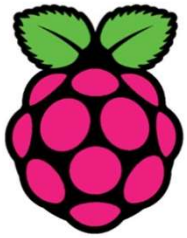
IoT Workshop

Building IoT application using open source tools

13/05/2017
TWESD'17, Monastir, Tunisia.



RaspberryPi

Introduction

In this tutorial, users will be guided to create a connected light using a Raspberry Pi. The connected light can be controlled and monitored from a mobile, tablet and desktop.

Software and hardware setup

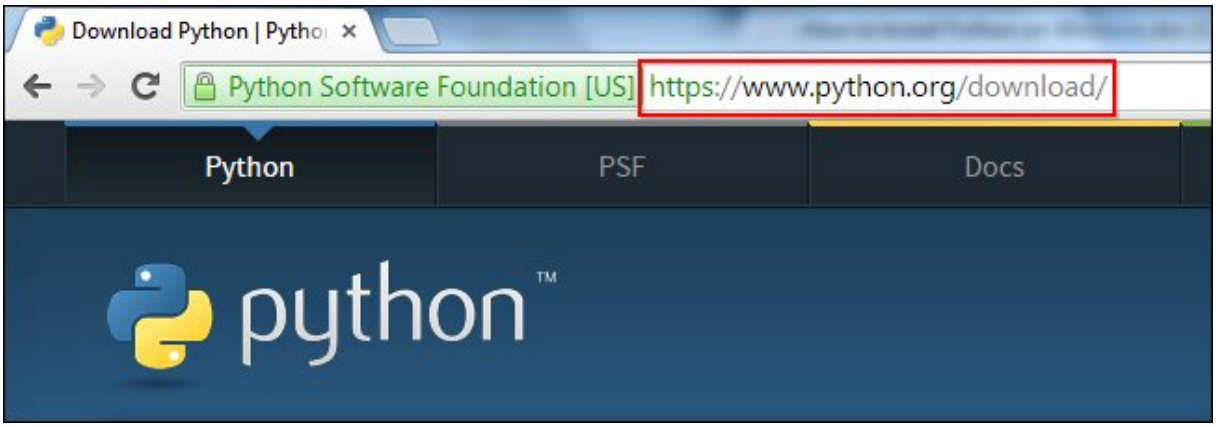
Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

For more details about this language: www.tutorialspoint.com/python/python_overview.htm

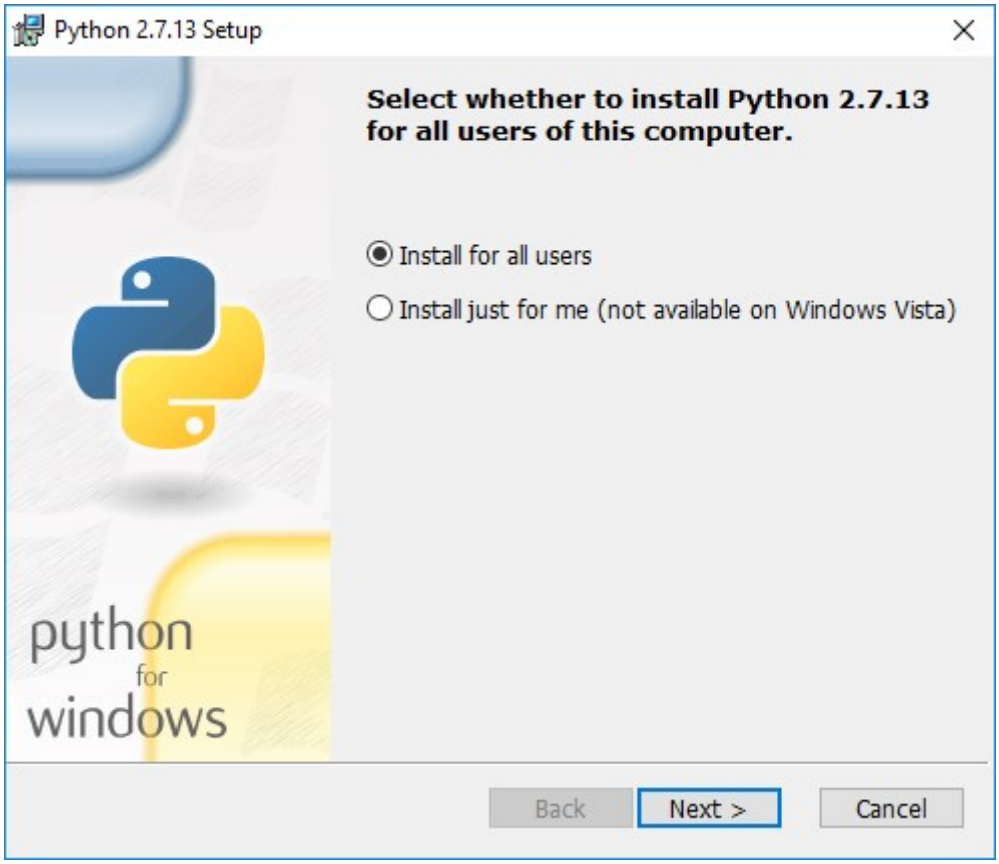
Python installation for windows

If you don't already have a copy of Python installed on your computer, you will need to open up your Internet browser and go to the Python download page (https://www.python.org/downloads/).

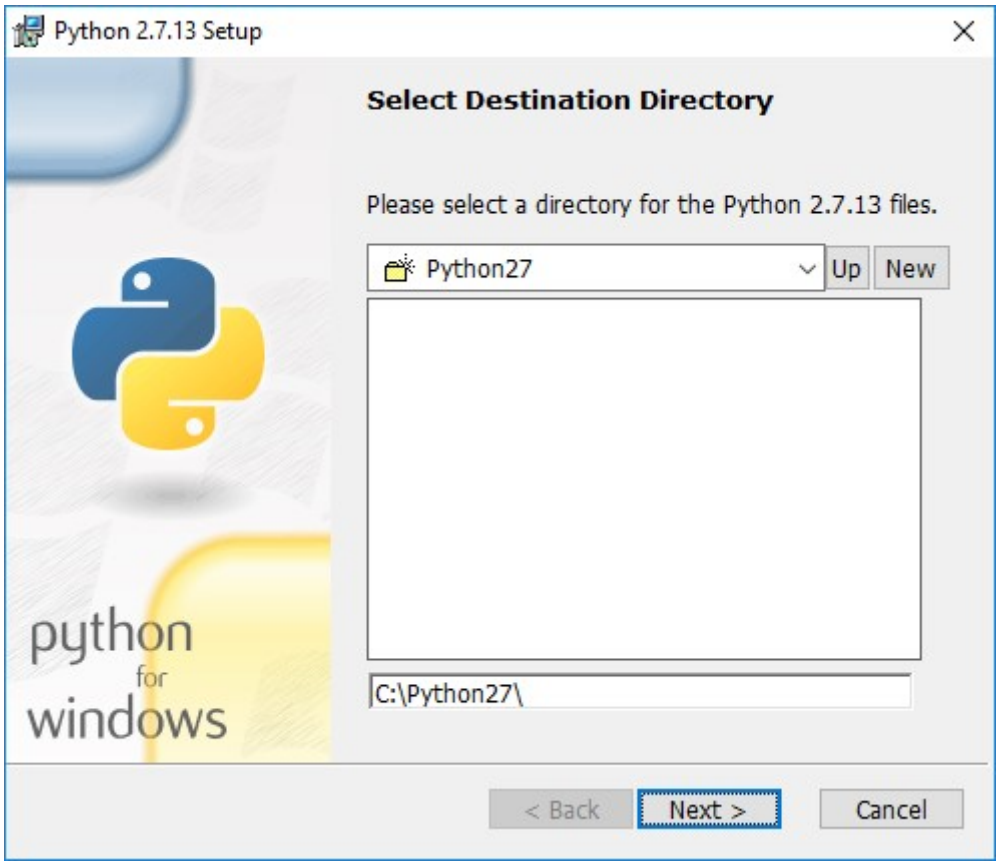


Now that you are on the download page, select which of the software builds you would like to download. For the purposes of this workshop we will use the most up to date 2.x version available (Python 2.7.13).

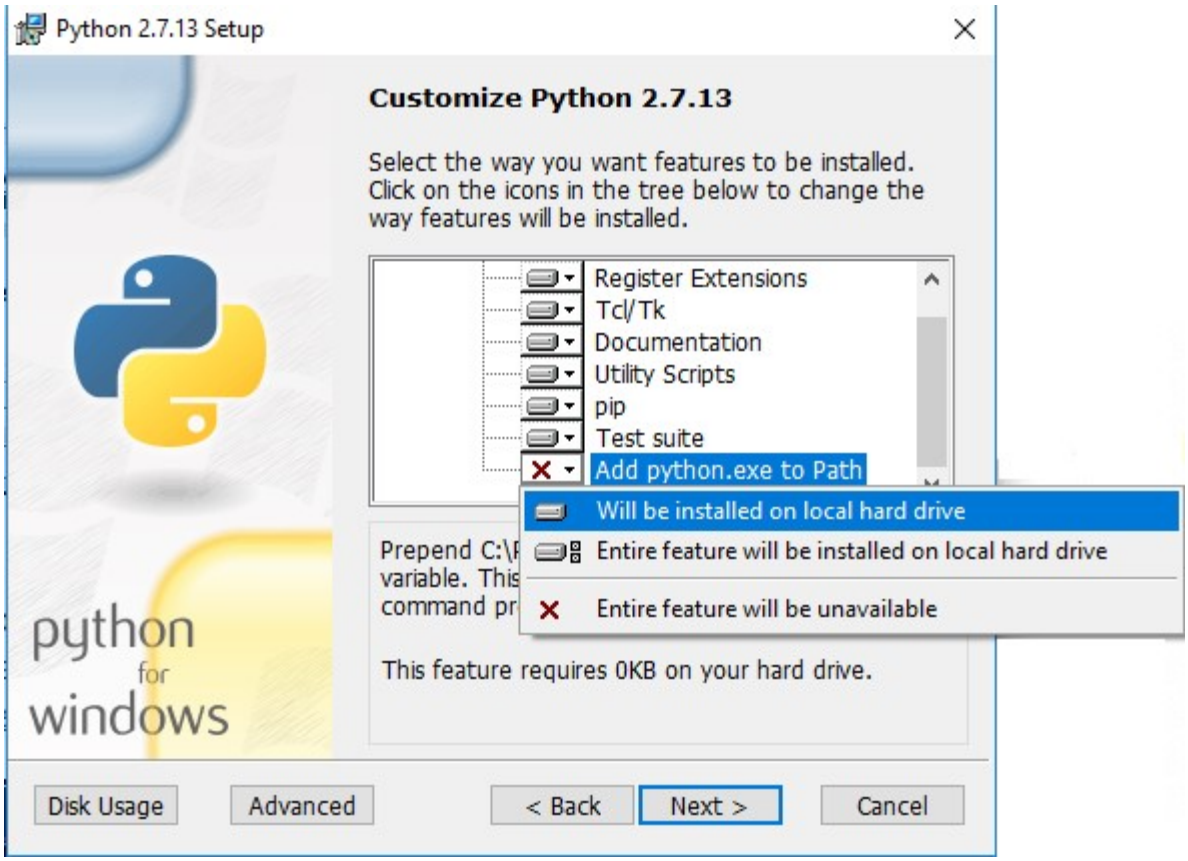
Once you have downloaded the Python MSI, simply navigate to the download location on your computer, double clicking the file and pressing Run when the dialog box pops up. If you are the only person who uses your computer, simply leave the “Install for all users” option selected. If you have multiple accounts on your PC and don’t want to install it across all accounts, select the “Install just for me” option then press “Next.”



If you want to change the install location, feel free to do so; however, it is best to leave it as is and simply select next.

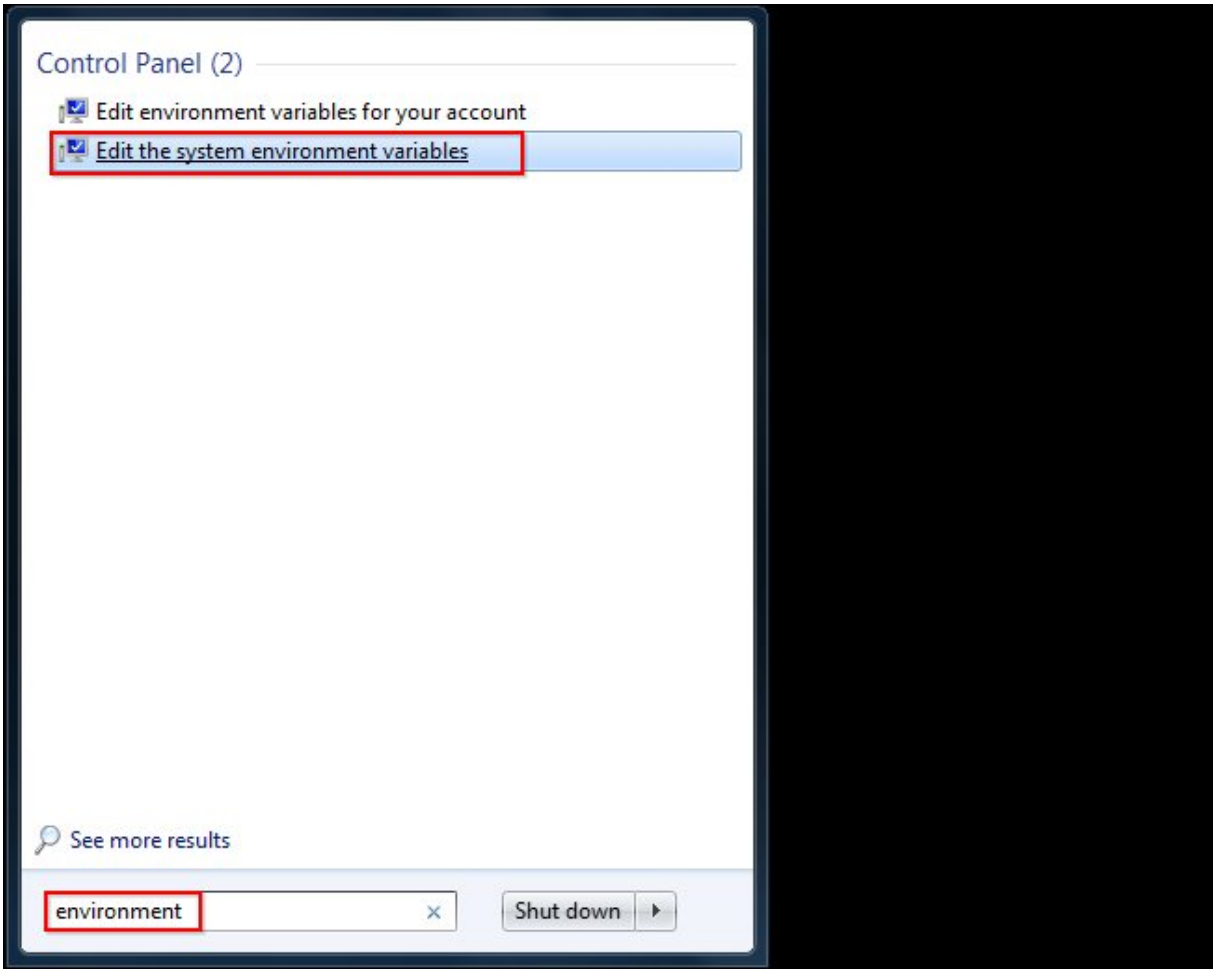


Scroll down in the window and find the “Add Python.exe to Path” and click on the small red “x.” Choose the “Will be installed on local hard drive” option then press “Next.”

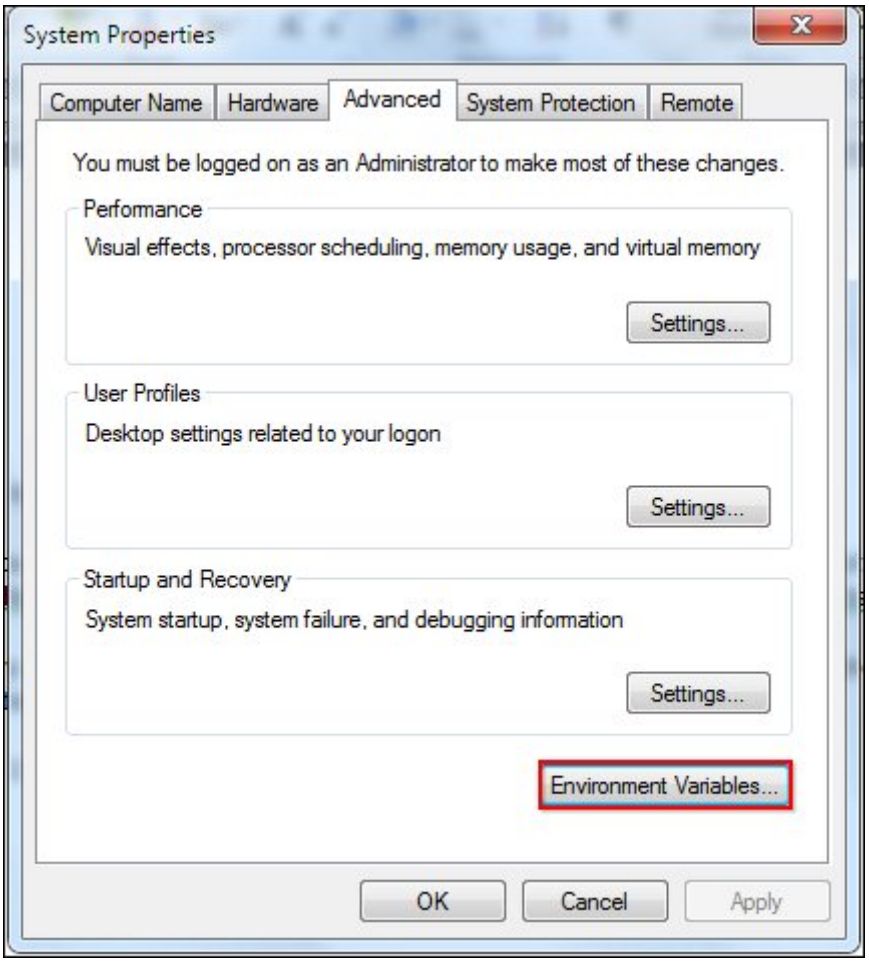


Now that you have completed the installation process, click on “Finish.”

Once you have successfully installed Python, it is time to add it to the System Path Variable. This will allow Python to run scripts on your computer without any conflicts of problems. Begin by opening the start menu and typing in “environment” and select the option called “Edit the system environment variables.”



When the “System Properties” window appears, click on “Environment Variables…”



Once you have the "Environment Variables" window open, direct your focus to the bottom half. You will notice that it controls all the "System Variables" rather than just this associated with your user. Add the following "***C:\Python27\;C:\Python27\Scripts;***" to 'PATH' variable.

Now that we have successfully completed the installation process and added our "Environment Variable," you are ready to check is python is correctly installed and works fine. Let's begin by opening a command line window and typing "python". You have now activated the command line for python, to exit just type the "exit()".

Pip module

pip is a package management system used to install and manage software packages written in Python. Many packages can be found in the Python Package Index (PyPI).

Python 2.7.9 and later (on the python2 series), and Python 3.4 and later include pip (pip3 for Python 3) by default. pip is a recursive acronym that can stand for either "Pip Installs Packages" or "Pip Installs Python". In case where pip is not installed, securely download get-pip.py (https://bootstrap.pypa.io/get-pip.py). And then run the following:

```
python get-pip.py
```

Paho-mqtt module

This module contains the source code for the Eclipse Paho MQTT Python client library, which implements versions 3.1 and 3.1.1 of the MQTT protocol.

This code provides a client class which enable applications to connect to an MQTT broker to publish messages, and to subscribe to topics and receive published messages. It also provides some helper functions to make publishing one off messages to an MQTT server very straightforward.

It supports Python 2.7 or 3.x, with limited support for Python 2.6.

The MQTT protocol is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. Designed as an extremely lightweight publish/subscribe messaging transport, it is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.

Paho is an Eclipse Foundation project.

The latest stable version is available in the Python Package Index (PyPi) and can be installed using:

```
pip install paho-mqtt
```

Raspberry Pi

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside of its target market for uses such as robotics. Peripherals (including keyboards, mice and cases) are not included with the Raspberry Pi. Some accessories however have been included in several official and unofficial bundles.

According to the Raspberry Pi Foundation, over 5 million Raspberry Pis have been sold before February 2015, making it the best-selling British computer. By November 2016 they had sold 11 million units.

Tutorial

Hardware

The following hardware is needed for this tutorial:

- Raspberry Pi connected to the Internet, either via ethernet (Model B) or via WiFi using a compatible WiFi USB dongle,
- Led,
- 10K Ohm resistor,
- MicroSD card (Class 10),
- Power supply,

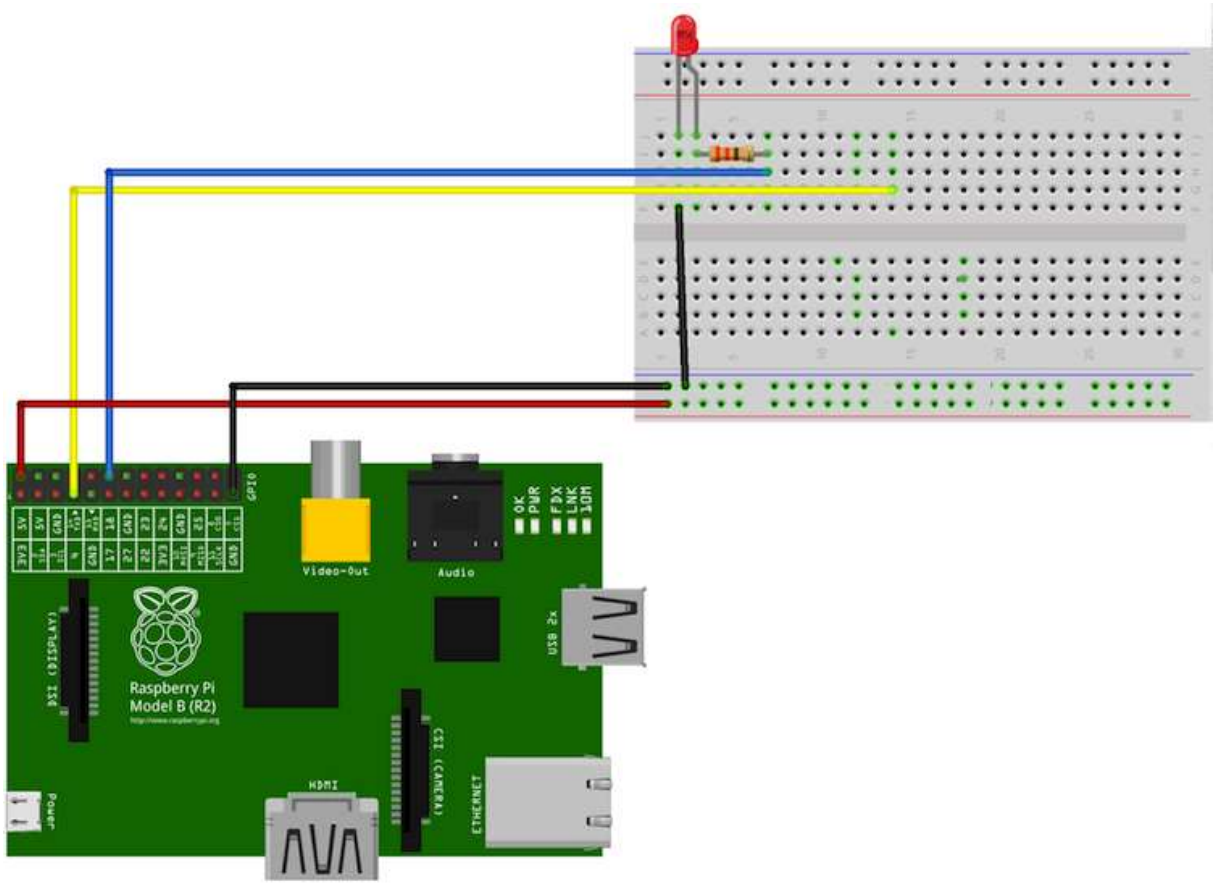
- Breadboard,
- Male/male jumper wires,
- Pin cobbler,
- Ribbon cable.

Software

The Raspberry Pi is programmed using a compatible GNU/Linux distribution. This software can be installed from the raw images ready for download from the official Raspberry Pi website (https://www.raspberrypi.org/downloads/), or by compiling a custom Linux image from scratch using a generation tool such as Buildroot (https://buildroot.uclibc.org/).

Wiring schematic

This schema represents the led and the resistor setup.



Code

Create a new file named “light.py” using a classic text editor.

Imports

In order to use the Paho module and command the Raspberry Pi GPIO (for led control), these modules are imported:

```
import paho.mqtt.client as paho
import RPi.GPIO as GPIO
```

Global variables

Some variables shall be defined to let the Raspberry Pi connect to MQTT broker:

```
# device topics
base_topic = '<your_name_goes_here>/light/'
command_topic = base_topic + 'command'
status_topic = base_topic + 'status'
command_on = "on"
command_off = "off"

# device and broker names
random_client_id = '<my_id>'       # set a random client_id (max 23 char)
broker_address = "iot.eclipse.org"

# Led Pin number
ledPin = 7
```

GPIO setup

Set GPIO mode to BOARD and set pin mode for the led pin. This will set the pin numbering to correspond to the wiring schematic:

```
# Led init
def LedInit():
  GPIO.setmode(GPIO.BOARD)  # use P1 header pin numbering convention
  GPIO.setup(ledPin, GPIO.OUT) # LED pin set as output
  GPIO.output(ledPin, GPIO.LOW)

# Led power
def LedPower(led_on):
  if led_on:
    GPIO.output(ledPin, GPIO.HIGH)
  else:
    GPIO.output(ledPin, GPIO.LOW)
```

MQTT callbacks

When the Raspberry Pi is connected to the remote MQTT broker, subscription to specific topics is performed through the on_connect function.

When a user sends a device command using any MQTT client, a message is published to the topic <YOUR-NAME>/light/command and it's received from the Raspberry Pi through the on_message function.

It is also possible to add further treatment for each topic subscription using the on_subscribe function.

```
# connection event callback
def on_connect(client, data, flags, rc):
  print('Connected, rc: ' + str(rc))
  client.subscribe(command_topic, 0)

# subscription event callabck
def on_subscribe(client, userdata, mid, gqos):
  print('Subscribed: ' + str(mid))

# received message event callback
def on_message(client, obj, msg):
  print "message received: topic = {}, payload = {}".format(msg.topic,
msg.payload)
  # parse received message
  topic = msg.topic
  in_payload = msg.payload
  # check message topic
  if topic == command_topic:
    # check the command property value
    if in_payload == command_on:
      print "turn on room light"
      LedPower(True)
    elif in_payload == command_off:
      print "turn off room light"
      LedPower(False)
    else:
      print "unhandled command: {}".format(in_payload)
    # confirm changes to broker
    if (in_payload == command_on) or (in_payload == command_off):
      out_payload = in_payload
      print "send back status: {}".format(out_payload)
      client.publish(status_topic, out_payload)
  else:
    print "unhandled topic: {}".format(topic)
```

Main function

```
# ------------------------------------------------------------- #
# Main function                                                 #
# ------------------------------------------------------------- #
if __name__ == "__main__":
  # LED init
  LedInit()

  # create the MQTT client
  client = paho.Client(client_id=random_client_id, protocol=paho.MQTTv31)

  # assign event callbacks
  client.on_message = on_message
  client.on_connect = on_connect
  client.on_subscribe = on_subscribe

  # client connection
  client.connect(broker_address)

  # Continue the network loop, exit when an error occurs
  client.loop_forever()
```