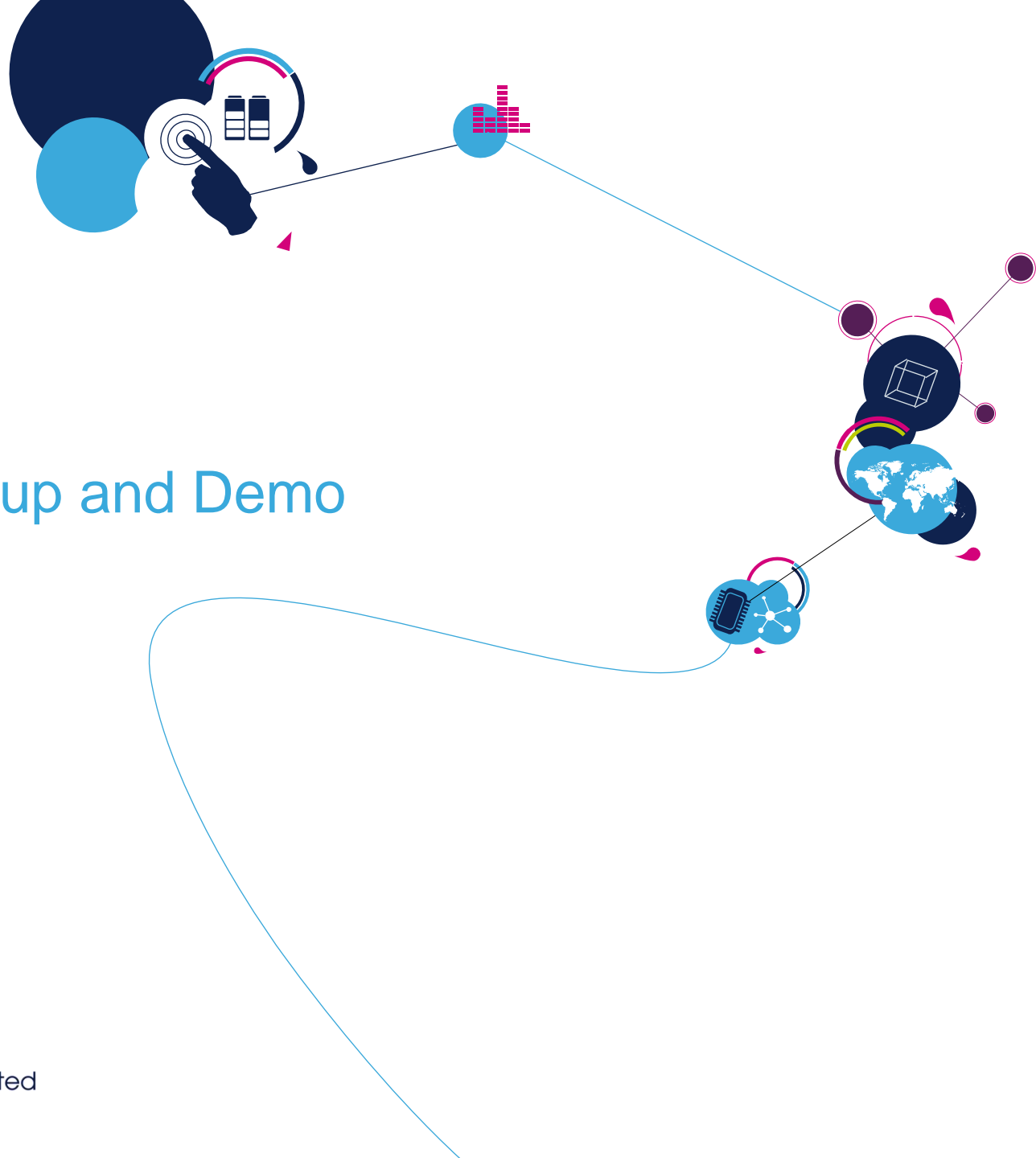


VL53L1X Setup and Demo

Apr 2018



- VL53L1X OverView
- VL53L1X Software package
- Presence HOD demo



VL53L1X OverView

Documentation



VL53L1X API Data Brief
VL53L1X API User Manual

VL53L1X Data Brief
VL53L1X Datasheet
VL53L0X Quick Start Guide
Applications Notes

VL53L1X GUI
User Manual



VL53L1X Eval GUI - STSW-IMG008

Get ranging live curves on your PC
Change key settings of the device
Data logging capabilities

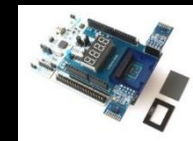
VL53L1X X-CUBE
User Manual



X-CUBE-53L1A1 package

Full integration in STM32 MCU (real-time)
All source code provided
Full access to product settings
Data logging capabilities
Ranging and Gesture detection demo

X-NUCLEO-53L1A1 HW
User Manual



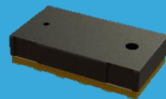
Hardware

P-NUCLEO-53L1A1
X-NUCLEO-53L1A1



VL53L1X C API package – STSW-IMG007

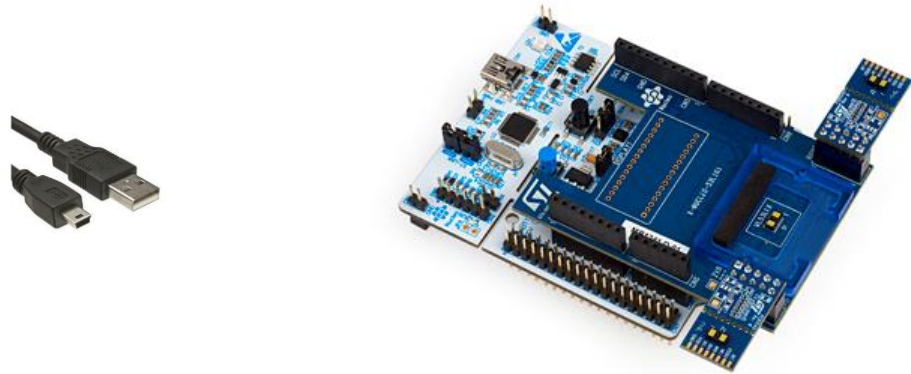
Discover API Source code



VL53L1X : Miniature ToF Ranging

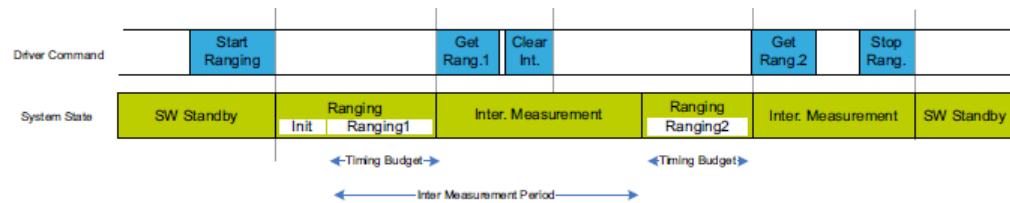
- VL53L1X Evaluation tools are all based on the same hardware pack composed of
 - Nucleo F401RE board
 - X-NUCLEO-53L1A1 Nucleo Expansion board
 - Optional two VL53L1X satellites
 - Several gap spacers and cover glass

Search for **P-NUCLEO-53L1A1** on st.com to order the pack and get documentation



- Advanced user detection
 - Ultra-low-power
 - Laptop, PC, tablets & IoT, portable handsets.
- Drones and Robotic
 - Take-off and landing phase
 - Obstacle avoidance
 - Ceiling detection
- Smart building and smart lighting
 - People detection, gesture control
- Sanitary
 - Robust user detection
- Smart shelves and vending machines
 - Goods inventory monitoring

- Autonomous mode is using sigma delta mechanism to measure distance:
 - Averages all targets in the full FoV
 - Allows multiple ROI
 - This mode is called Timed measurement, means ranging continuously and autonomously with a programmable inter-measurement period.



Note: Inter measurement period is start ranging1 to ranging2. Inter measurement timing is in b/w one ranging result to another.

Autonomous functionalities

8

Timing budget

- Autonomous ranging with selectable timing budget from 20ms to 1000ms.
- In short distance, minimum timing budget is 20ms.
- Ideally, for all distance modes, minimum timing budget is 33ms.
- If the inter-measurement timing is shorter than the timing budget, once the device completes the ranging, the next ranging starts immediately.

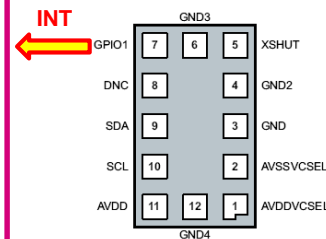
Key figures

- ROI size is possible (4x4 to 16x16 spad array)

Ranging characteristics (tbc)

- VL53L1X running in low power autonomous mode, no data sharing through I2C, just GPIO level change if human / object is detected.
- 4 detection thresholds criteria :
 - Above High
 - Below Low
 - In Window (\leq High AND \geq Low)
 - Out of Window ($>$ High OR $<$ Low)
 - No target (Interrupt is generated when no target is present.)

Key figures



- No I2C data output (low power mode)
- GPIO1 level changes (INT) to wake-up the host
- Host can then make VL53L1 switch to active mode for full ranging capabilities

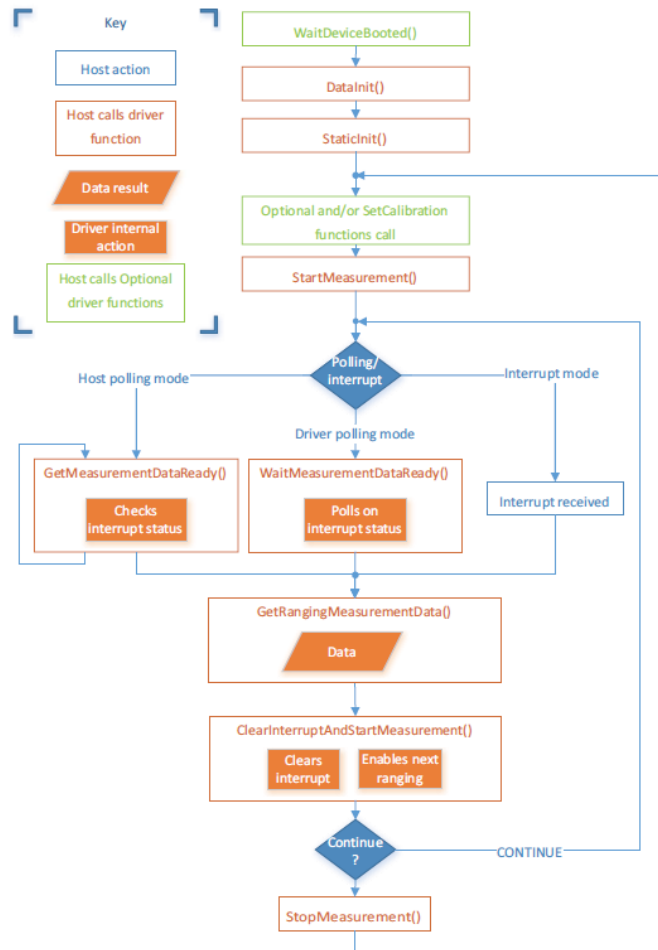
- Timing Budget and Inter measurement budget statically configured as part of Initialized process.
- It allows user to configure timing budget and Inter measurement budget by calling sub sequent API in Application level.
 - For Example : In Init time, let say default timing budget set to 41ms and inter measurement timing budget set to 500ms.
 - If user wants to update with new value then call it by following API subsequently.
 - VL53L1_SetMeasurementTimingBudgetMicroSeconds(Dev, 50000); // 50ms
 - VL53L1_SetInterMeasurementPeriodMilliseconds(Dev, 1000); //1sec



Software Package

- On www.st.com
 - X-CUBE-53L1A1 : [Link](#)
 - VL53L1X GUI([STSW-IMG008](#)) : [Link](#)
 - VL53L1X API([STSW-IMG007](#)) : [Link](#)
- Data sheet , User manual and GUI UM can get at st.com.

Software Flow



`VL53L1_WaitDeviceBooted()`: It ensures that the device is booted and ready.

`DataInit()`: It performs device initialization. Call once only.

`StaticInit()`: It loads device settings.

`StartMeasurement()`: It must be called to start a measurement

`WaitMeasurementDataReady()`: It polls on the device interrupt status until ranging data are ready.

`GetRangeMeasurementDataReady()`: To know if new ranging data is ready.

`GetRangeMeasurementData()`: To get ranging data.

`ClearInterruptAndStartMeasurement()`: Interrupt, is cleared. To get consistent results, it is mandatory to call this function after getting the ranging measurement.

VL53L1X Calibration

- To gain full performance of the device, calibration functions should run once at the customer production line.

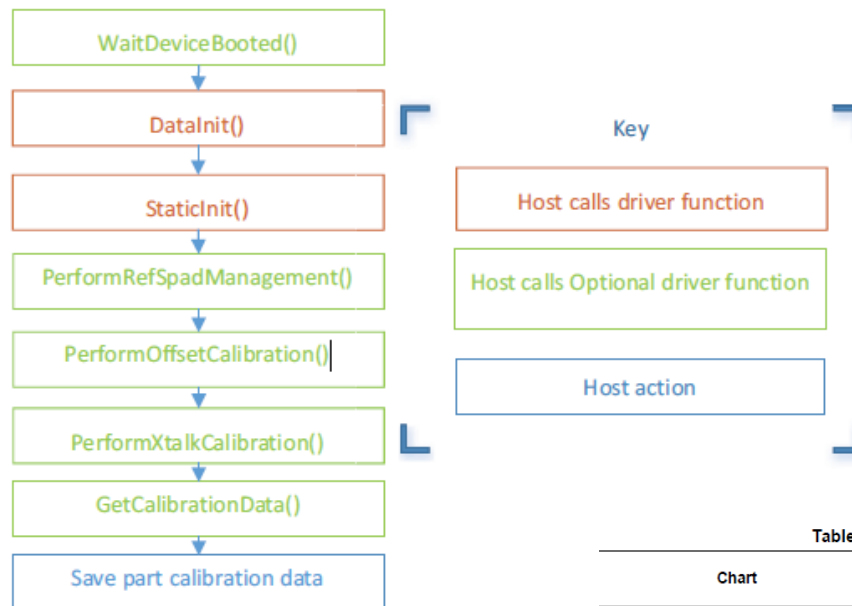


Table 5. Offset calibration set up

Chart	Chart distance (<i>CalDistanceMilliMeter</i>)	Ambient conditions
Gray target (17 % reflectance at 940 nm)	Recommended value: 140 mm	Dark (no IR contribution)

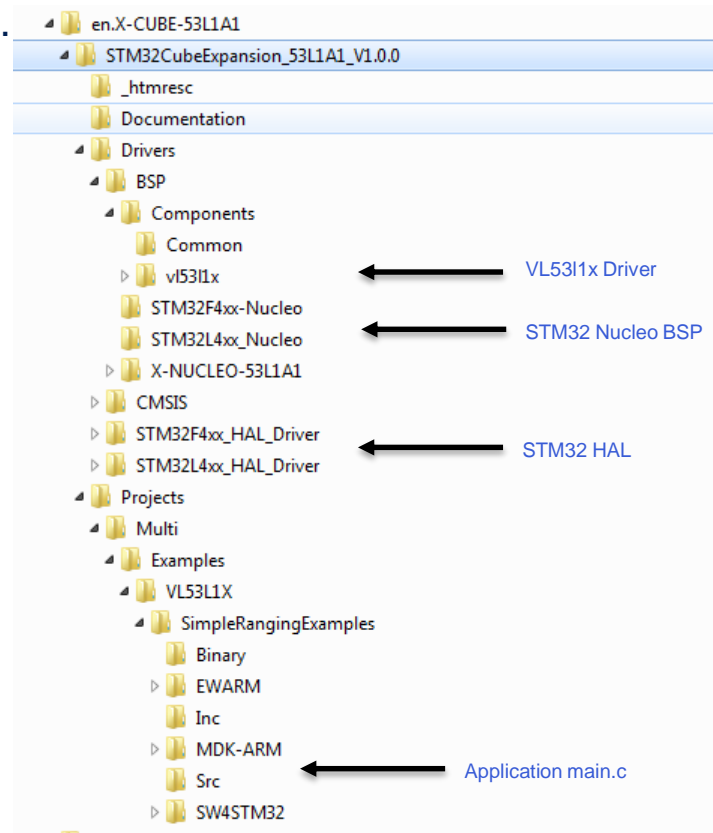
Table 6. Crosstalk calibration set up

Chart	Chart distance (<i>XtalkCalDistance</i>)	Ambient conditions
Gray target (17 % reflectance at 940 nm)	As defined in Section 3.3.3: Crosstalk calibration distance characterization	Dark (no IR contribution)

Note: X-talk calibration distance chart, same as Ewok.

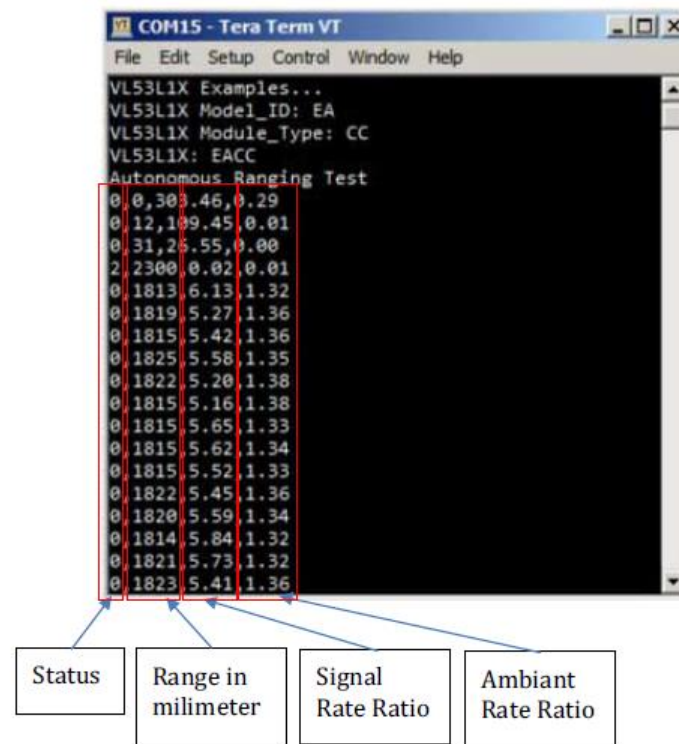
SDK

- We see software distribution of an application using the X-NUCLEO-53L1A1 expansion with either the NUCLEO-F401RE or NUCLEO-L476RG board.



Ranging Result

- Ranging result can be seen by serial communication, either using telnet or tera term.

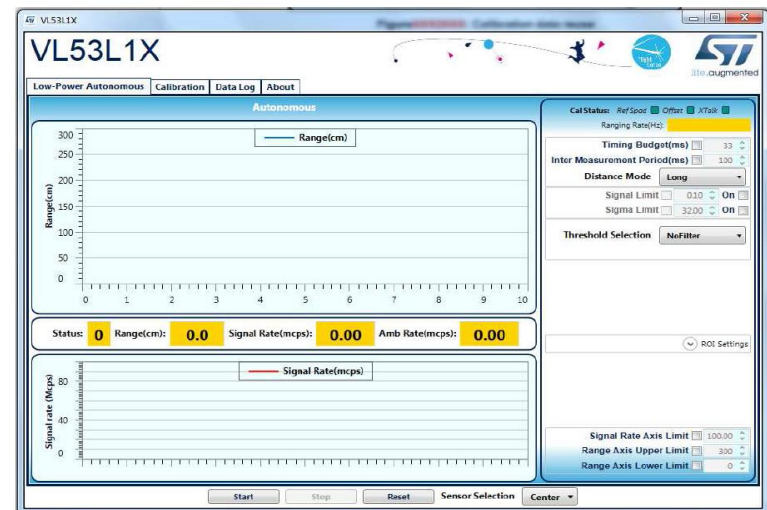


- Download from st.com searching for “STSW-IMG008”
 - Run the installer with Admin privileges
- PC Graphical User Interface to
 - Display (in live) key ranging data (distance, signal rate)
 - Change key parameters of VL53L1X
 - Perform calibration phases (offset and xTalk with cover glass)
 - Get data logging (.csv file)
- GUI is running on the PC connected to a P-NUCLEO-53L1A1 pack
 - Flash FW to Nucleo board
 - Start ranging and data will be transferred via USB to COM and display it in GUI
- It supports following OS.
 - Windows 7
 - Windows 8
 - Windows 10 (run as an admin must)



VL53L1X - GUI

- The “Low-power Autonomous” tab provides ranging distance and signal strength reflected from target.
- You can configure various parameters :
 - Distance mode
 - Threshold option
 - Timing budget
 - Inter measurement period
 - ROI selection
- Calibration Tab
 - It guides the steps to calibrate.
- Date logging
 - It captures range measurement data and save to a file during ranging.
- About
 - Get the versioning of GUI application and flash firmware.



Source code Files

- Drivers: Deal with only following 7 files.

```
vl53l1_api.o  
vl53l1_api_core.o  
vl53l1_api_preset_modes.o  
vl53l1_core.o  
vl53l1_core_support.o  
vl53l1_register_funcs.o  
vl53l1_wait.o
```

- Platform

```
vl53l1_platform.o
```

- Application

```
main.o
```

- X-Nucleo-53L1A1

```
X-NUCLEO-53L1A1.o
```

Analysis Application file

- The released SW package supports
 - Interrupt mode:** an Interrupt pin (GPIO1) sends an interrupt to the host when a new measurement is available.
 - Polling Mode:** Host has to check the status of ongoing measurement by polling API.
 - Just to flip this flag to switch mode in main.c

```
#define isInterrupt 0 /* If isInterrupt = 1 then device working in interrupt mode, else device working in polling mode */

void AutonomousLowPowerRangingTest(void)
{
    static VL53L1_RangingMeasurementData_t RangingData;
    printf("Autonomous Ranging Test\n");
    status = VL53L1_WaitDeviceBooted(Dev);
    status = VL53L1_DataInit(Dev);
    status = VL53L1_StaticInit(Dev);
    status = VL53L1_SetMeasurementTimingBudgetMicroSeconds(Dev, 50000);
    status = VL53L1_SetInterMeasurementPeriodMilliseconds(Dev, 500);
    status = VL53L1_StartMeasurement(Dev);

    if(status){
        printf("VL53L1_StartMeasurement failed \n");
        while(1);
    }
    if (isInterrupt){
        do // interrupt mode
        {
            _WFI();
            if(IntCount !=0 ){
                IntCount=0;
                status = VL53L1_GetRangingMeasurementData(Dev, &RangingData);
                if(status==0){
                    printf("%d,%d,%.2f,%.2f\n", RangingData.RangeStatus,RangingData.RangeMilliMeter,
                        RangingData.SignalRateRtnMegaCps/65536.0,RangingData.AmbientRateRtnMegaCps/65536.0);
                    status = VL53L1_ClearInterruptAndStartMeasurement(Dev);
                }
            }
        } while(1);
    }
    else{
        do // polling mode
        {
            status = VL53L1_WaitMeasurementDataReady(Dev);
            if(!status)
            {
                status = VL53L1_GetRangingMeasurementData(Dev, &RangingData);
                if(status==0){
                    printf("%d,%d,%.2f,%.2f\n", RangingData.RangeStatus,RangingData.RangeMilliMeter,
                        (RangingData.SignalRateRtnMegaCps/65536.0),RangingData.AmbientRateRtnMegaCps/65536.0);
                    status = VL53L1_ClearInterruptAndStartMeasurement(Dev);
                }
            }
        } while (1);
    }
}
```

Interrupt is coming from PIN4 which is configured as an Interrupt, get by this function

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)

```
{
    if ((GPIO_Pin==VL53L1X_INT_Pin)
    {
        IntCount++;
    }
}
```

← In polling, host must check ranging data is ready ?

Interrupt mode - Threshold

- Threshold can be set based on signal or distance or both parameters.
- Distance detection mode is governed by CrossMode value.
- 4 detection thresholds criteria :
 - Above High (CrossMode =1)
 - Below Low (CrossMode =0)
 - In Window (<=High AND >=Low) (CrossMode =3)
 - Out of Window (>High OR <Low) (CrossMode =2)
 - No Target (IntrNoTarget =1)

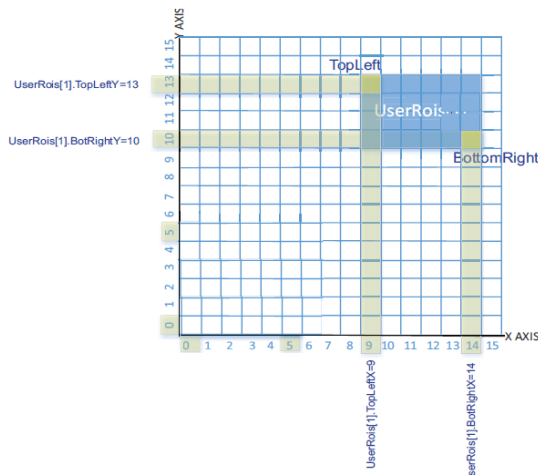
For example:

```
/* Autonomous ranging loop*/
void AutonomousLowPowerRangingTest(void)
{
    static VL53L1_RangingMeasurementData_t RangingData;
    VL53L1_DetectionConfig_t DetectionConfig;
    //char StrDisplay[5]="\0";
    printf("Autonomous Ranging Test\n");
    status = VL53L1_WaitDeviceBooted(Dev);
    status = VL53L1_DataInit(Dev);
    status = VL53L1_StaticInit(Dev);
    status = VL53L1_SetDistanceMode(Dev, VL53L1_DISTANCEMODE_LONG);
    status = VL53L1_SetMeasurementTimingBudgetMicroSeconds(Dev, 50000);
    status = VL53L1_SetInterMeasurementPeriodMilliseconds(Dev, 500);
    memset(&DetectionConfig, 0, sizeof(DetectionConfig));
    /*Example, target is detected only with in 10cm to 50cm*/
    DetectionConfig.DetectionMode = VL53L1_DETECTION_DISTANCE_ONLY;
    DetectionConfig.IntrNoTarget = 0;
    DetectionConfig.Distance.CrossMode = VL53L1_THRESHOLD_IN_WINDOW;
    DetectionConfig.Distance.Low = 100;
    DetectionConfig.Distance.High = 500;
    status = VL53L1_SetThresholdConfig(Dev, &DetectionConfig);
    status = VL53L1_StartMeasurement(Dev);

    if(status){
        printf("VL53L1_StartMeasurement failed \n");
        while(1);
    }
    if (!status){
```

VL53L1X - ROI

- The receiving SPAD array includes 16x16 SPADs which covers full FoV. It allows to narrow down FoV by programming smaller ROI. Minimum ROI size is 4x4.
- In order to set ROI, top left and bottom right coordinates are required.



Example to set one ROI:

```
VL53L1_UserRoi_t roiConfig;
roiConfig.TopLeftX = 9;
roiConfig.TopLeftY = 13;
roiConfig.BotRightX = 14;
roiConfig.BotRightY = 10;
status = VL53L1_SetUserROI(&VL53L1Dev,
roiConfig);
```

- Receiver field of View can be narrow down from 27° to 15°.

VL53L1X – ROI Result

- It allows to reduce FoV, but on other hand penalties on ranging distance. Thus, caution to use it.

Table 9. Typical performances in partial ROI in dark conditions

Parameter	Target reflectance	16x16	8x8	4x4
Max. distance (cm)	White 88 %	360	308	170
	Grey 54 %	340	254	143
	Grey 17%	170	119	45
Diagonal FoV (degrees)		27	20	15
Ranging error (mm)		± 20	± 20	± 20

- From the above table, if application consider 15 ° FoV, then sensor can range max ½ of the actual ranging capability.

VL53L1X – Power consumption

Parameter	Min.	Typ.	Max.	Unit
HW standby	3	5	7	uA
SW standby ⁽²⁾	4	6	9	
Inter measurement		20		
Ranging average (AVDD + AVDDVCSEL) ^{(3) (4)}		16	18	mA
Average power consumption at 10 Hz with 33 ms timing budget			20	mW
Average power consumption at 1 Hz with 20 ms timing budget when no target detected		0.9		
Average power consumption at 1 Hz with 20 ms timing budget when target detected		1.4		

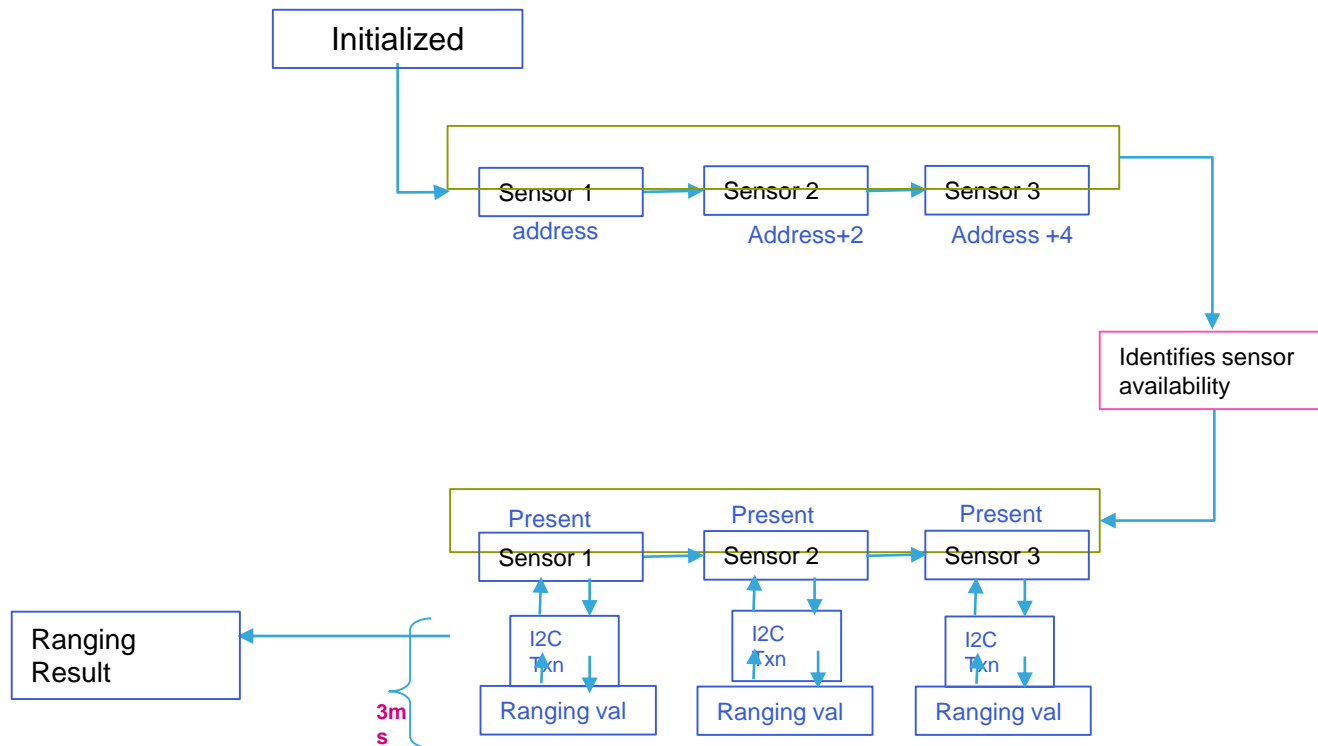
- VL53L1X supports THREE different IDES
 - AC6 (Free to use unlimited code size)
 - IAR (License limited, 1 year)
 - Keil-ARM (Limited Code size , 32Kb) , ST agreement License version supports STM32M0XXX series.
- MCU
 - STM32F432XX
 - STM32L476XX
- Among 3 IDES, IAR is fast to debug and easy to calculate source code size.

VL53L1X's driver Code analysis on IDE

25

- After analyzing VL53L1X driver code (Without Calibration), the ROM size is ~9Kb and RAM is 280 bytes. Detail analysis and map file can be found on attached email , date : March 15, 2018 5:00 PM, subject: VL53L1X Multi sensors.
- When analyzing multisensory code, main.c impacts a lot rather than driver code.
 - ROM = 40% increase. (For ex: 1x= 1K, 2x=1.4K...)
 - RAM = 100% increase (For ex: 1x= 1K, 2x=2K...)

- Initialized each sensor's I2C address by incrementing 2.



- Initialized each sensor's I2C address by incrementing 2.

```
for (i = 0; i < MAX_DEV_USE; i++) {
    VL53L1_Dev_t *pDev;
    pDev = &VL53L1Devs[i];
    pDev->I2cDevAddr = PROXIMITY_I2C_ADDRESS;
    pDev->Present = 0;
    // VL53L1Devs[i].Ready=0;
    Status = XNUCLE053L1A1_ResetId(i, 0);
    HAL_Delay(3);
    Status = XNUCLE053L1A1_ResetId( pDev->DevId, 1);
    HAL_Delay(3);
    FinalAddress=PROXIMITY_I2C_ADDRESS+(i+1)*2;| ← Program address

    do {

        /* Try to read one register using default 0x52 address */
        Status = VL53L1_RdWord(pDev, VL53L1_IDENTIFICATION_MODEL_ID, &DevId);
        if (Status) {
            printf("#%d - %02X Read id fail\n", i, DevId);
            break;
        }
        if (DevId == VL53L1_MODEL_ID) {
            /* Sensor is found => Change its I2C address to final one */
            Status = VL53L1_SetDeviceAddress(pDev,FinalAddress); ← Set address to sensor
            if (Status != 0) {
                printf("#d VL53L1X_SetDeviceAddress fail\n", i);
                break;
            }
            pDev->I2cDevAddr = FinalAddress;
            /* Check all is OK with the new I2C address and initialize the sensor */
            Status = VL53L1_RdWord(pDev, VL53L1_IDENTIFICATION_MODEL_ID, &DevId);
            printf(" ID #%d - DeVID %02X \n", i, DevId);
            if( Status == 0 ){
                pDev->Present = 1;
                nSensorPresent++;
            }
            else{
                printf("VL53L1_DataInit %d fail\n", i);
                break;
            }
        }
        } « end if DevId==VL53L1_MODEL_ID »
        else {
            printf("#%d unknown ID %x\n", i, DevId);
            Status = 1;
        }
    } « end do » while (0);
}
```

- Check present sensor and ranging.

```
VL53L1_Error MultiSensorRanging(){
    int Status = VL53L1_ERROR_NONE;
    int i,nReady=0, StateChange =0;
    /* Start ranging until blue button is pressed */
    do {

        /* wait for all enabled devices to have a measure */
        nReady=0;
        /* Start ranging until blue button is pressed */
        do {
            HAL_Delay(1);

            for( i=0; i<MAX_DEV_USE; i++)
            {
                /* Skip devices not present or not enabled */
                if( ! VL53L1Devs[i].Present || ! VL53L1Devs[i].Enabled ) ← Check present
                {
                    continue;
                    Device

                }

                if (isInterrupt){
                    __WFI();

                    if(IntCount !=0 ){
                        IntCount=0;
                        status = VL53L1_GetRangingMeasurementData(&VL53L1Devs[i], &RangingData);
                        if(status==0){
                            printf(" Interrupt Dev ID= [%d], Status= [%d ] Data= [ %d ] mm\n", VL53L1Devs[i].DevId,Rangir
                        )
                        status = VL53L1_ClearInterruptAndStartMeasurement(&VL53L1Devs[i]);
                        nReady++;
                    }
                }
            }
        }
        else
        {
            printf("*****\n");
            status = VL53L1_WaitMeasurementDataReady(&VL53L1Devs[i]);
            if(!status)
            {
                status = VL53L1_GetRangingMeasurementData(&VL53L1Devs[i],&RangingData);
                if(status==0){
                    printf(" Dev ID= [%d], Status= [%d ] Data= [ %d ] mm\n", VL53L1Devs[i].DevId,RangingData.RangeSta
                )
                status = VL53L1_ClearInterruptAndStartMeasurement(&VL53L1Devs[i]);
                nReady++;
            }
        }
    }
}
```

29

- Figure 5. Interrupt configurations**



VL53L1X benefits versus VL53L0X

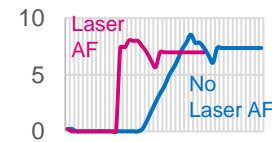
30

2x longer distance detection



2x faster ranging operation

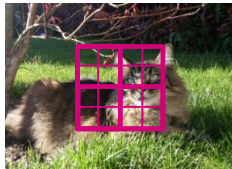
Fastest miniature ToF product on the market



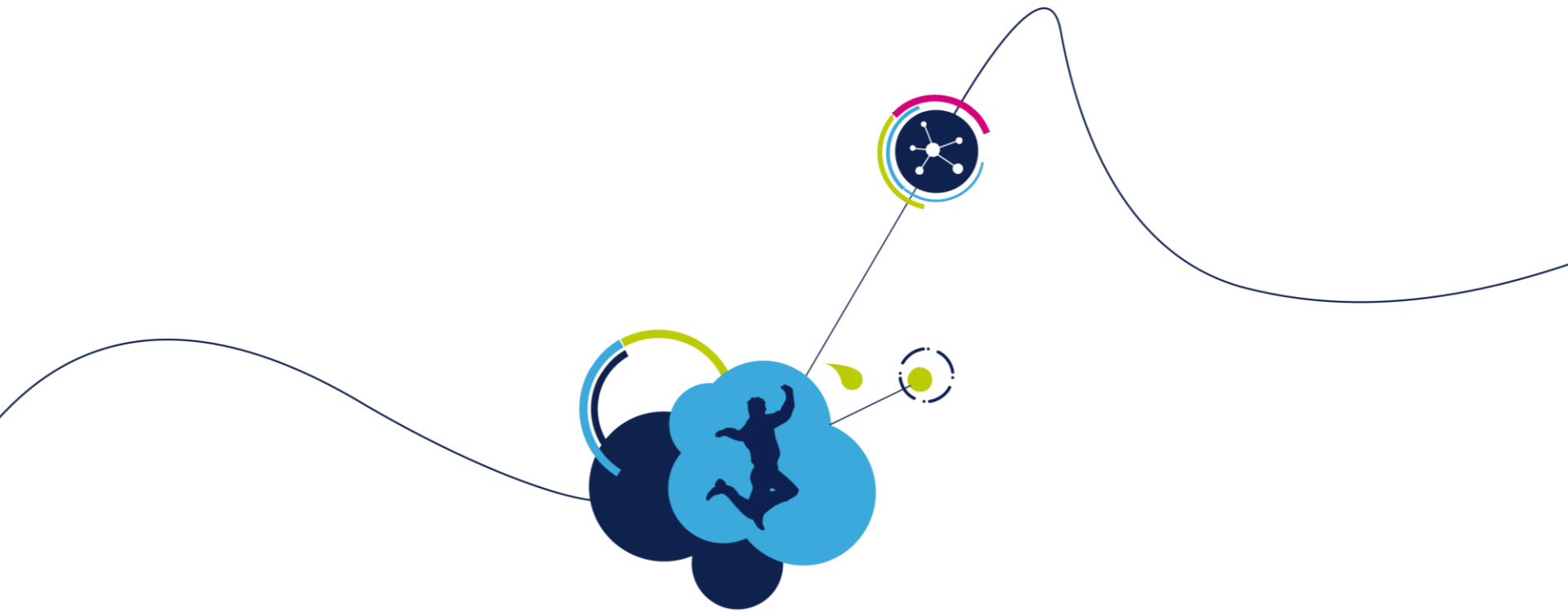
- 20ms typical ranging timing budget, could further down. Faster “jump to focus”
- Run 50Hz frequency

FoV Control

Customizable array



- Sensing array can be configured from 16x16 to 4x4.
- FoV can be reduced from 27 ° to 15 °.



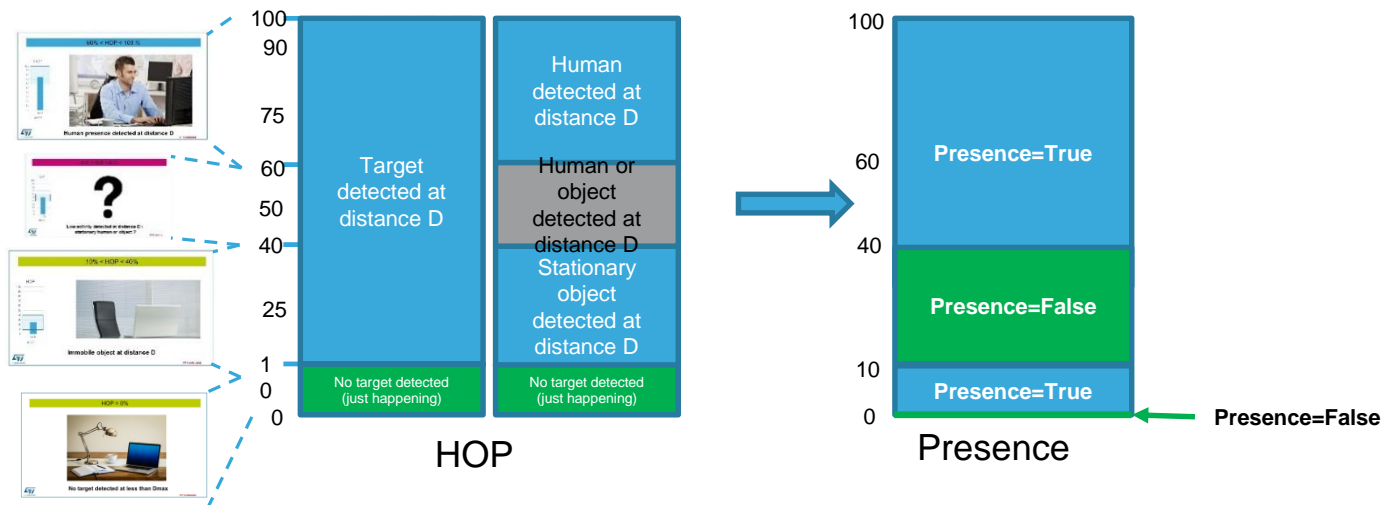
Presence HOD Demo



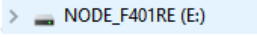
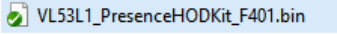
• HOD Demo outputs

- HOP : Human Occupancy Probability (integer value between 0 and 100)
- Human Presence : True/False (based on HOP)
- D : Distance of the detected target (object or human)

• How to interpret HOD outputs ?





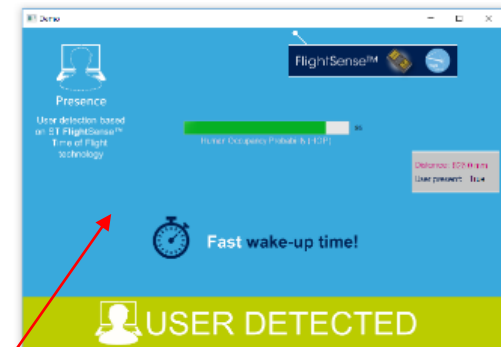
- Unzip the PresenceHODKit FW package on your PC
- Connect hardware to the PC through USB with Nucleo board (please install STLink007 and STLink009 firstly if you are using Win7)
- Check hardware is recognized as a mass storage
 - Nucleo 
- Drag & Drop pre-compiled binary to mass storage:
 - This will flash hardware with the program 



- Test different scenarios

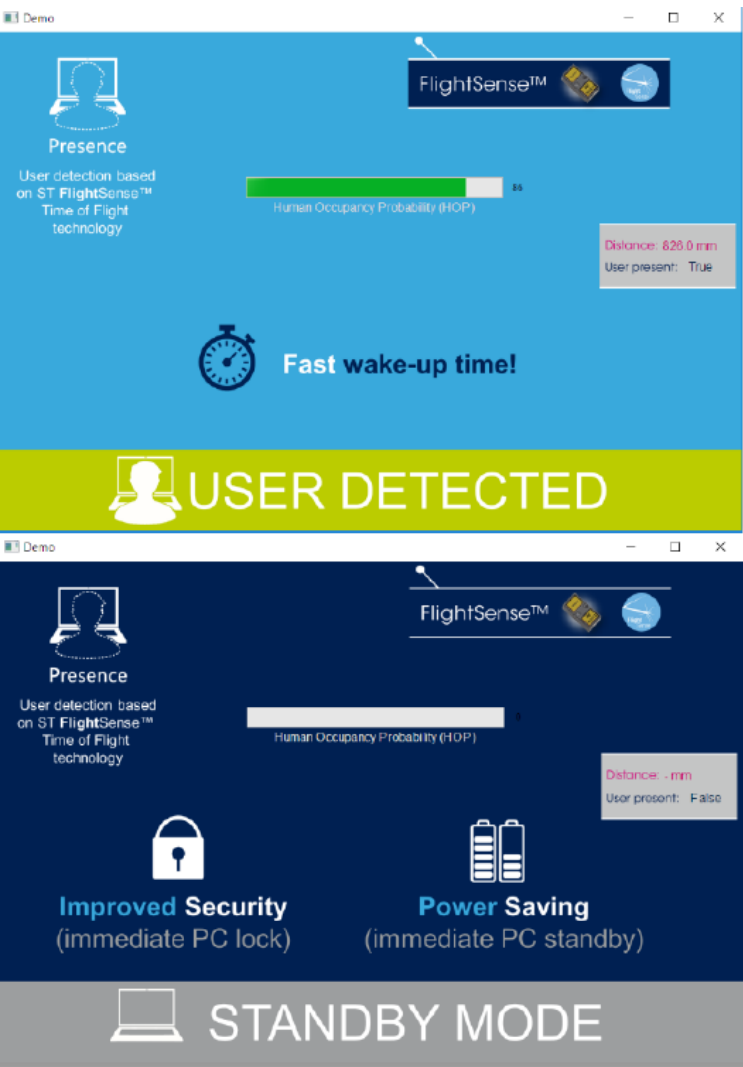
- Stay in front of the sensor
 - Presence is 1, HOP always > 40%
- Leave the sensor FoV
 - After 3 sec (HOD parameter), Presence flag goes to 0, HOP at 0% and no data reported anymore
- Come back in the sensor FoV
 - Presence is 1, HOP at 100%
- Stop moving
 - Presence remains at 1 with HOP decreasing slowly and converging at 75% (micro-variation detected)
- Move a bit
 - Presence still at 1 with HOP jumping to 100%
- Leave the sensor FoV and let a chair (detected by the sensor)
 - After ~20 sec, Presence flag goes to 0 with HOP converging at 25%

- Unzip HODDemo1.0.0.zip and click HODDemo.exe to start ST HODDemo graphical tool see the output of HOP(also distance)



Tool purpose

Demo window



The demo window of the *HOD Viewer* shows the HOP value and the suggested computer behaviour (power-on or standby).

ST Confidential

Q & A