

Machine Learning for IoT

Lab 1 – Getting Started

HowTo 1: Working Environment Setup

You are required to install a collection of tools that will be used across the lab activities. To do that, you can follow the instructions reported below.

NB: We strongly suggest using a Unix operating system on your PC. All the instructions provided refers to Ubuntu.

- 1.1. In future labs, we will adopt the MQTT protocol to exchange messages between the Raspberry and the PC. For testing purposes, we will use the *Mosquitto* tool.

Setup the Mosquitto MQTT clients:

- Open a new shell and enter the following commands:

```
sudo apt update
sudo apt install -y mosquitto-clients
```

- 1.2. Download and install Python 3.7 (if not already installed)

```
sudo apt install -y python3.7 python3.7-venv
```

- 1.3. Setup the Python environment:

- Open a new shell.
- Create a new folder and create a new Python virtual environment named *py37*

```
python3.7 -m venv py37
```

- Activate the virtual environment:

```
source py37/bin/activate
```

NB: the virtual environment must be activated every time you open a new shell

Suggestion: check the official Python documentation at

<https://docs.python.org/3/tutorial/venv.html>

- 1.4. Setup the required python packages:

- Download from the *Portale* the *requirements.txt* file
- Upgrade pip:

```
pip install -U pip
```

- Install the requirements running the pip command:

```
pip install -r requirements.txt
```

HowTo 2: Raspberry Board Setup

2.1. Download and setup the Raspberry Pi OS:

- Download the OS image from the official Raspberry website: https://downloads.raspberrypi.org/raspios_full_armhf_latest
- Unzip the downloaded file.
- Download and install *balenaEtcher* from <https://www.balena.io/etcher/>
- Insert the SD card on your PC.
- Run *balenaEtcher* and flash the OS image in the SD card.
- Create a void file named `ssh` inside the main root of the SD card.
- Insert the SD card on your Raspberry.
- Power on the Raspberry connecting the board to the power supply.

2.2. Connect to the Raspberry via the `ssh` command:

- Connect the Raspberry to your PC via the ethernet cable
- Open a new shell and enter the following command:

```
ssh pi@raspberrypi.local
```

- Insert the default password (`raspberry`)

2.3. Setup the WiFi Connection on your Raspberry through the *raspi-config* utility

- Enter the following command:

```
sudo raspi-config
```

- Open Network Options, then Wireless LAN, and enter your network SSID and password.

2.4. Setup the Mosquitto MQTT broker and MQTT clients

- Enter the following commands:

```
sudo apt update
sudo apt install -y mosquitto mosquitto-clients
```

2.5. Setup the Python virtual environment

- Create a new folder and create a new Python virtual environment named *py3*

```
mkdir WORK_DIR
cd WORK_DIR
python3 -m venv py37
```

- Activate the virtual environment:

```
source py37/bin/activate
```

NB: the virtual environment must be activated every time you log-in into the board

2.6. Setup the required python packages:

- Download from the *Portale* the *rpi_requirements.txt* file.
- Copy the file to the *WORK_DIR* folder of your Raspberry (suggestion: use the Unix `scp` command).
- Install the requirements running the `pip` command (from the Raspberry):

```
pip install -r rpi_requirements.txt
```

2.7. Setup TensorFlow 2.3.0:

- Download from the *Portale* the TensorFlow wheel
- Copy the file to the *WORK_DIR* folder of your Raspberry
- Install the package running the `pip` command:

```
pip install tensorflow-2.3.0-cp37-none-linux_armv7l.whl
```

2.8. Install other dependencies:

```
sudo apt install -y libgpiod2
```

HowTo 3: Test the MQTT Connection

Mosquitto will be used to check if your PC and the Raspberry can communicate and exchange messages between each other.

3.1. Run a subscriber on your PC:

- Open a shell on your PC.
- Enter the following command:

```
mosquitto_sub -h raspberrypi -t test
```

3.2. Run a publisher on your Raspberry:

- Open a shell on your board.
- Send the “Hello World!” message with the following command:

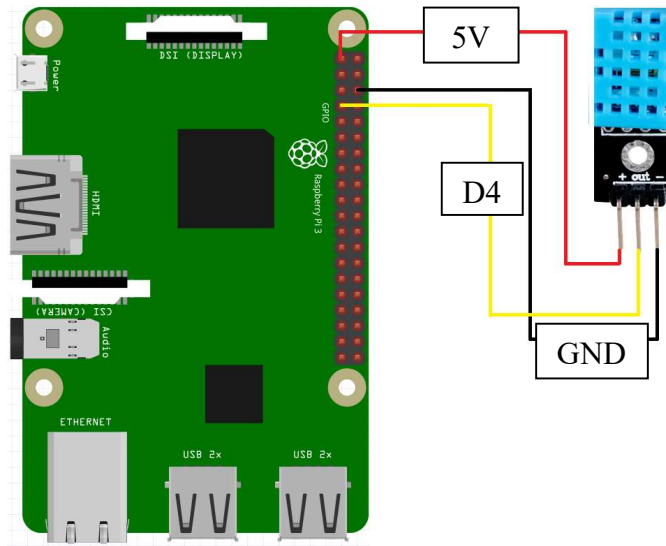
```
mosquitto_pub -h localhost -t test -m "Hello World!"
```

- Check if your PC has received the message on the subscriber shell.

Exercise 4: Sense Temperature and Humidity with the DHT-11

4.1. Write a Python script to collect temperature and humidity measurements with the Raspberry and the DHT-11 sensor. The script should take as inputs three parameters: the frequency of measurements (in seconds), the period of measurements (in seconds), and the output filename.

- Connect the DHT-11 sensor to your board.



- Use the `adafruit_dht` package to read temperature and humidity values:

```
from board import D4
import adafruit_dht

dht_device = adafruit_dht.DHT11(D4)
temperature = dht_device.temperature
humidity = dht_device.humidity
```

- Store the measurements in a csv file, where the first column reports the date (in *dd/mm/yyyy* format), the second the hour (in the *hh:mm:ss* format), the third the temperature value, and the fourth the humidity value.

Suggestion: Check the *datetime* Python package
<https://docs.python.org/3.7/library/datetime.html>

- Test your script with different values of frequency and period.
- Output example with frequency=5s and period=20s:

```
18/10/2020,09:45:34,21,65
18/10/2020,09:45:40,21,65
18/10/2020,09:45:45,21,65
18/10/2020,09:45:51,21,65
```

Exercise 5: Record Audio with the USB Microphone

- 5.1. Write a Python script to collect audio samples with the Raspberry and the USB microphone.
- Connect the USB microphone to the Raspberry.
 - Use the *pyaudio* package to drive the microphone (with the blocking mode) and the *wave* package to store the samples on disk.

Suggestion: check the packages documentation at
<http://people.csail.mit.edu/hubert/pyaudio/docs/#>
<https://docs.python.org/3/library/wave.html>

- Write the code to register a 3 second sample. Try different resolutions (Int8, Int16, Int32) and sampling rates (44.1 kHz and 48 kHz).
- Measure the execution time (in seconds) needed for sampling with the *time* method of the *time* package.
- Measure the execution time (in seconds) needed to store the data on disk.
- Measure the output .wav size (in KB) with *os.path.getsize* method from the *os* package
- Check the audio quality in the different cases and comment the results.

Exercise 6: (optional) Take Pictures with the PiCamera

- 6.1. Setup the PiCamera
- Connect the PiCamera to the Raspberry.

Suggestion: check the documentation at
<https://projects.raspberrypi.org/en/projects/getting-started-with-picamera>

- Enable the camera through the `raspi-config` utility:

```
sudo raspi-config
```

Select *Interfacing Options*, then *Camera*, and hit *Enter*. Choose *Yes*, then *Ok*. Go to *Finish* and you will be prompted to reboot.

- 6.2. Write a Python script to take pictures with the Raspberry and the PiCamera. The script should take as inputs: the image width (in pixels), the image height (in pixels), the framerate (in FPS), the number of pictures, the output format (png or jpg), the output directory.
- Use the *picamera* package to drive the camera module in Python.
 - Measure the execution time (in seconds) needed for taking a picture and storing the picture on disk.
 - Measure the output file size (in MB) with the *os.path.getsize* method.
 - Test the script with different resolutions (e.g., 64x64 and 1024x1024) and output formats. Comment the results.