An introduction to Python and Raspberry Pi

Microprocessor Systems and Applications

GITHUB.COM/CLAUDIUGROZA/MSA

February 2018

Contents

1. Python

1.1 Virtual environment

Roughly taking a description from the official documentation [Vir], a virtual environment is described as:

> "a tool to keep the dependencies required by different projects in separate places, by creating virtual Python environments for them. It solves the “Project X depends on version 1.x but, Project Y needs 4.x” dilemma, and keeps your global site-packages directory clean and manageable"

1.1.1 Instalation and basics

Virtual environment can be installed using the *pip* installer:

```
$ sudo pip install virtualenv
```

The creation of a virtual environment in the current directory is made using the following command:

```
$ virtualenv venv
```

> where *venv* is a newly created directory containing a copy of the Python distribution

Before you start working with the previously defined virtual environment, you must first enable it:

```
$ source venv/bin/activate
```

> where *source* is the working directory for your project

You are now able to observe the *venv* prefix on the left side of the prompt:

```
(venv)Computer:Source UserName$
```

Starting with this point any package that is installed using *pip* will be placed in *venv* directory defined earlier and it will be completely isolated from your global Python installation.

Once you have finished working with the virtual environment mode, it can be disabled using the following command:

```
$ deactivate
```

Now, the global Python interpreter was switched back to default.

1.2 Command line interpreter

The purpose of this section is to make a quick introduction to how the interpreter actually works.
The following command will start the Python interpreter [Int] in an interactive mode:

```
$ python
```

1.2.1 Interactive mode

The mode expects to take a command within the primary prompt ″> > >″. In case of continuation lines (block statement) you should expect an secondary prompt indication ″. . .″.

We will start with a basic example that prints all the integers from 0 to 4 :

```
>>> for i in range(0,5): #primary prompt
...     print i           #secondary prompt
...
```

You should expect to see the following output:

```
0
1
2
3
4
```

The *quit* command will close the current interactive mode session:

```
>>> quit()
```

The next example shows a basic if-else statement:

```
>>> if True == True:
...     print "I'm always right."
... else:
...     print "I'm never right."
...
```

1.3 Documentation and resources

Let us suppose we are interested in a quick overview of a type, for example an integer. The next command will print a wide description of the *Int* class:

```
>>> help(int)
```

In most of the cases, our interest will be limited to the functionality defined by that type:

```
>>> dir(int)
```

Expect to see a long list of methods:

```
['__abs__', '__add__', ... ]
```

Next step will be to access the docs for *abs* method. A wild guess given by its name would be that returns the absolute value of an integer. To be sure about this intuition we can check it with the following command:

```
>>> abs.__doc__
```

2. Raspberry Pi

2.1 Introduction

Raspberry Pi is a series of small single-board computers that was originally developed to promote basic computer science education and has since been popularized by open source community and makers of Internet of Things (IoT) devices. The popularity of Raspberry Pi comes from two reasons:

1. the power of interfacing with devices using **GPIO - General Purpose Input Output** [Ras]
2. the ability to run an operating system

2.2 Raspbian and RPi.GPIO library

In order to gain access to the GPIO block, the most common approach would be to use a Python wrapper. A library instance that facilitates access to the hardware features of the board is named **RPi.GPIO**. The newest versions of Raspbian OS come with the library preinstalled. However, if you need to install the library then the following fetch-install commands should accomplish the purpose:

```
$ sudo apt-get update
$ sudo apt-get install rpi.gpio
```

2.3 The first circuit

A possible wiring configuration is illustrated by Figure 2.1.

Considering we already have a board running Raspbian, the following parts are required to complete the first circuit:

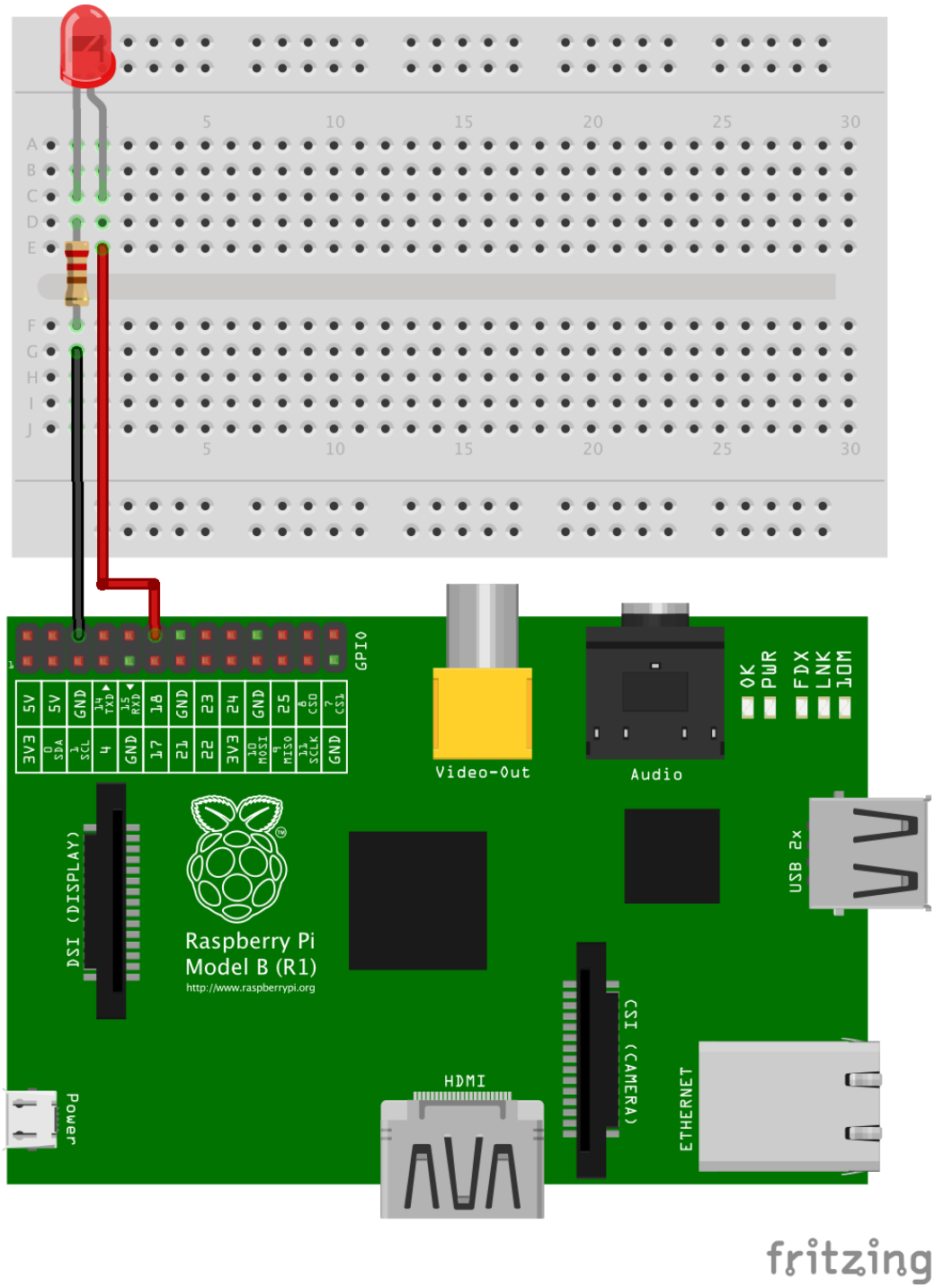

Figure 2.1: LED wiring example

- a small sized breadboard
- one LED
- one 330 Ohm resistor
- two jumper wires

We consider there are two formats of numbering the pins from the board: *BCM* and *BOARD*. From now on, the *BOARD* format will be used in our coding examples. The pin mapping numbers are shown in Figure 2.2.

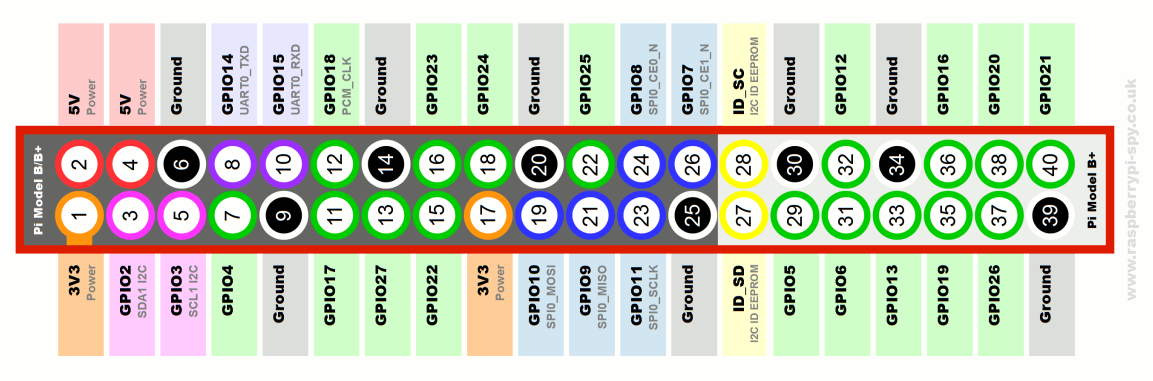

Figure 2.2: BOARD numbering format

2.4 The first coding example

The next example shows how we can control the LED circuit we introduced in Section 2.3. We start by invoking the Python interpreter:

```
$ python
```

The GPIO wrapper needs to be imported before using any associated functionality:

```
>>> import RPi.GPIO as GPIO
```

Like we mentioned before, the physical pins from the board can have different numbering schemas. We need to specify which numbering convention to use:

```
>>> GPIO.setmode(GPIO.BOARD)
```

The following command will configure **pin number 12** for **OUTPUT** direction mode:

```
>>> GPIO.setup(12,GPIO.OUT)
```

It is time to see some action by switching the LED to **ON** state. This means we need to apply a high voltage state to the output pin:

```
>>> GPIO.output(12,GPIO.HIGH)
```

Finally, to switch the LED back **OFF** we need to apply a low voltage state:

```
>>> GPIO.output(12,GPIO.LOW)
```

A healthy habit is to always free up the hardware resources once we finished using them:

```
>>> GPIO.cleanup()
```

3. Assignments

1. Write a program that toggles every second the state of an LED. E.g. 1 second ON, 1 second OFF, then repeat.
2. Extend the program from 1) by adding a new LED part. The second LED should have the inverted state of the first LED. E.g. (LED_1 = ON, LED_2 = OFF), (LED_1 = OFF, LED_2 = ON), then repeat.

4. Bibliography

4.1 References

[Int] *Python Interpreter*. https://docs.python.org/2/tutorial/interpreter. [Online; accessed February-2018]. 2018 (cited on page 6).

[Vir] *Python Virtual Environment*. http://docs.python-guide.org/en/latest/dev/virtualenvs/. [Online; accessed February-2018]. 2018 (cited on page 5).

[Ras] *Raspberry Pi Docs*. https://www.raspberrypi.org/documentation/usage/gpio/. [Online; accessed February-2018]. 2018 (cited on page 9).

4.2 Image credits

- First page illustration.
  http://www.northeastern.edu/levelblog/2018/01/25/guide-iot-careers
- Chapter header background.
  https://blogs.microsoft.com/iot/2015/03/17/simplifying-iot/
- Circuit schematics generated with Fritzing.
  http://fritzing.org
- Raspberry Pi numbering schema.
  https://www.raspberrypi-spy.co.uk