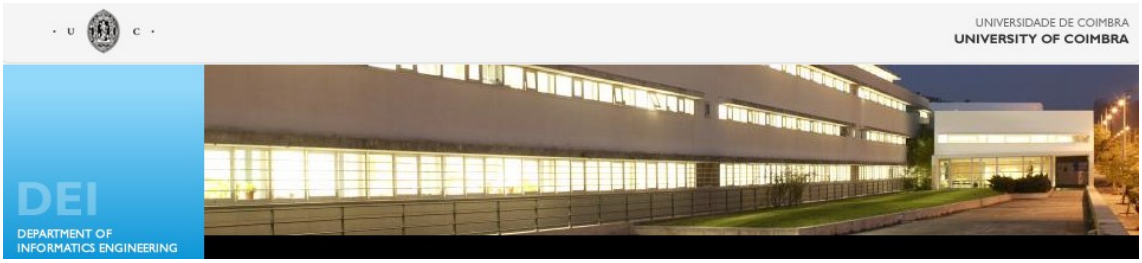

Sensing and Actuation Networks and Systems [2021-2022]

Assignment 02 – Setting up the development environment

Introduction

This assignment aims to set up and configure the development environment needed for future assignments. Students will get familiarised with Python virtual environments and the Raspberry Pi devices. The GPIO (Generic Purpose Input/Output) interface of the Raspberry Pi is introduced and configured.

Objectives

Students successfully concluding this work should be able to:

- Set up a Python virtual environment
- Install Python GPIO packages with pip
- Configure the Raspberry Pi GPIO interface

Support Material

- Tutorial for creating Python virtual environment and configuring the GPIO interface
- Raspberry Pi 2 Model B with net cable (for use during PL)

Theoretical background

This section covers some concepts that will be used during this assignment.

Python virtual environments

A virtual environment[1] is a Python environment such that the Python interpreter, libraries and scripts installed into it are isolated from those installed in other virtual environments, and (by default) any libraries installed in a "system" Python, i.e., one which is installed as part of the operating system.

A virtual environment is a directory tree which contains Python executable files and other files which indicate that it is a virtual environment. When working in a command shell, users can make a virtual environment active by running an `activate` script in the virtual environment's executables directory.

When your Terminal has a Python virtual environment activated, all Python related activity is sandboxed to your virtual environment. To leave an activated virtual environment, use the `deactivate` command. Remember to type `deactivate` to leave a virtual environment, not exit. If you type exit in a virtual environment, it will exit the Terminal.

Raspberry Pi

The Raspberry Pi[2] is a tiny, single-board, affordable computer developed in the UK, originally designed with the purpose of teaching basic computer science in schools and in developing countries, but currently is used for many purposes including weather monitoring and home and industrial automation.

There are three series of Raspberry Pi and several generations of each have been released. For this course, we will be using the Raspberry Pi 2 Model B.

The GPIO interface

The GPIO (General-Purpose Input/Output)[3] interface is a standard interface used to connect microcontrollers to other electronic devices. For example, it can be used with sensors, diodes, displays, and System-on-Chip modules. The GPIO can be used in three modes:

- *Input*: the default mode, where it is possible to receive input from the connected device via GPIO. For example, to read data from the status of a button (i.e., on or off);
- *Output*: to deliver data to the connected device, for instance to switch a led lamp;
- *UART interface*: the *Universal Asynchronous Receiver-Transmitter* mode enables the definition of custom advertising packets.

For this course, we will be focusing on the *input* and *output* modes.

[1] https://docs.python.org/3/library/venv.html
[2] https://www.raspberrypi.org/
[3] https://community.estimote.com/hc/en-us/articles/217429867-What-is-GPIO-

Exercises

The Raspberry Pi have been previously configured with the following addresses:

Raspberry Pi	Name	DNS name	Address
srsa01	raspberrypi01	srsa-pi-1	10.6.1.2
srsa02	raspberrypi02	srsa-pi-2	10.6.1.3
srsa03	raspberrypi03	srsa-pi-3	10.6.1.4
srsa04	raspberrypi04	srsa-pi-4	10.6.1.5
srsa05	raspberrypi05	srsa-pi-5	10.6.1.6
srsa06	raspberrypi06	srsa-pi-6	10.6.1.7
srsa07	raspberrypi07	srsa-pi-7	10.6.1.8
srsa08	raspberrypi08	srsa-pi-8	10.6.1.9
srsa09	raspberrypi09	srsa-pi-9	10.6.1.10

Test the connectivity from your PC to the Raspberry Pi using the following command:

```
ping <ip_address>

ping <dns_name>
```

You should see the exchanged packets between the PC and the Raspberry Pi. Notice the fields reported by the exchanged packets.

Creating a virtual environment on the Raspberry Pi

Goal: Understand the concept of virtual environments. Set up the development environment that will be used in following assignments of the course.

Activities: Connect to the Raspberry Pi using the SSH protocol. From a console use the following command:

```
ssh srsa@<ip_address>
```

And authenticate with the credentials provided at the beginning of this document. For Windows PCs, you can use the Putty utility tool to connect via SSH.

Create a directory where you will store your code. Use a distinctive name in order to differentiate your directory from other students using the same device. Change to this directory.

```
$mkdir /user/<your_directory>
$cd /user/<your_directory>
```

From this directory, execute the following command, which creates a new Python virtual environment using the `venv` tool. The `-m venv` tells the Python interpreter that we are going to run the `venv` module and the `<your_venv>` parameter is the name of the folder where the virtual environment will be created.

```
python -m venv <your_venv>
```

Once again, use a distinctive name for your virtual environment. To use a Python virtual environment, you must *activate* it, as following:

```
#From the folder <your_directory>
$ source venv/bin/activate
(venv) $
```

When the terminal has a Python virtual environment activated, all Python related activity is sandboxed to it. Identify the changes in the terminal when activating the virtual environment.

Installing Python GPIO packages with pip

Goal: Learn the basic concepts related to package installation and management by installing the GPIO-related packages.

Activities: Install the following two packages:

- *GPIOZero library*, an entry-level and easy to use GPIO library for controlling simple electronics;
- *PiGPIO library*, an advanced GPIO library with many features for more complex electronic interfacing.

Use the `pip` command (Python Install Packages) to install the packages. As first step, upgrade the pip tool, using the following command:

```
(venv) $ pip install –upgrade pip
```

This command might take some time to complete, and potentially output a lot of text to the Terminal. With pip upgraded, confirm which Python packages are already installed. For this, use the following command:

```
(venv) $ pip list
```

Install the GPIO packages using the pip install command as follows

```
(venv) $ pip install gpiozero pigpio
```

Execute the pip list command again and notice if there is any difference. Take a snapshot of the packages you have previously installed using the pip freeze command:

```
(venv) $ pip freeze > requirements.txt
```

This command *freezes* all installed packages into a file named requirements.txt, which is a common filename to use for this purpose. Look inside your requirements.txt file and notice its content. You can use the following command:

```
(venv) $ cat requirements.txt
```

This is a good practice in case if you move your Python project to another machine or a new virtual environment. You can use the `requirements.txt` file to install all required packages for your project to work. Notice that whenever you install new packages with pip install you also

will need to re-run `pip freeze > requirements.txt` to capture new packages and their dependencies.

Using Wireshark

Goal: Use Wireshark to display information about the connections on the previous activities.

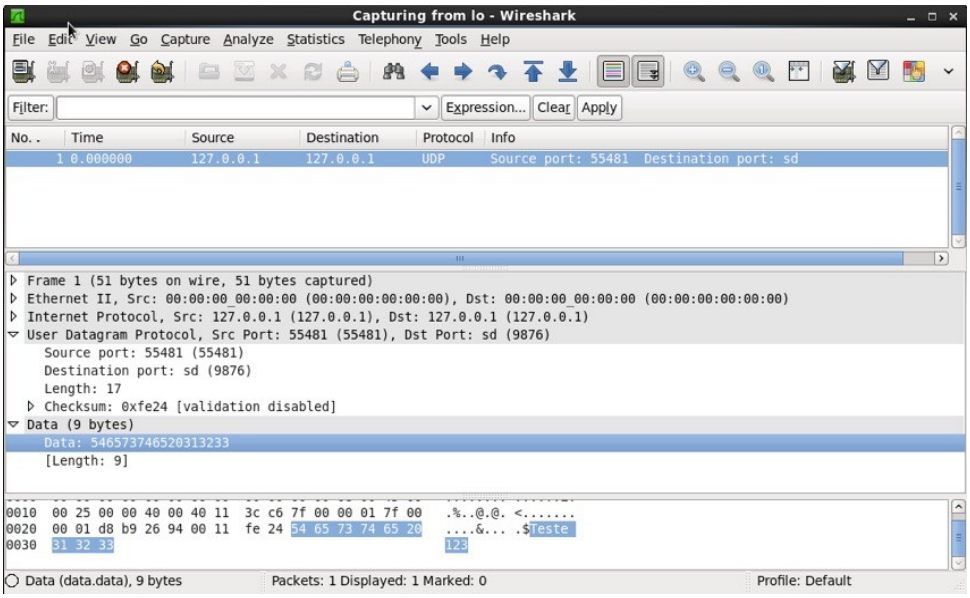

Figure 1. Example of capture from Wireshark

Activities: Using Wireshark, identify the communication flow from the previous exercises (ping command, SSH communication). Use the filters `ip.addr==<address>`, `tcp.port==22`, and `ip.proto==icmp` or `ip.proto==icmpv6` to facilitate the analysis. Identify the packets exchanged in the activities from this assignment.