Raspberry Pi Communication

# Introduction

This documents presents the necessary programs and program sequence for a successful communication between the PC, Raspberry Pi, Robot Roomba and RoboPeakLidar.

# Files and directory’s used on Pi

* ~/server/udp\_receive.py

Communication between the computer and Roomba. No input required , because by using manufacturer information, the pi assigns automatically /dev/roomba when it is connected. A lot of information is given to the user by executing it in the terminal and showing the prinf’s.

* ~/server/udp\_lidar.py

Communication between the computer and Lidar. No input required, because by using manufacturer information, the pi assigns automatically /dev/rplidar when it is connected. A lot of information is given to the user by executing it in the terminal and showing the prinf’s.

* ~/RPLidarPi/Debug/RPLidarPi

This is the C++ folder and executable, which needs as input /dev/rplidar. This file is launched automatically by udp\_lidar.py..

* crontab

The crontab is used to execute a set of scripts at startup of the Pi. The crontab can be accessed with:

$ sudo crontab -e

# Communication with the iRobot Roomba (Roomba)

The Pi is used as a simple repeater. The correct sequence f bytes is made by a MATLAB toolbox, based on the Roomba 500 Open Interface Spec document.

The Pi knows when it has to merely repeat the bytes to the Roomba or when it has to read the serial port to send information back to the user (sensor measurements).

By using manufacturer information, the pi assigns automatically /dev/roomba when it is connected.

At this moment, all bytes have to be converted to strings to be send over UDP (Python2). Also, incoming data has to be converted from strings to bytes. In order to be more efficient, Python3 should be used to be able to send bytes over UDP (this will be much more efficient !). Also, using only the MATLAB udp communication could also speed up the process.

# Communication with the RoboPeakLidar (Lidar)

The Lidar is controlled by a C++ program, which transfers the measurement data over UDP to the local host on port 7070 (the raspberry pi) and then, to the PC on port 7070. This is done by the python program ~/server/udp\_lidar.py. The PC has to send the start byte 201 to the Pi before receiving any Lidar data. This, so that the Pi know the IP of the client

## C++ program quick start

In case you need to change something in the C++ code, here are the steps needed to compile a working executable for the Pi, using Eclipse. Skip this part if you don’t need to change anything to the existent code.

### Settings for Eclipse

1. Install Eclipse
2. Unzip C++ code (RPLidar.zip)
3. Create a C++ project in Eclipse
4. Drag the unzipped files in the project
5. Add the necessary paths in eclipse.
   * right click project
   * C/C++ build
   * Settings
     + GCC C++
       - Include, add the following paths (click on workspace, add graphically)
         * sdk/include
         * sdk/src
         * skd/src/arch
         * sdk/src/hal
6. Add the necessary libraries :
   * right click project
   * C/CC++ build
   * Settings
     + GCC C++
       - Linker libraries
         * pthread
7. Build
8. Left click on arrow next to run
   * Run configurations
   * new launch configurations
   * Arguments → /dev/ttyUSB0 (for example)

Or, use the backup settings :

### Cross compiling

By following the link above, or the CrossComple.pdf, you will learn how to cross compile a C++ program for the Raspberry Pi on a regular PC.

<http://www.hertaville.com/development-environment-raspberry-pi-cross-compiler.html>

$ scp -r path/to/your/directory/you/want/to/copy username@ipadress:/path/to/the/dir/you/want/to/put/your/folderss

## Increase Pi output current

In order to get currect measurements (not datasend=0), a smaal change has to be made in the config file of the boot partition of the pi. in /boot/config.txt : max\_usb\_current=1 (last line)

<http://answers.ros.org/question/208932/cannot-see-the-scan-data-of-rplidar-on-raspberry-pi/>