VL53L8CX Linux Driver

Getting started with Linux Driver and Raspberry PI 3

Introduction

The VL53L8CX is a state of the art, Time-of-Flight (ToF), laser-ranging sensor enhancing the ST FlightSense product family. Housed in a miniature reflowable package, it integrates a SPAD array, physical infrared filters, and diffractive optics (DOE) to achieve the best ranging performance in various ambient lighting conditions with a range of cover glass materials.

The purpose of this user manual is to describe how to start using the Linux Driver based on VL53L8CX Ultra Lite Driver (ULD).

Figure 1: VL53L8CX sensor module

References:

1) VL53L8CX Datasheet (DS14161)
2) VL53L8CX User Manual (UM3109)

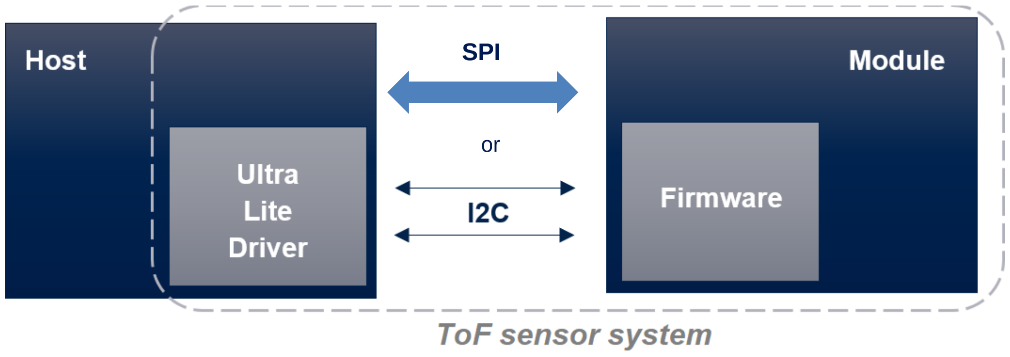
Contents

1 Functional description

1.1 System overview

The VL53L8CX system is composed of a hardware module and the Ultra Lite Driver software (VL53L8CX ULD) running on a host (see figure below). The hardware module contains the ToF sensor. ST delivers the software driver which is referred to in this document as "the driver". This document describes the functions of the driver which are accessible to the host. These functions control the sensor and get the ranging data.

Figure 2 : VL53L8CX system overvie

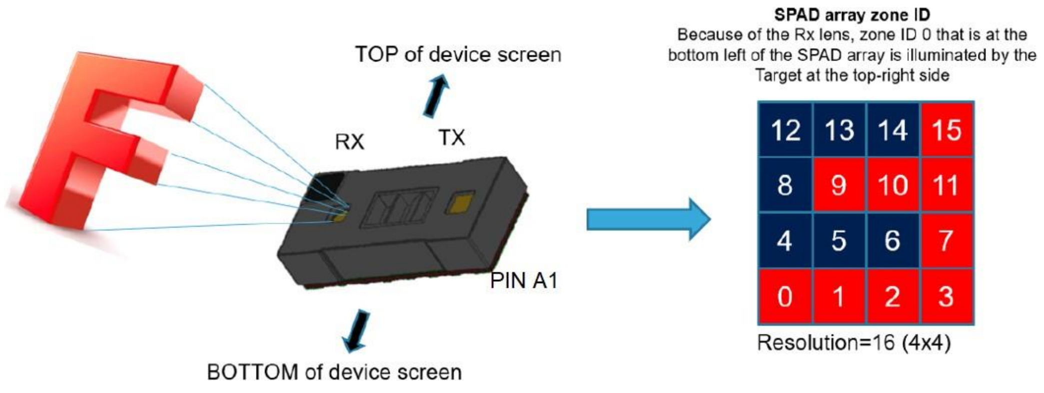


1.2 Effective orientation

The module includes a lens over the RX aperture which flips (horizontally and vertically) the captured image of the target. As a consequence, the zone identified as zone 0 in the bottom left of the SPAD array is illuminated by a target located at the top right-hand side of the scene.

For an application, when the user is looking at the screen of the device, it is recommended to orientate the VL53L8CX with the RX aperture to the left and the TX aperture to the right as shown in *Figure 3*.

Figure 3 : VL53L8CX effective orientation

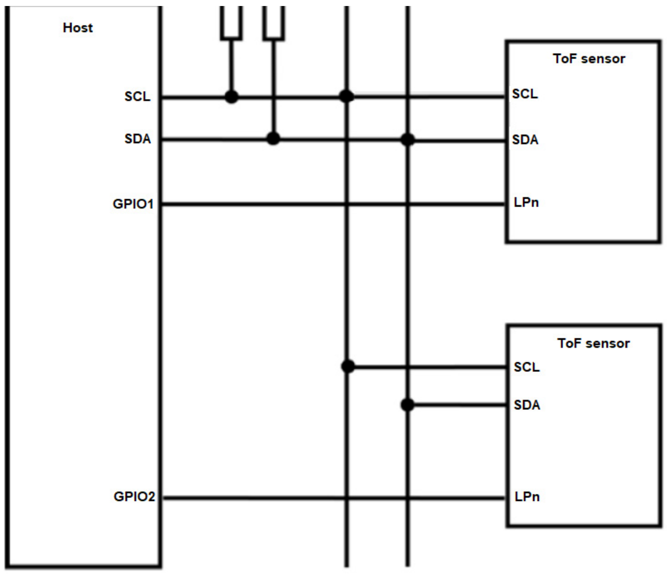

TOP of device screen
RX
TX
PIN A1
BOTTOM of device screen
SPAD array zone ID
Because of the Rx lens, zone ID 0 that is at the bottom left of the SPAD array is illuminated by the Target at the top-right side
12 13 14 15
8 9 10 11
4 5 6 7
0 1 2 3
Resolution=16 (4x4)

1.3 Setting I2C address

The communication between the driver and the firmware is handled by I2C or SPI, with a capability of operating up to 1MHz for I2C and 3MHz for SPI. The implementation requires pull-ups on the different lines. Please see VL53L8CX datasheet for more information.

The VL53L8CX device has a default I2C address of 0x52. However, it is possible to change the default address to avoid conflicts with other devices, or facilitate adding multiple VL53L8CX modules to the system for a greater system FoV. The I2C address can be changed using vl53l8cx_set_i2c_address() function.

Figure 4: Multiple sensors on I2C bus



To allow a device to have its I2C address changed without affecting others on the I2C bus, it is important to disable the I2C communication of the devices not being changed. The procedure is the following one:

1) Power up the system as normal.
2) Pull down the LPn pin of the device that will not be having its address changed.
3) Pull up the LPn pin of the device that has the I2C address changed.
4) Program the I2C address to the device using function *vl53l8cx_set_i2c_address()*.
5) Pull up the LPn pin of the device not being reprogrammed.

All devices should now be available on the I2C bus. Repeat the above steps for as many VL53L8CX devices in the system that require a new I2C address

2 Package content and data flow

2.1 Driver architecture and content

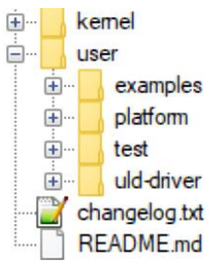
The VL53L8CX Linux driver is based on the VL53L8CX Ultra Lite Driver (ULD). All driver ULD features are described into the User Manual.

The Linux driver is a package composed of several folders:

- kernel / user folder: contain the two supported modes of the driver
- examples: contain several examples to use the sensor
- platform: contain mandatory macros and platform functions. It needs to be filled by the customer.
- test: contain the main input file
- uld-driver: contain the VL53L8CX Ultra Lite Driver (ULD)

The described driver architecture is represented on *Figure 5*.

Figure 5: Linux driver architecture



```
kernel
user
    examples
    platform
    test
    uld-driver
changelog.txt
README.md
```

Note: Platform.h file contains mandatory macros to use ULD. All the file content is mandatory to correctly use Ultra Lite Driver

2.2 Supported system modes

The VL53L8CX Linux driver support the User space mode and Full Kernel mode. The mode can be selected using compilation flag STMVL53L8CX_KERNEL into the test Makefile. By default the driver is set in User space mode.

3 Running I2C example on Raspberry PI 3

The proposed Linux implementation is customized to run on a Raspberry PI 3, but can be adapted to run on any Linux embedded application, as far as the VL53L8CX sensor is connected through I2C.

3.1 Hardware connection

The most simple to run an example with Linux driver is to use a VL53L8CX SATEL. Pins need to be connected following the schematics on figures below.

Figure 6: VL53L8CX SATEL to Raspberry PI 3 - I2C schematics

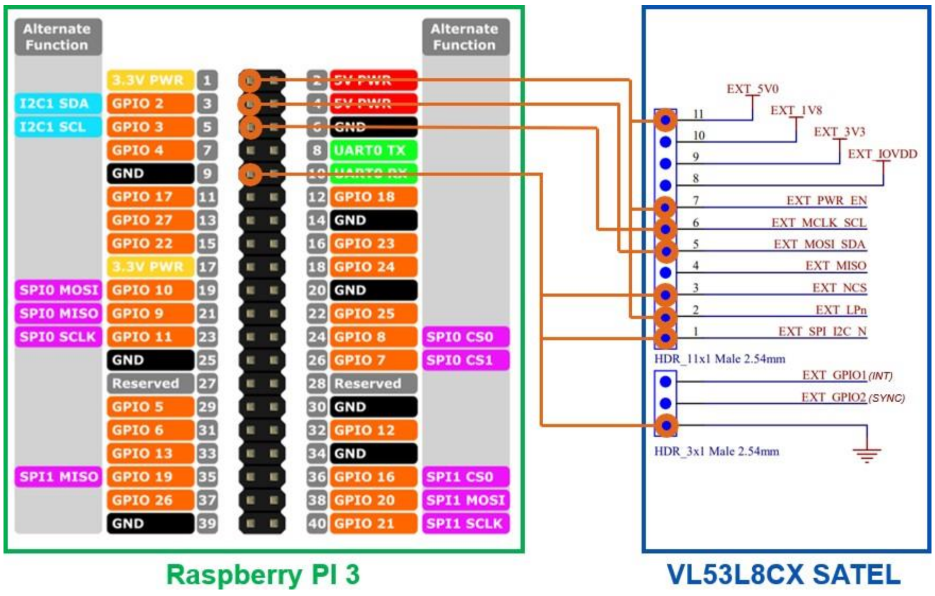
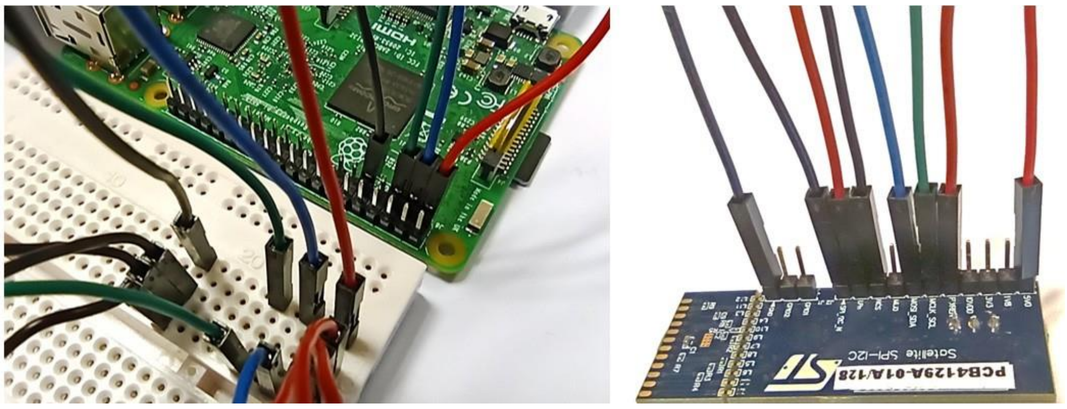


Figure 7:VL53L8CX SATEL to Raspberry PI 3 - photos



3.2 User space compilation

User space mode only requires to compile the makefile located into the ./test folder. The following sequence is required:

- Go to test folder
  - cd VL53L8CX_ULD_Linux/user/test
- Build program
  - make

Then user can run application as described in section 3.4.

3.3 Kernel compilation

Kernel compilation requires installation if kernel source headers and device tree blog to be run. The following sequence is required:

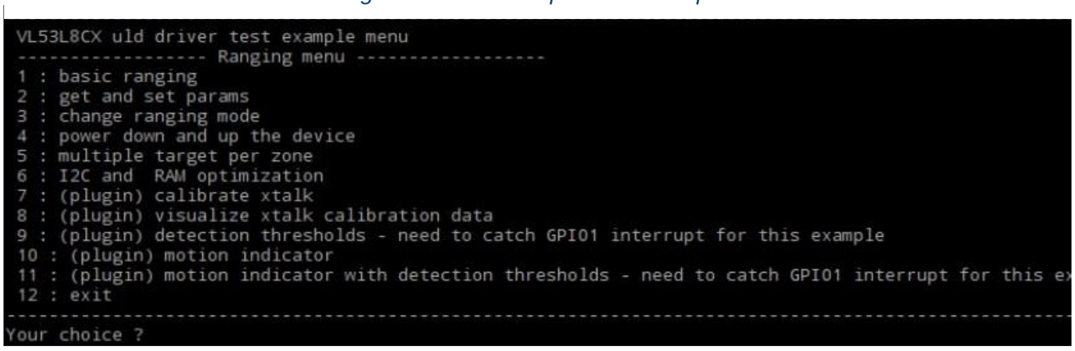
- Install the Raspberry PI Kernel source headers (refer to official documentation to download a version matching with kernel version)
- Update the boot configuration, adding or un-commenting the following lines of file /boot/config.txt
  - dtparam=i2c_arm=on
  - dtparam=i2c1=on
  - dtparam=i2c1_baudrate=1000000
  - dtoverlay=stmvl53l8cx
- Compile the device tree blob
  - cd VL53L8CX_ULD_Linux /kernel
  - make dtb
  - sudo reboot
- Go to test folder and enable kernel cflag into the test Makefile
  - cd VL53L8CX_ULD_Linux/user/test
  - nano VL53L8CX_ULD_Linux/user/test/Makefile
  - CFLAGS_RELEASE += STMVL53L8CX_KERNEL
- Build program
  - make
- Compile the kernel module
  - cd VL53L8CX_ULD_Linux/kernel
  - make clean
  - make
  - sudo make insert
- Then user can run application as described in section 3.4.

3.4 Run a test application

Once the compilation is done, user can go to the test folder and run the example menu. The menu allows using examples located in the ./examples folder.

- Go to test folder and run the test code.
  - cd VL53L8CX_ULD_Linux/user/test
  - ./menu

Figure 6: Screen capture of example menu



```
VL53L8CX uld driver test example menu
----------------- Ranging menu -----------------
1 : basic ranging
2 : get and set params
3 : change ranging mode
4 : power down and up the device
5 : multiple target per zone
6 : I2C and  RAM optimization
7 : (plugin) calibrate xtalk
8 : (plugin) visualize xtalk calibration data
9 : (plugin) detection thresholds - need to catch GPIO1 interrupt for this example
10 : (plugin) motion indicator
11 : (plugin) motion indicator with detection thresholds - need to catch GPIO1 interrupt for this ex
12 : exit
----------------------------------------------------------------------------------------------------
Your choice ?
```

Note: Examples 9 and 11 require to catch the interrupt raised on INT pin. User needs to add wires and to implement GPIO callback for using such examples.

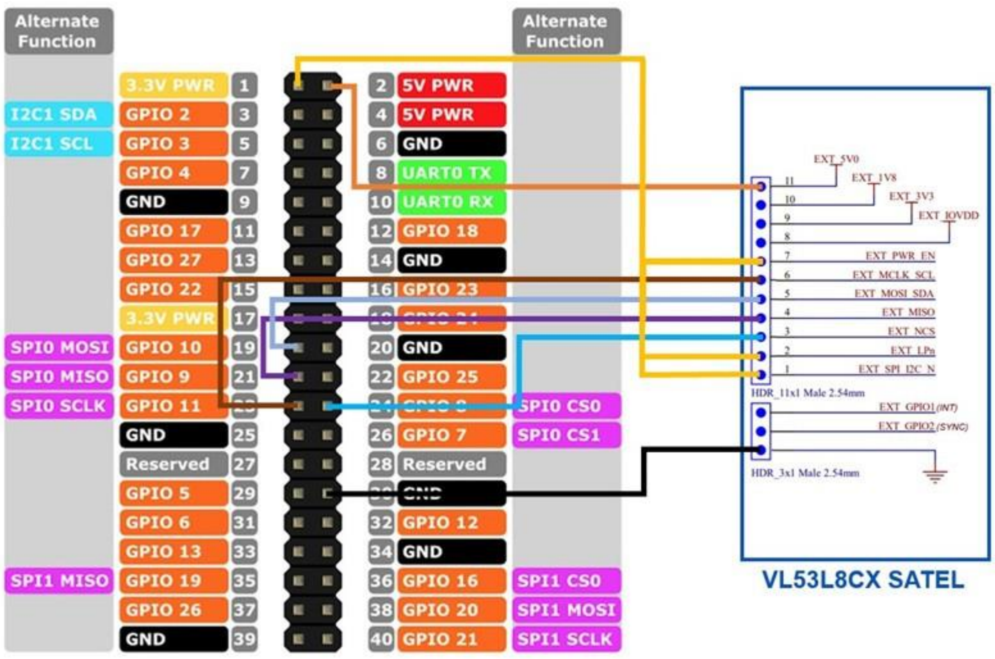
4 Running SPI example on Raspberry PI 3

The proposed Linux implementation is customized to run on a Raspberry PI 3, but can be adapted to run on any Linux embedded application, as far as the VL53L8CX sensor is connected through SPI. The example given by STMicroelectronics has been tested using 2 sensors.

4.1 Hardware connection

The most simple to run an example with Linux driver is to use a VL53L8CX SATEL. Pins need to be connected following the schematic below.

Figure 7:VL53L8CX SATEL to Raspberry PI 3 - SPI schematics

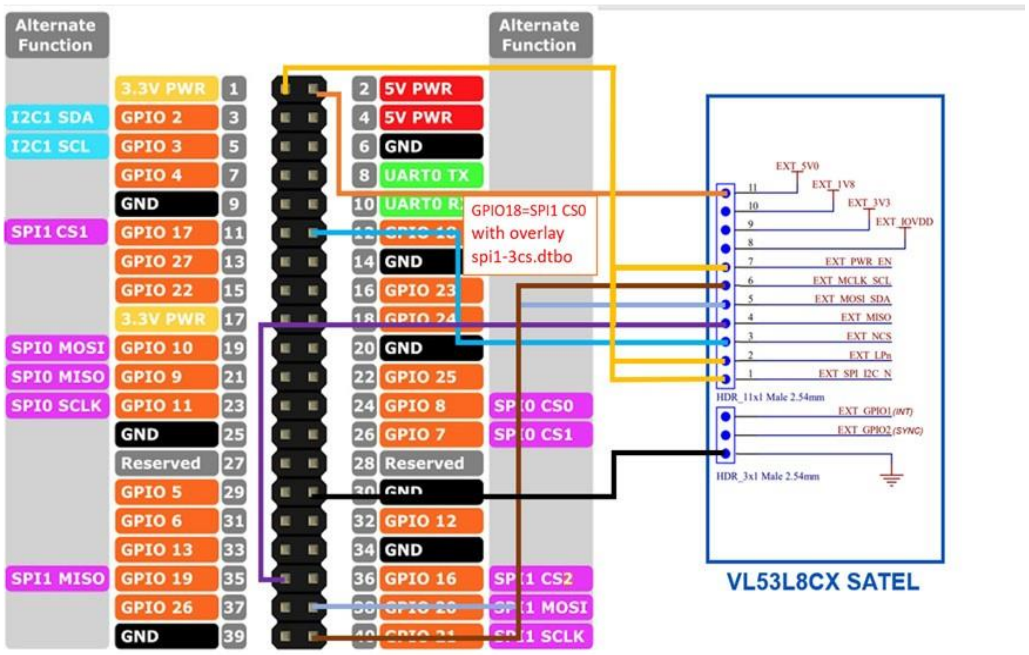


It is assumed that the satellite is populated with level shifters and regulators to generate 3v3, IOVDD & 1v8 from the 5v supplied on the EXT_5V0 pin. If the satellite has no regulators soldered then some external regulator shall be added and connected to this setup to supply them.

A second satellite can be connected to SPI0 plugged on the same wires as satellite 1 but with its NCS pin shall be connected to GPIO7 (SPI0 CS1).

If needed, a third satellite can be added if the SPI1 is enabled with spi1-cs3.dtbo device tree overlay. And a fourth satellite can be connected to SPI1 plugged on the same wires as this satellite 3 but with its NCS pin shall be connected to GPIO17 (SPI1 CS1).

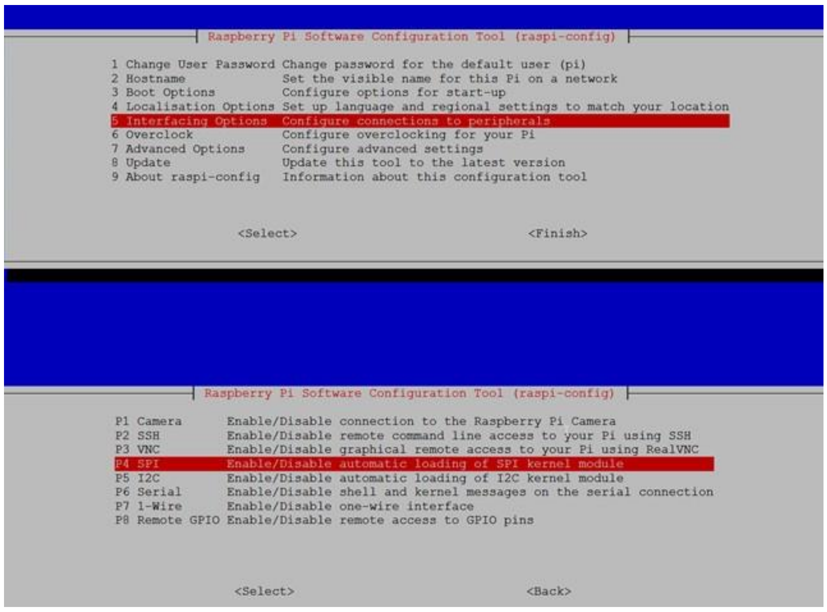
Figure 8:VL53L8CX SATEL to Raspberry PI 3 - SPI schematics (multi) with overlay sp1-3cs



4.2 User space compilation

Run 'sudo raspi-config' in order to enable SPI interface in the linux kernel.

Figure 9: SPI RPI configuration

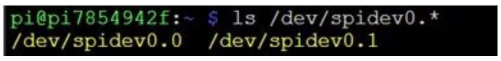


```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the default user (pi)
2 Hostname             Set the visible name for this Pi on a network
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

               <Select>                          <Finish>
```

```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

               <Select>                          <Back>
```

Then reboot the Raspberry PI. Afterwards check if /dev/spidev0.0 exist.

Figure 10: SPI - checking if spidev0 exists



```
pi@pi7854942f:~ $ ls /dev/spidev0.*
/dev/spidev0.0  /dev/spidev0.1
```

User space mode only requires to compile the makefile located into the ./test folder. The following sequence is required:

- Go to test folder
  - cd VL53L8CX_ULD_Linux/user/test
- Enable the SPI cflags option in the makefile
  - nano VL53L8CX_ULD_Linux/user/test/Makefile
  - CFLAGS_RELEASE += -DSPI
- Build program
  - make

Then user can run application as described in section 4.4.

4.3 Kernel compilation

Kernel compilation requires installation of kernel source headers and device tree blog to be run. The following sequence is required:

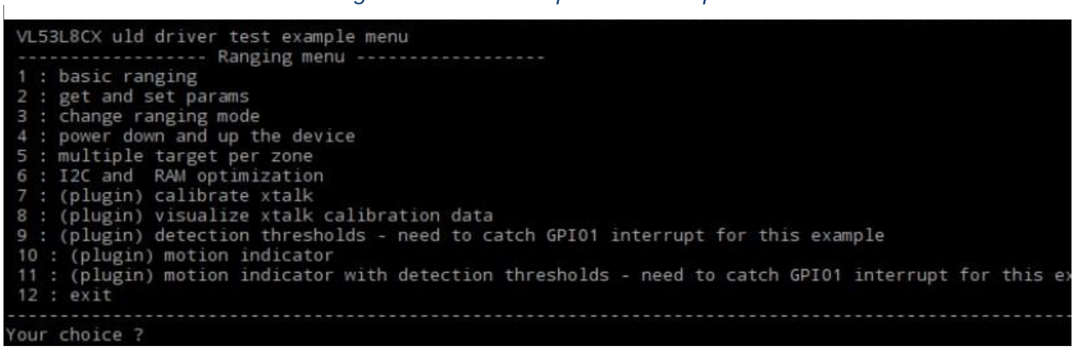
- Install the Raspberry PI Kernel source headers (refer to official documentation to download a version matching with kernel version)
- Update /boot/config.txt file (kernel mode only)
  - sudo nano /boot/config.txt
- Adding or un-commenting the following lines of file /boot/config.txt
  - dtparam=spi=on
  - dtoverlay=stmvl53l8cx
- Compile the device tree blob
  - cd VL53L8CX_ULD_Linux/kernel
  - make SPI=1 dtb
  - sudo reboot
- Go to test folder and enable kernel cflag into the test Makefile
  - cd VL53L8CX_ULD_Linux/user/test
  - nano VL53L8CX_ULD_Linux/user/test/Makefile
  - CFLAGS_RELEASE += STMVL53L8CX_KERNEL
- Build program
  - make
- Compile the kernel module
  - cd VL53L8CX_ULD_Linux/kernel
  - make clean
  - make
  - sudo make insert
- Then user can run application as described in section 4.4.

4.4 Run a test application

Once the compilation is done, user can go to the test folder and run the example menu. The menu allows using examples located in the ./examples folder.

- Go to test folder and run the test code.
  - cd VL53L8CX_ULD_Linux/user/test
  - ./menu

Figure 13: Screen capture of example menu



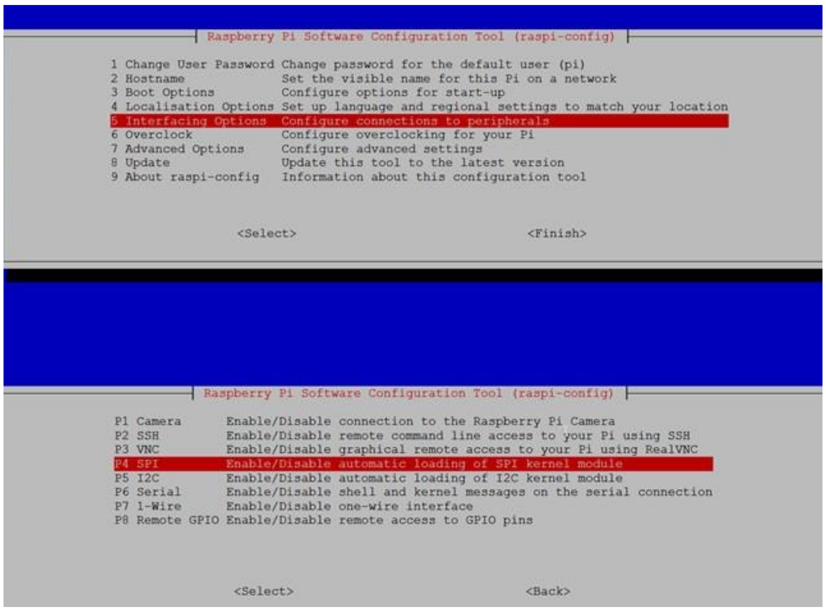
```
VL53L8CX uld driver test example menu
----------------- Ranging menu -----------------
1 : basic ranging
2 : get and set params
3 : change ranging mode
4 : power down and up the device
5 : multiple target per zone
6 : I2C and  RAM optimization
7 : (plugin) calibrate xtalk
8 : (plugin) visualize xtalk calibration data
9 : (plugin) detection thresholds - need to catch GPIO1 interrupt for this example
10 : (plugin) motion indicator
11 : (plugin) motion indicator with detection thresholds - need to catch GPIO1 interrupt for this ex
12 : exit
------------------------------------------------------------------------------------------------
Your choice ?
```

Note: Examples 9 and 11 require to catch the interrupt raised on INT pin. User needs to add wires and to implement GPIO callback for using such examples.

4.5 Multi device user space compilation and test

Run ‘sudo raspi-config’ in order to enable SPI interface in the linux kernel.

Figure 14: SPI RPI configuration

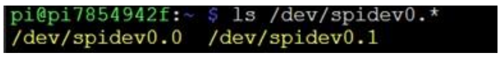


```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the default user (pi)
2 Hostname             Set the visible name for this Pi on a network
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

                <Select>                            <Finish>
```

```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

                <Select>                            <Back>
```

Then reboot the Raspberry PI. Afterwards check if /dev/spidev0.x exist.

Figure 15: SPI - checking if spidev0 exists



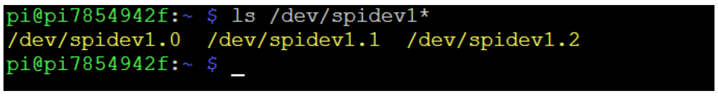
```
pi@pi7854942f:~ $ ls /dev/spidev0.*
/dev/spidev0.0  /dev/spidev0.1
```

In order to use more than 2 satellites the SPI1 should be enabled through adding the following in /boot/config.txt and reboot

```
# Additional overlays and parameters are documented /boot/overlays/README
dtoverlay=spi1-3cs
```

Afterwards check /dev/spidev1.x are existing

Figure 16: SPI - checking if spidev1 exists



```
pi@pi7854942f:~ $ ls /dev/spidev1*
/dev/spidev1.0  /dev/spidev1.1  /dev/spidev1.2
pi@pi7854942f:~ $ _
```

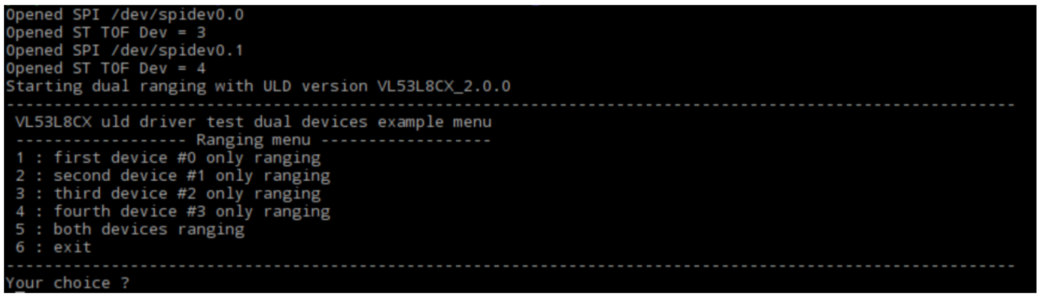
User space mode only requires to compile the makefile located into the ./test folder. The following sequence is required:

- Go to test folder
  - cd VL53L8CX_ULD_Linux/user/test
- Enable the SPI cflags option
  - nano VL53L8CX_ULD_Linux/user/test/Makefile
  - CFLAGS_RELEASE += -DSPI
- Build program
  - make

Once the compilation is done, user can run the multi_ranging example.

- Run the multi_ranging example.
  - ./multi

Figure 17: multi VL53L8CX SPI use space example.



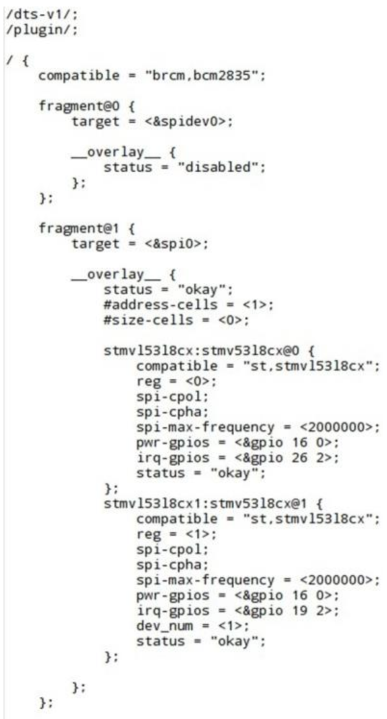
```
Opened SPI /dev/spidev0.0
Opened ST TOF Dev = 3
Opened SPI /dev/spidev0.1
Opened ST TOF Dev = 4
Starting dual ranging with ULD version VL53L8CX_2.0.0
----------------------------------------------------------------------------------------------------------
 VL53L8CX uld driver test dual devices example menu
 ------------------ Ranging menu ------------------
 1 : first device #0 only ranging
 2 : second device #1 only ranging
 3 : third device #2 only ranging
 4 : fourth device #3 only ranging
 5 : both devices ranging
 6 : exit
----------------------------------------------------------------------------------------------------------
Your choice ?
```

4.6 Multi device kernel space compilation and test

To enable multi-SPI devices in Kernel space, some modification needed.

- Update the device tree
  - cd VL53L8CX_ULD_Linux/kernel
  - sudo nano stmvl53l8cx_spi.dts
- Add the following line to enable 2nd SPI device, if need more than 2 devices, add similar modification

Figure 18: Adding second SPI device



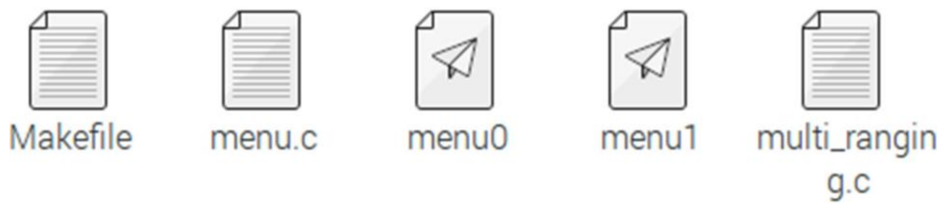
```
/dts-v1/;
/plugin/;

/ {
    compatible = "brcm,bcm2835";

    fragment@0 {
        target = <&spidev0>;

        __overlay__ {
            status = "disabled";
        };
    };

    fragment@1 {
        target = <&spi0>;

        __overlay__ {
            status = "okay";
            #address-cells = <1>;
            #size-cells = <0>;

            stmvl53l8cx:stmv53l8cx@0 {
                compatible = "st,stmvl53l8cx";
                reg = <0>;
                spi-cpol;
                spi-cpha;
                spi-max-frequency = <2000000>;
                pwr-gpios = <&gpio 16 0>;
                irq-gpios = <&gpio 26 2>;
                status = "okay";
            };
            stmvl53l8cx1:stmv53l8cx@1 {
                compatible = "st,stmvl53l8cx";
                reg = <1>;
                spi-cpol;
                spi-cpha;
                spi-max-frequency = <2000000>;
                pwr-gpios = <&gpio 16 0>;
                irq-gpios = <&gpio 19 2>;
                dev_num = <1>;
                status = "okay";
            };

        };
    };
```

for the 3rd and 4th devices on SPI1.

- Update /boot/config.txt file
  - sudo nano /boot/config.txt
- Adding or un-commenting the following lines of file /boot/config.txt
  - dtparam=spi=on
  - dtoverlay=stmvl53l8cx
- Compile the device tree blob
  - cd VL53L8CX_ULD_Linux /kernel
  - make SPI=1 dtb
  - sudo reboot
- Go to test folder and enable kernel cflags into the test Makefile
  - cd VL53L8CX_ULD_Linux/user/test
  - nano VL53L8CX_ULD_Linux/user/test/Makefile
  - CFLAGS_RELEASE += STMVL53L8CX_KERNEL
- Build program
  - make
- Rename ./menu to ./menu0.Then modify the device name for the second device:
  - nano VL53L8CX_MassMarket_Linux/user/platform/platform.c
    p_platform->fd = open("/dev/stmvl53l8cx1", O_RDONLY);
  - make

- Rename ./menu to ./menu1.

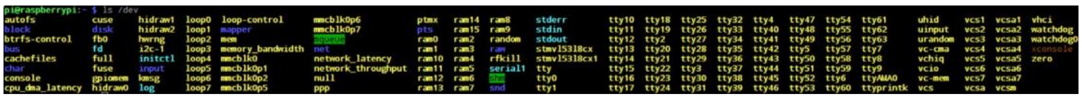
Figure 19: List of all files after renaming



- Compile the kernel module
  - cd VL53L8CX_ULD_Linux/kernel
  - make clean
  - make
  - sudo make insert

- Two devices should be visible under /dev folder.
  - sudo ls /dev

Figure20: Visible VL53L8CX devices



```
pi@raspberrypi:~ $ ls /dev
autofs           cuse     hidraw1  loop0  loop-control      mmcblk0p6           ptmx   ram14  ram8     stderr        tty10  tty18  tty25  tty32  tty4   tty47  tty54  tty61      uhid     vcs1  vcsa1  vhci
block            disk     hidraw2  loop1  mapper            mmcblk0p7           pts    ram15  ram9     stdin         tty11  tty19  tty26  tty33  tty40  tty48  tty55  tty62      uinput   vcs2  vcsa2  watchdog
btrfs-control    fb0      hwrng    loop2  mem               mqueue              ram0   ram2   random   stdout        tty12  tty2   tty27  tty34  tty41  tty49  tty56  tty63      urandom  vcs3  vcsa3  watchdog0
bus              fd       i2c-1    loop3  memory_bandwidth  net                 ram1   ram3   raw      stmvl5318cx   tty13  tty20  tty28  tty35  tty42  tty5   tty57  tty7       vc-cma   vcs4  vcsa4  xconsole
cachefiles       full     initctl  loop4  mmcblk0           network_latency     ram10  ram4   rfkill   stmvl5318cx1  tty14  tty21  tty29  tty36  tty43  tty50  tty58  tty8       vchiq    vcs5  vcsa5  zero
char             fuse     input    loop5  mmcblk0p1         network_throughput  ram11  ram5   serial1  tty           tty15  tty22  tty3   tty37  tty44  tty51  tty59  tty9       vcio     vcs6  vcsa6
console          gpiomem  kmsg     loop6  mmcblk0p2         null                ram12  ram6   shm      tty0          tty16  tty23  tty30  tty38  tty45  tty52  tty6   ttyAMA0    vc-mem   vcs7  vcsa7
cpu_dma_latency  hidraw0  log      loop7  mmcblk0p5         ppp                 ram13  ram7   snd      tty1          tty17  tty24  tty31  tty39  tty46  tty53  tty60  ttyprintk  vcs      vcsa  vcsm
```

- To run the test after the compilation just type
  - ‘./menu0
  - ‘./menu1

5 Document revision history

5.1.1.1 Table 1. Acronyms and abbreviations

Date	Version	Changes
11-Nov-2022	1.0	Initial release
12-Dec-2023	2.0	Added a section describing how to run example in SPI user space
07-Feb-2024	3.0	Updated various sections as driver now supports SPI in kernel mode
09-Dec-2024	4.0	Adding SPI user space support