Problem solving guide

Intro

This document will take you over some systematic steps to analyse and identify where the problem In the communication between your computer, Pi, Roomba and Lidar. Numbers indicate the steps to be followed sequentially. Bullet points indicate the steps to take if there is a problem

Before anything else

1. A ssh connection is required to the Pi. In order to do this, please follow Pi Communication.
2. One major problem of this robotic platform, is due to the malfunction of the Roomba when it hits a low battery level. Typical problems, in order of decreasing battery level are :
   - Actuator power decreases
   - Sensor data becomes rubbish (especially the encoders, but after also tactical sensors)
   - Pi can get power, but can't make any connection with the Roomba

Pi Communication

1. Check connection with pi Where # is the number of your Roomba

```
$ ping 192.168.1.10#
```

- Are you connected to Wifi_Roomba ?
- Is the Pi connected to Wifi_Roomba ?
    - Connect Pi to a screen, reboot and watch booting sequence.
    - If Kernel Panic occurs, you will have to rewrite the SD card, with the backup
    - IF you arrive at the login, log : pi, password : pi
    - Check your IP address by

```
$ ifconfig
```

    - If you have a valid IP address, but it is not ending by the correct #, please contact a Teaching Assistant or prof. Slaets, and tell him that the static IP address has changed. This could be due to a change of MAC address of the Wi-Fi module (= is not the original Wi-Fi module). In the current implementation, it is the Rooter who manages the static IP addresses by checking the MAC addresses of each Wi-Fi module/dogle connected to the pi.
    - If you don't see any valid IP address under wlan0 try

```
$ wicd-curses
```

    - Follow the instructions on the interface and try to connect to Wifi_Roomba

2. Connect via ssh to the pi to gain access to the terminal. Password is pi. Now that your are connected to the pi, we can start debugging the other problems

```
$ ssh pi@192.168.1.10#
```

Roomba Communication

1. Once you're are connected via the Pi kill any processes that could interfere with the debugging :

```
$ ps aux | grep udp
```

This will show all the programs that contains "udp". You will have to kill the program "udp_receive" with the correct "process id", numbers (4 normally) at the start

```
$ sudo kill ####
```

2. Go to the server directory, in order to execute the udp_roomba and to get feedback of the running python program

```
$ cd server/
$ python udp_receive
```

3. Now, the Roomba will beep and go in full modus (buttons on the Roomba don't work).
   - If this doesn't work, the chances are high that the Roomba has a low battery level (see Before anything else)
4. Send the commands to the Roomba with your MATLAB script. Normally you will see :

```
DATA RECEIVED = [#########]
DATA SEND = [#########]
```

- If data send = [], your Pi hasn't received any information from the Roomba, this will usually mean that the Roomba has a low battery level.
- If data send is not in a correct space, incoherent data bytes (cfr. Roomba Open Interface), this will usually mean that the Roomba has a low battery level.
- If Data send is what you expect, but you don't receive it on your PC then :
  - Go to the server folder of the RoombaCommunicationToolbox
  - In getDataUDP, display the string data you get by displaying it. If this is good, this means something is going wrong in the conversion to bytes.

Lidar Communication

1. Once you're are connected via the Pi kill any processes that could interfere with the debugging :

```
$ ps aux | grep lidar
```

This will show all the programs that contains "lidar". You will have to kill the program "udp_lidar" and "……RPLidarPi/Debug/./RPLidarPi" with the correct "process id", numbers (4 normally) at the start. Having multiple instances of the latter one (c++ code for the lidar communication) will definitely make your program send "0 data".

```
$ sudo kill ####
$ sudo kill ####
```

2. Go to the server directory, in order to execute the udp_lidar and to get feedback of the running python program and follow the instructions / information on the screen.

```
$ cd server/
$ python udp_receive
```

3. First, the code waits for the start signal “201” on port “7071”. By sending this code, the program will start and now the IP address of the client
4. Now, the python code will run the C++ program, which will also display some information.
    - If segmentation error occurs, kill all programs as described in step 1 and execute program again in step 2. This will normally solve the problem

SD Card backup

To be more flexible, we will back up the root and the boot partition separately.

1. Identify the location of the sd cart. Partition name will be boot(1) and root(2).

```
df -h
```

2. Backup up boot (* the letter you found in step 1, b if you have one partition on your computer). Take a short time (seconds). This will make boot.iso in home folder. If is input file, of is output file, order is NOT important.

```
sudo dd if=/dev/dev/sd*1 of=~boot.iso
```

3. Backup up root (* the letter you found in step 1, b if you have one partition on your computer). Takes up to 30min. This will make root.iso in home folder

```
sudo dd if=/dev/dev/sd*2 of=~root.iso
```

4. Copy to clean sd. Small preparation in Gparted (gui for partition manager). Make a first partition on 100MB formatted to fat16. Rest is ext4. Make sure the size of your root.iso is smaller as this second partition. If this is not the case, use Gparted to shrink the original SD cart, and repeat step 3. Copying root.iso can take up to 30min

```
sudo dd of=/dev/dev/sd*1 if=~boot.iso bs=4M
sudo dd of=/dev/dev/sd*2 if=~root.iso bs=4M
```

If your SD cart doesn’t work (kernel panic), don’t use the bs=4M, but much slower copying (up to one hour).

MISC: Useful command codes

Listen to port #### :

```
$ netcat –ul ####
```