top of the container and to measure more than one similar setup (minimum 10 sets) to validate the consistency in measurement. Once the same nonlinear curve with ranging deviation is found across all setups, you can adopt a nonlinearity algorithm to reduce nonlinearity in ranging.

This algorithm uses a lookup table that is created based on noted data points and ranging deviation on different water levels. The lookup table can be easily modified based on the container's size or setup, and the data received. The only input to this algorithm is the total distance from the sensor to the container's base, which needs to be stored in the host memory (NVM). This algorithm reads this value before the start of ranging, and calculates how much offset to apply dynamically on different levels.

Alternatively, the user can apply a linear regression model to all data points to find the intercept and coefficient for the best fit line. These two parameters can be used in ranging to compensate the ranging deviation.

Following is a snippet of characterization tool; it allows the user to perform a defined number of times the characterization setup. For example, in the characterization plot below, the red lines show the characterization of the same setup and liquid three times. The green line is the mean derived from all characterized values. This can be used to create a lookup table (Figure 10. Data for lookup table).

Figure 10. **Data for lookup table**



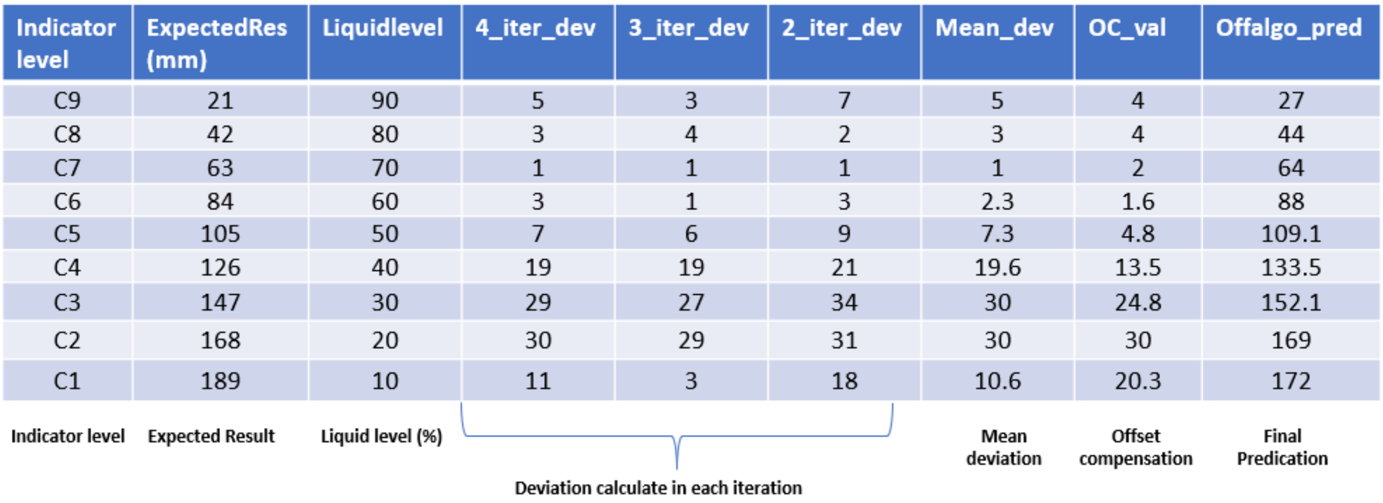
| Indicator level | ExpectedRes (mm) | OC_val |
|---|---|---|
| C9 | 21 | 4 |
| C8 | 42 | 4 |
| C7 | 63 | 2 |
| C6 | 84 | 1.6 |
| C5 | 105 | 4.8 |
| C4 | 126 | 13.5 |
| C3 | 147 | 24.8 |
| C2 | 168 | 30 |
| C1 | 189 | 20.3 |

Indicator level
Expected Result
Offset Compensation Value

The final data points are created in FinalResult.csv (Figure 11. Sample data points of characterization).

The characterization process begins by continuously pouring liquid at each level and clicking the "Characterize" button until all levels are finished. This process continues until the defined number of setups or iterations are characterized. At the end, the user can see the final characterization graph by using a nonlinearity correction algorithm. For the experiment, it includes a linear regression model as well.

Figure 11. **Sample data points of characterization**



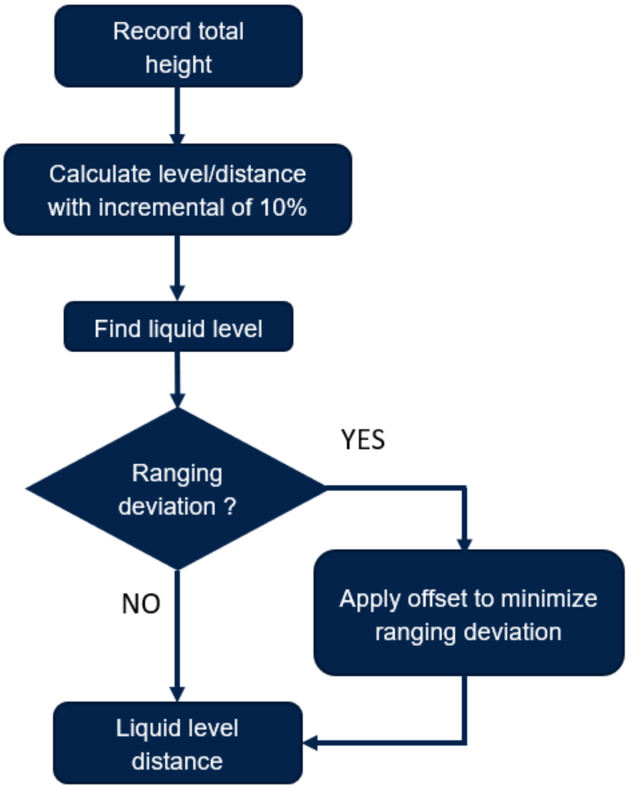
| Indicator level | ExpectedRes (mm) | Liquidlevel | 4_iter_dev | 3_iter_dev | 2_iter_dev | Mean_dev | OC_val | Offalgo_pred |
|---|---|---|---|---|---|---|---|---|
| C9 | 21 | 90 | 5 | 3 | 7 | 5 | 4 | 27 |
| C8 | 42 | 80 | 3 | 4 | 2 | 3 | 4 | 44 |
| C7 | 63 | 70 | 1 | 1 | 1 | 1 | 2 | 64 |
| C6 | 84 | 60 | 3 | 1 | 3 | 2.3 | 1.6 | 88 |
| C5 | 105 | 50 | 7 | 6 | 9 | 7.3 | 4.8 | 109.1 |
| C4 | 126 | 40 | 19 | 19 | 21 | 19.6 | 13.5 | 133.5 |
| C3 | 147 | 30 | 29 | 27 | 34 | 30 | 24.8 | 152.1 |
| C2 | 168 | 20 | 30 | 29 | 31 | 30 | 30 | 169 |
| C1 | 189 | 10 | 11 | 3 | 18 | 10.6 | 20.3 | 172 |

Indicator level
Expected Result
Liquid level (%)
Deviation calculate in each iteration
Mean deviation
Offset compensation
Final Predication

## 3.4 Nonlinearity correction algorithm flow

The following flow chart depicts the flow of the nonlinearity correction algorithm. The total height of the container is stored in the host memory and it is provided as input for the algorithm.

Figure 12. Algorithm flow


Record total height
Calculate level/distance with incremental of 10%
Find liquid level
Ranging deviation ?
YES
Apply offset to minimize ranging deviation
NO
Liquid level distance

## 3.5 Reference code for nonlinearity correction algorithm

The code snippet below determines the distances between the sensor and the liquid level, by increments of 10%.

```
/*Find water level position based on real ranging distance and apply offset value to correct deviation
Input : totaldistance= height of container 2. rangevalue = received by sensor
Output : rangevalue_out= After compensate value (og_val) which you received from characterization method applied
*/
void Liquidlevelmeasureerrorcomponsate(uint16_t totaldistance,uint16_t rangevalue,uint16_t *rangevalue_out)
{
        uint16_t pos=0;
        uint16_t i=0; uint16_t value;
        if (totaldistance ==0)
            totaldistance=TOTAL_HT_CONTAINER; // Test setup container's height
        // In this test, MAX_LABEL =10
        for (i=0;i<MAX_LABEL;i++)
        {
            if  (rangevalue <totaldistance+5)
            {
                if (rangevalue < ogalgo_data_inst[i].expected_res)
                {

                    // Check first
                    if (i >0)
                    {
                        value = (ogalgo_data_inst[i].expected_res + ogalgo_data_inst[i-1].expected_res)/2;
                    }
                    else
                        value = ogalgo_data_inst[i].expected_res;

                    if(rangevalue <= value)
                    {
                        /* Find water level position based on ranging value */
                        if (i ==0 )
                        {
                            pos=1;
                            break;
                        }
                        pos=1+(i-1); // adding one bcz position starts from 1
                        break;
                    }
                    else
                    {
                        pos=1+i;
                        break;
                    }
                }
                else
                {
                    if (rangevalue > totaldistance+5)
                        pos=99;
                    else
                        pos=6;  // Need to find which one has max OG value or store it directly here
                }
            }
            else
                pos=99;
        }
```

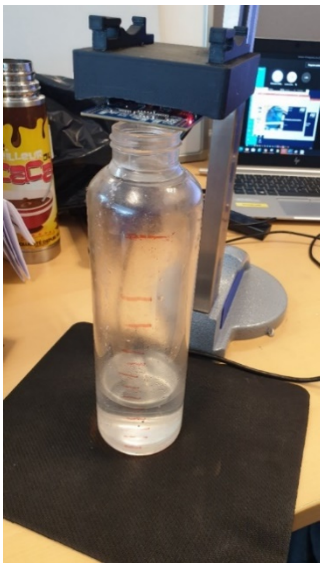
Based on the liquid level, the algorithm detects if it is over ranging or under ranging. It then uses a lookup table to apply offset to improve ranging accuracy. The offset compensation value applied depends on the environment or setup.

```
        // position = water level position, og_val= got compensated value from characterization method
        switch(pos)
        {
            case 1:
                    *rangevalue_out=rangevalue + ogalgo_data_inst[pos].og_val;  // Category Underranging,
                break;
            case 2:
                    *rangevalue_out=rangevalue + ogalgo_data_inst[pos].og_val; // Category Underranging, 30mm deviation  b/w C3&C2
                break;
            case 3:
                    *rangevalue_out=rangevalue + ogalgo_data_inst[pos].og_val; // Category Underranging, 25mm deviation b/w C4&C3
                break;
            case 4:
                    *rangevalue_out=rangevalue + ogalgo_data_inst[pos].og_val; // Category Underranging, 12mm deviation b/w C2&C2
                break;
            case 5:
                    *rangevalue_out=rangevalue - ogalgo_data_inst[pos].og_val; // Category Overranging, 4mm deviation b/w C2&C2
                break;
            case 6:
                    *rangevalue_out=(uint16_t)(rangevalue - ogalgo_data_inst[pos].og_val); // Category Overranging, 1mm deviation
                break;
            case 7:
                    *rangevalue_out=(uint16_t)(rangevalue - ogalgo_data_inst[pos].og_val); // Category Overranging, 2mm deviation
                break;
            case 8:
                    *rangevalue_out=(uint16_t)(rangevalue - ogalgo_data_inst[pos].og_val); // Category Overranging, 4mm deviation
            break;
        case 9:
                *rangevalue_out=(uint16_t)(rangevalue - ogalgo_data_inst[pos].og_val); // Category Overranging, 4mm deviation
            break;
        default:
                printf("Not meet condition");
        }
```

# 4 Test setup and results

The test setup uses a narrow transparent glass bottle, with a width of 220 mm and radius of 50 mm. The sensor is mounted slightly above the bottle, but all the measurements are relative to the initial height. The test is conducted on the water level with a timing budget of 50 ms. This application is not usually time critical, so the rolling average along with a bigger timing budget can be used to allow the ToF to accumulate more signals that ultimately enhance ranging performance.

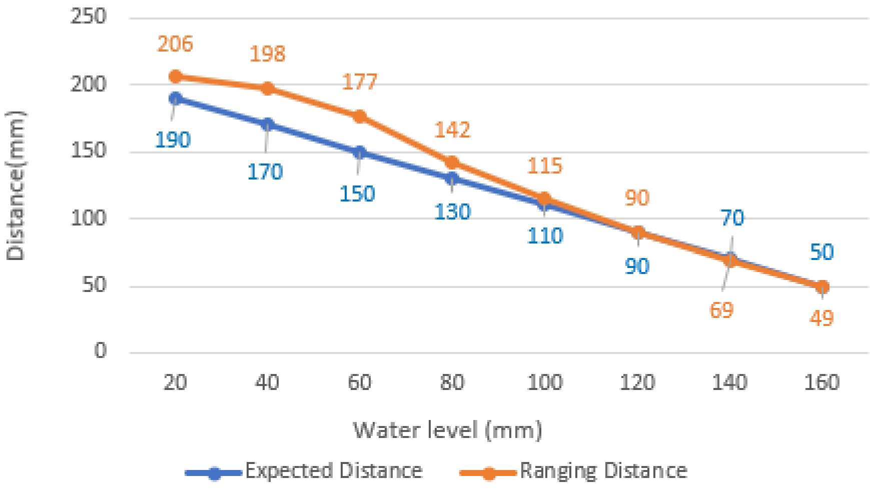
Figure 13. **Water bottle with black mat background**



## 4.1 Test results without nonlinearity correction algorithm and without cover glass

The following graph shows the ranging data with different water levels. It can be observed that there is a nonlinearity between expected and actual measurement in low water level.

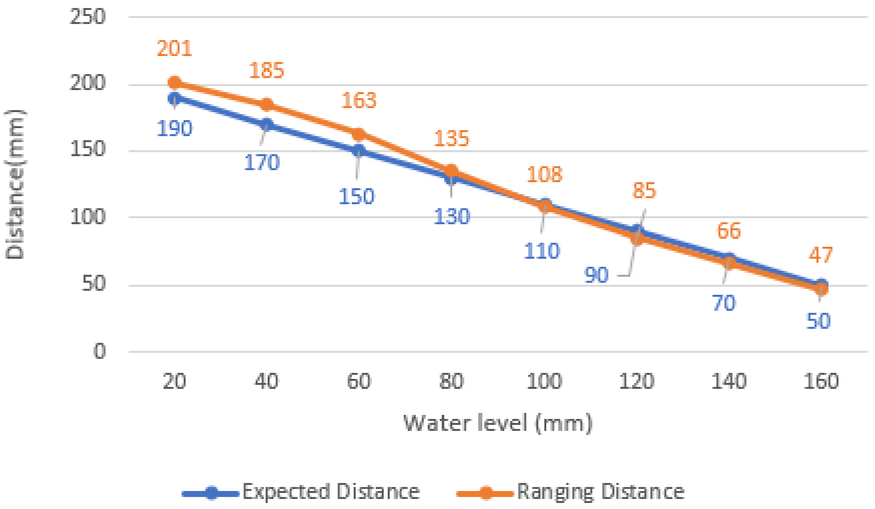
Figure 14. **Without nonlinearity algorithm and without cover glass**



## 4.2 Test results with offset calibration, without nonlinearity correction algorithm, and with cover glass

The following graph illustrates the data with offset calibration and with cover glass, but without nonlinearity correction algorithm. The offset calibration is done with the same setup, maintaining 100 mm distance from sensor to water level. The results are more accurate, yet over range persists in low water levels. Offset calibration is done using the feature available and described in the VL53L4CD UM2931.

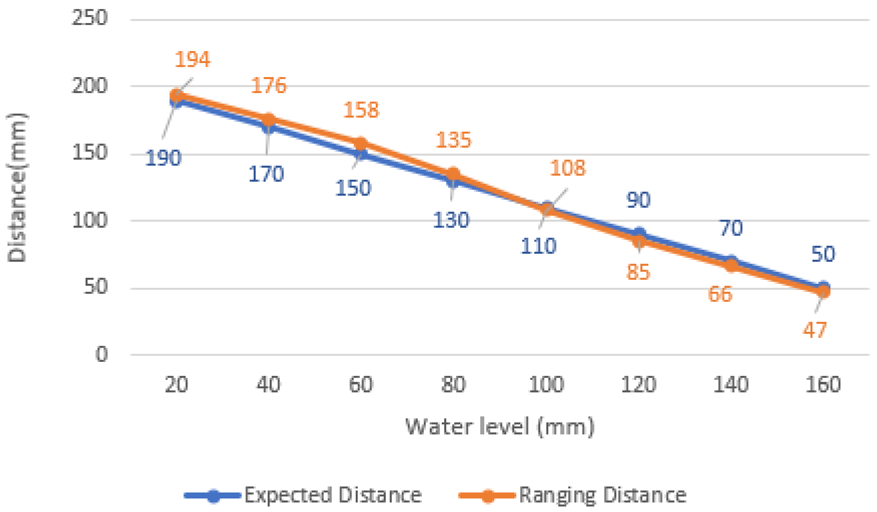
Figure 15. With offset calibration, without nonlinearity correction algorithm, and with cover glass



## 4.3 Test results with offset calibration, with nonlinearity correction algorithm, and with cover glass

The following figure shows that the measurements are quite linear between expected and actual measurement after adopting the nonlinearity correction algorithm and offset calibration. It may be further fine tuned depending on the specific container setup.

Figure 16. With offset calibration, with nonlinearity correction algorithm, and with cover glass



## 4.4 Non transparent liquid results

Tests on different types of non transparent liquids such as milk and syrup show much better results than transparent liquid. It is recommended to adopt the nonlinearity algorithm for other transparent liquids like oil, gasoline, soda to get high accuracy in ranging measurement.

# 5 Description of the experiment context

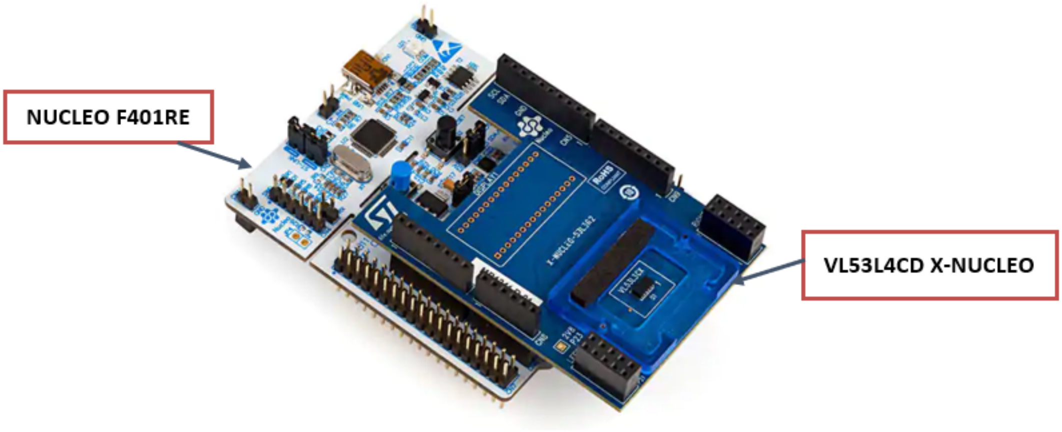
## 5.1 Hardware required

The VL53L4CD standard package is used as a proof of concept.

It can be found on st.com and includes:

- X-NUCLEO-53L4A1 expansion board for NUCLEO boards
- P-NUCLEO-53L4A1 expansion pack including the X-NUCLEO-53L4A1 and the NUCLEO-F401RE

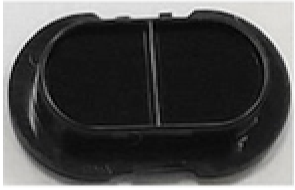
Figure 17. **VL53L4CD expansion board with NUCLEO F401RE board**



## 5.2 Cover glass

ST recommends using a light blocker cover glass to get better results compared to one window standard cover glass. Furthermore, the light blocker cover glass helps to maintain good performance results in a humid environment.

Figure 18. **Light blocker cover glass by Hornix**



## 5.3 Software requirements

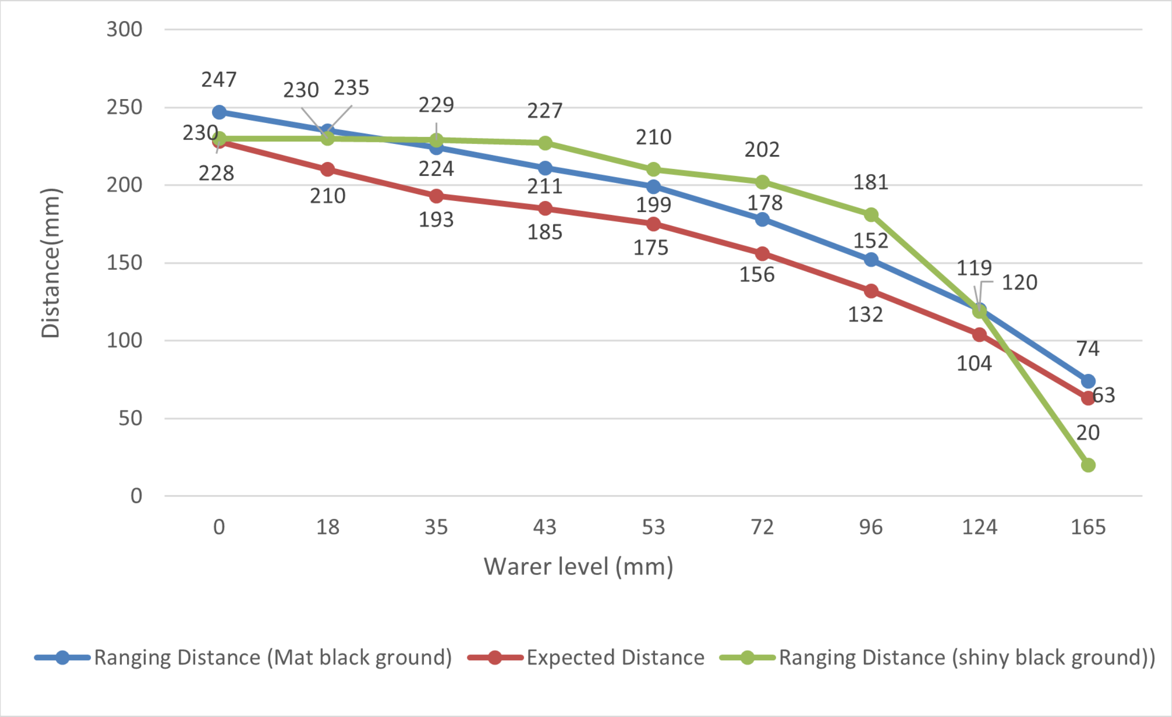
The GUI, ST characterization tool, and application example code described in this application note are available on st.com under the reference STSW-IMG039_L4CD. ST recommends using the GUI for evaluation.

# 6 Known limitations

The following known limitations have been identified and should be considered for final setup:

- Ranging is not reliable if the distance between the sensor and the water level is less than 20 mm.
- A highly reflective bottom impacts the accuracy for low water levels.
  The lower reflectivity base has better linearity in ranging than high reflectivity. It is recommended to use a low reflective base, like black mat reflector when selecting a container. Figure 19. Water level measurement with different colored base and no compensation shows the effect of the base's reflectivity on the ranging measurement without compensation.
- Ranging results are not reliable if water or liquid is in motion (for example ripples on the surface), or if bubbles form on the surface.

Figure 19. **Water level measurement with different colored base and no compensation**



## Revision history

Table 2. Document revision history

| Date | Version | Changes |
|---|---|---|
| 27-Oct-2022 | 1 | Initial release |

## Contents