Table of Contents

Welcome

Project Willy

Getting started

Build of Willy

Architecture

Raspberry Pi’s

Components

Lessons learned

Archive

Guide to Linux

1. Introduction

On Willy the Robot, the four Raspberry Pi’s and the laptop use Linux as their operating system. This is an alternative operating system with an emphasis on using the Command Line Interface (CLI) besides the graphical environment. On the Raspberry Pi’s, the CLI is your only interface most of the time.

2. Remote access via the Secure Shell protocol (SSH)

The Raspberry Pi’s are all mounted in a way that connecting a display is possible, but peripherals as a keyboard might be inconvenient.

Using SSH can be done via the terminal on the laptop mounted on WTR, or via the application PuTTY on Windows on your own PC.

> ℹ Using your own PC requires you to be connected to WTR’s Network via its Wi-Fi access point or an ethernet cable. (Be cautious when moving the robot if you do.)

Connecting to a remote raspberry pi or laptop can be done by invoking the SSH client:

```
ssh <username>@<ip address or hostname>
```

You will be prompted for a password, after entering that you will be logged in to the remote server.

If you wish to do this via your own PC, connect to the network of WTR, then use any SSH client to connect to the desired remote raspberry pi or laptop using the same credentials.

3. Commonly used commands

The following commands might often be used within Linux, as an example, the "sensor node" raspberry pi will be used.

You can see the current folder you are in by the tooltip before your cursor. For instance: `ubuntu@sensornode:~/catkin_ws/$` shows that you’re within the `catkin_ws` directory.

> ℹ The `~` within a path resembles the home directory of the user. By default, this is `/home/<username>`

Commands are often used with a path to the folder or directory following. Example: viewing the script for an IMU sensor from the laptop would need this list of commands:

```
willy@willy:~$ ssh ubuntu@sensornode
<password authentication>
ubuntu@sensornode:~$ cd ~/catkin_ws/src/demo_package
ubuntu@sensornode:~/catkin_ws/src/demo_package/$ ls
demo_script.cpp
ubuntu@sensornode:~/catkin_ws/src/demo_package/$ nano demo_script.cpp
```

The following table will list several commonly used commands or applications. Some of them might need elevation (done by typing `sudo` before the command) to be allowed to execute or edit some files.

> 💡 If you are told permission is denied after entering the command already, using `sudo !!` will rerun that command with elevated privileges.

Command	Elevation required	Description
ls	-	List the contents of the current directory
cd	-	Short for "Change Directory"
nano	~	A text editor commonly included within Linux distributions, very useful for quickly viewing and editing files. To save files, use Control+X, press Y to confirm saving the file, you're then offered the opportunity to change the filename you're saving to. Press Enter unless you need to. This would also automatically exit the editor. If permission is denied to save the file, use `sudo` before invoking the editor.
vi	~	An alternative text editor that is built within Linux itself, it is slightly more difficult to beginners due to its two "modes" of operation, to type something in the file, press `I` first, then if you're done, press `ESC`, then type in `:wq` to save the file and quit the editor
htop	-	An improved version of the built in `top` task monitor. The F1-10 buttons have functions listed below on the screen, and the arrow keys lets you navigate the list of tasks. Hopefully, you do not need to use this unless a process needs to be killed.
ping	-	Ping lets you test whether or not a node is running or if you're connected to the network. For best results, you use an IP Address, such as the one for WTR's Router Node.
dmesg	-	In the event that a module on the sensor node calls the wrong arduino, this command will show a system log file that can tell you on which "port" an arduino is connected. such as `/dev/ttyACM0`. Tip is to reconnect the necessary arduino when you do so, so that it is shown more recently in the `dmesg` output.

3.1. Command Aliases

Certain commands often wind up to be too long to repeat every so often. For this, the command line interface allows you to configure aliases for these frequently used commands.

Aliases are stored within a hidden file in the user's home folder. It is usually called `.bash_aliases`. To edit this, change to the home directory using `cd ~` and then open the editor of choice: `nano .bash_aliases`

You will see output like this:

```
alias startmotorcon='sudo chmod 777 /dev/ttyACM0 && rosrun rosserial_python serial_node.py _port:=/dev/ttyACM0'
alias startjoystick='sudo chmod 777 /dev/input/js0 && roslaunch teleop_twist_joy teleop.launch'
alias startwilly='cd ~/Documents/willy && ./start-willy.sh'
alias controlwilly='python /home/willy/Documents/controlWilly/controlWilly.py'
alias rqtconfig='rosrun rqt_reconfigure rqt_reconfigure'
alias rqtgraph='rosrun rqt_graph rqt_graph'
alias rqtconsole='rosrun rqt_console rqt_console'
```

Aliases are defined by starting a line with `alias` followed by the new name for your command. The apostrophes are of importance and could be swapped with their double variants "" if need be.

Test your preferred alias command before editing this file to make sure you can use it.

First, you will have to make sure that the intended command returns your expected results, such as changing to a directory or starting a node. Then, copy that command over to a different text editor or a physical notepad. Using the above example, add a new line to the file with your preferred editor, adhering to the syntax.

When finished, exit the editor after saving the file. Then in the CLI, enter:

```
source .bashrc
```

and try out your new command. If done well; it should function immediately. If not, an error message could show up and all aliases would not work.