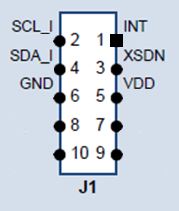
VL53L1X Ultra Lite Linux driver  
  
  
Compilation guide

&

Wiring example with VL53L1X satellite on Raspberry Pi3 board

# VL53L1X satellite connection on raspberry Pi3



Raspberry connector pins -> VL53L1X pins

3.3 V PWR pin 1 -> pin 5 (VDD)

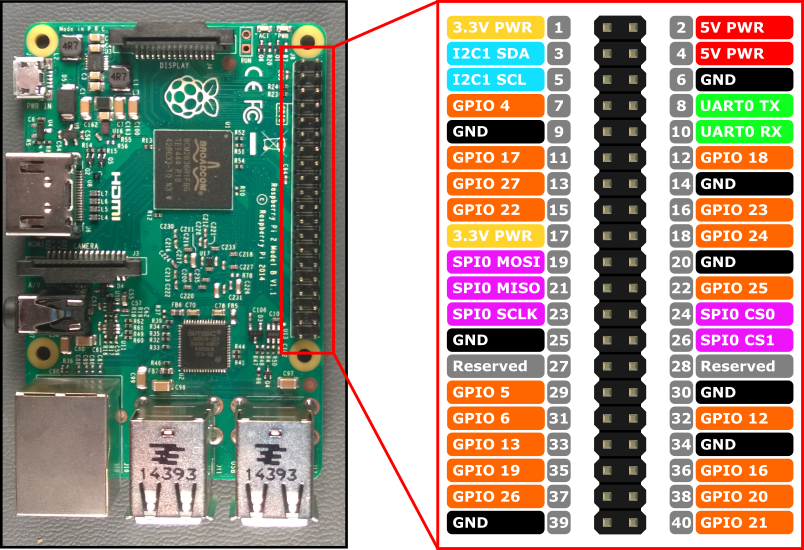
I2C1 SDA pin 3 -> pin 4 (SDA)

I2C1 SCL pin 5 -> pin 2 (SCL)

GPIO4 pin 7 -> pin 3 (XSDN)

GND pin 9 -> pin 6 (GND)

GPIO17 pin 11-> pin 1 (INT)



To release the XSDN and then run the VL53L1X the GPIO4 must be programmed as output and set active

sudo su

cd /sys/class/gpio

echo 4 > export

cd gpio4

echo "out" > direction

echo "1" > value

i2cdetect -r -y 1

expected output of i2cdetect command

0 1 2 3 4 5 6 7 8 9 a b c d e f

00: -- -- -- -- -- -- -- -- -- -- -- -- --

10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

20: -- -- -- -- -- -- -- -- -- 29 -- -- -- -- -- --

30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

70: -- -- -- -- -- -- -- --

Notice that the 29 is there (0x29 is the default I²C address of the VL53L1X device)

# User space library and example

The user space library found in Linux\_build\user\_lib directory is based on VL53L1X ULD bare driver and supports two operating modes

* The polling mode
  + the application uses the VL53L1X\_CheckForDataReady() function to poll while waiting for a ranging completion
* The interrupt mode
  + the application blocks on a device’s virtual file entry reading managed by a kernel module. The application is unblocked and continues its execution as soon as the kernel module received the end of ranging interrupt from the VL53L1X

An example of IOCTL + kernel module mechanism is coded in VL53L1X\_UltraLite\_WaitForInterrupt() function in vl53l1\_linux\_platform.c file.

Directions to compile the library and the example test application.

The Makefile shall be modified in order to match your own toolset.

The Makefile can be used as it is for a native compilation on Rapsberry using gcc toolset.

**Polling mode test application compilation** (no need for kernel module in such case)

make POLLING=1

To run the test after the compilation just type

./ultra\_lite

**Interrupt driven mode test application compilation**

In this case the kernel module compilation and insertion is required as described hereafter

make

# Kernel module for VL53L1X end of ranging interrupt support

The st\_tof\_module.c file found in Linux\_build\kernel\_module directory suggests a possible kernel module implementation to handle the interrupt accordingly to the expectation of VL53L1X\_UltraLite\_WaitForInterrupt() function.

Directions to compile the st\_tof\_module kernel module

The Makefile is designed to build the kernel module in native mode on raspberry

It relies on KDIR environment variable to find the kernel header files (generally located in /usr/src/linux-headers-xxx

The command ‘uname –a’ displays your kernel version

If those kernel’s header files are not installed the following command shall do it for you

sudo apt-get install raspberrypi-kernel-headers

Compilation of st\_tof\_module (example for a kernel version 4.9.35-v7)

// adapt the KDIR content to fit YOUR kernel version displayed by uname –a

export KDIR=/usr/src/linux-headers-4.9.35-v7+

make

Insert the module

sudo insmod st\_tof\_module.ko intr\_gpio\_nb=17

Note the intr\_gpio\_nb parameter is optional, it is 17 by default if it is not passed

It can be usefull if you select a different pin on raspberry Pi connector to wire the VL53L1X INT signal

Allow user space accesses to the /dev/st\_tof\_dev device created by this module insertion

(if you don’t you’ll need for sudo to launch ./ultra\_lite application)

sudo chmod 777 /dev/st\_tof\_dev

Then launch the test application compiled by make (without the POLLING define!)

./ultra\_lite