

## Chapitre 9

# Documenter et tester ses scripts en Python

## 1. Introduction

L'écriture de tests unitaires est désormais incontournable dans l'élaboration d'un programme informatique. Dans ce domaine, Python est livré avec des modules de choix qui répondent aux attentes des développeurs les plus ambitieux. Python offre aussi la possibilité d'inspecter son code interactivement avec le REPL, et de vérifier instantanément le contenu d'un objet, son type et les méthodes qu'il offre. Pour assister le développeur dans les tâches liées à la documentation, à l'analyse de performance et à la résolution de problèmes en rapport avec le code, la gamme des modules proposés par le langage est assez vaste. Par exemple, lorsque la taille d'un projet devient critique, l'usage de tests unitaires permet d'implémenter plus rapidement de nouvelles fonctionnalités, et de détecter les régressions de code dès le début de l'implémentation. Ce qui fait ainsi gagner du temps et de la productivité. La recherche de performance va aider le développeur à identifier les fonctions gourmandes en exécution de celles qui le sont moins, afin de procéder à de la refactorisation et/ou de la réécriture de code. Dans le cas de l'écriture de scripts destinés au Raspberry Pi où les ressources sont restreintes, auditer et benchmarker son code peut améliorer le temps d'exécution d'un programme. Encore une fois, même si ce livre n'a pas pour but d'enseigner la conduite de projet, sachez qu'il est important de connaître les outils que propose Python dans ce domaine afin de mieux apprécier l'écosystème dans son ensemble.

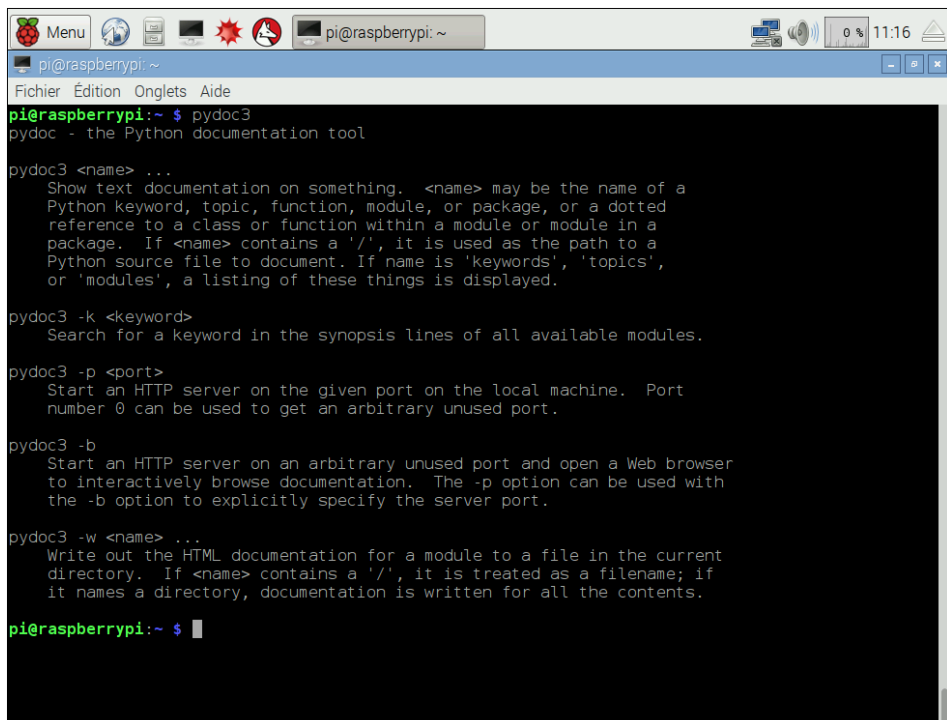
## 2. Consulter de la documentation avec pydoc3

Tout d'abord, l'outil le plus courant qui permet de chercher dans la documentation des modules porte le nom de `pydoc3`. L'utilisation de `pydoc3` intervient lorsque l'on souhaite afficher la documentation d'un module, d'une classe ou d'un mot-clé. Par exemple, la documentation de tous les mots-clés passés en revue dans les chapitres précédents (`with`, `def`, `lambda`, etc.) peut être consultée avec `pydoc3`. Point important à noter avant d'aller plus loin : la majorité de la documentation installée avec le système par défaut de Python est écrite dans la langue anglaise.

Cet outil s'utilise exclusivement en ligne de commande. Ouvrez une console en cliquant sur **Menu - Accessoires - LXTerminal**, comme expliqué au chapitre Raspberry Pi 3, premier contact. Tapez ensuite la commande :

```
pi@raspberrypi:~ $ pydoc3
```

Ce qui devrait afficher le résultat suivant :



```

pi@raspberrypi:~ $ pydoc3
pydoc - the Python documentation tool

pydoc3 <name> ...
  Show text documentation on something. <name> may be the name of a
  Python keyword, topic, function, module, or package, or a dotted
  reference to a class or function within a module or module in a
  package. If <name> contains a '/', it is used as the path to a
  Python source file to document. If name is 'keywords', 'topics',
  or 'modules', a listing of these things is displayed.

pydoc3 -k <keyword>
  Search for a keyword in the synopsis lines of all available modules.

pydoc3 -p <port>
  Start an HTTP server on the given port on the local machine. Port
  number 0 can be used to get an arbitrary unused port.

pydoc3 -b
  Start an HTTP server on an arbitrary unused port and open a Web browser
  to interactively browse documentation. The -p option can be used with
  the -b option to explicitly specify the server port.

pydoc3 -w <name> ...
  Write out the HTML documentation for a module to a file in the current
  directory. If <name> contains a '/', it is treated as a filename; if
  it names a directory, documentation is written for all the contents.

pi@raspberrypi:~ $

```

Cela n'est rien d'autre que l'aide de `pydoc3`. Attention cependant car chaque version de Python installée sur le système est livrée avec sa propre version de `pydoc`. Comme expliqué au chapitre Raspberry Pi 3, premier contact, différentes versions de l'interpréteur Python doivent cohabiter sur le Raspberry Pi. Il en est de même pour `pydoc` qui est livré dans ses versions 2.7 et 3.4. Par défaut, le binaire `pydoc` pointe vers `/usr/bin/pydoc2.7`. Pensez donc à toujours utiliser `pydoc3` pour lire la documentation en rapport avec la version 3 de Python.

`pydoc3` permet donc de chercher et d'afficher la documentation de nombreux *topics* ou sujets. Le sujet recherché correspond au terme passé en paramètre de la commande. Dans cet exemple, le terme `with` est recherché et affiche la documentation qui lui est associée :

```

pi@raspberrypi: ~
Fichier Édition Onglets Aide
The "with" statement
*****

The "with" statement is used to wrap the execution of a block with
methods defined by a context manager (see section *With Statement
Context Managers*). This allows common "try"... "except"... "finally"
usage patterns to be encapsulated for convenient reuse.

    with_stmt ::= "with" with_item ("," with_item)* ":" suite
    with_item ::= expression ["as" target]

The execution of the "with" statement with one "item" proceeds as
follows:

1. The context expression (the expression given in the "with_item")
   is evaluated to obtain a context manager.
2. The context manager's "__exit__()" is loaded for later use.
3. The context manager's "__enter__()" method is invoked.
4. If a target was included in the "with" statement, the return
   value from "__enter__()" is assigned to it.

Note: The "with" statement guarantees that if the "__enter__()"
method returns without an error, then "__exit__()" will always be
called. Thus, if an error occurs during the assignment to the
target list, it will be treated the same as an error occurring
within the suite would be. See step 6 below.

5. The suite is executed.

:

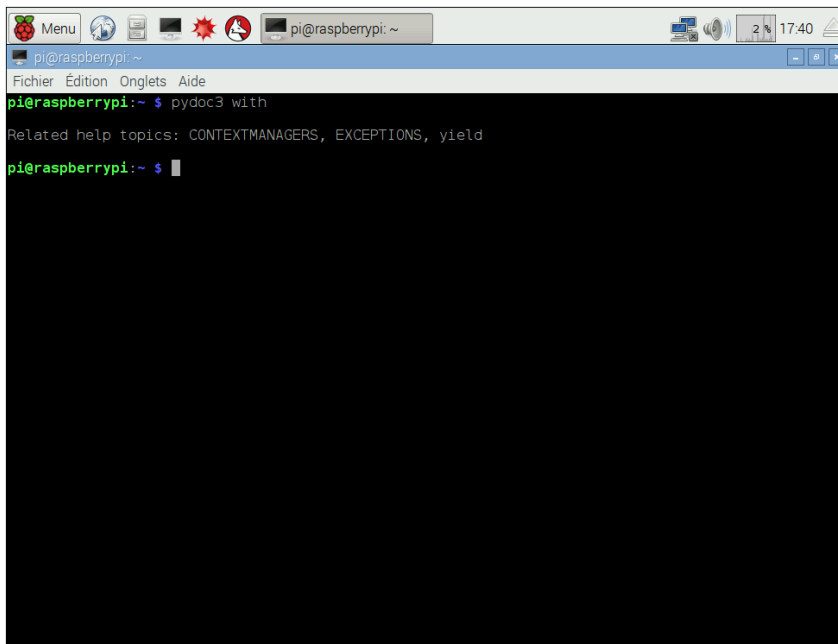
```

## 212 Python et Raspberry Pi - Apprenez à développer sur votre nano-ordinateur

Par défaut, la lecture de la documentation repose sur l'utilisation d'un programme nommé `less`, qui est le pager par défaut de la plupart des distributions Linux actuelles. La navigation dans `less` peut être difficile pour les néophytes qui débutent avec la ligne de commande. Voici un tableau rassemblant les commandes de base :

Défiler vers le haut d'un caractère	[Flèche en haut]
Défiler vers le bas d'un caractère	[Flèche en bas]
Défiler vers le haut d'une page	[Page Up]
Défiler vers le bas d'une page	[Page Down]
Rechercher un terme	Barre oblique (slash) / suivi du terme
Prochaine occurrence du terme	Lettre en minuscule n
Précédente occurrence du terme	Lettre en majuscule N
Ferme (quitter) la documentation	Lettre en minuscule q ou en majuscule Q

Naviguer dans `less` est relativement facile lorsque ces quelques raccourcis sont assimilés. Une fois `less` fermé, `pydoc3` suggère des sujets annexes qui seraient susceptibles d'intéresser le développeur :



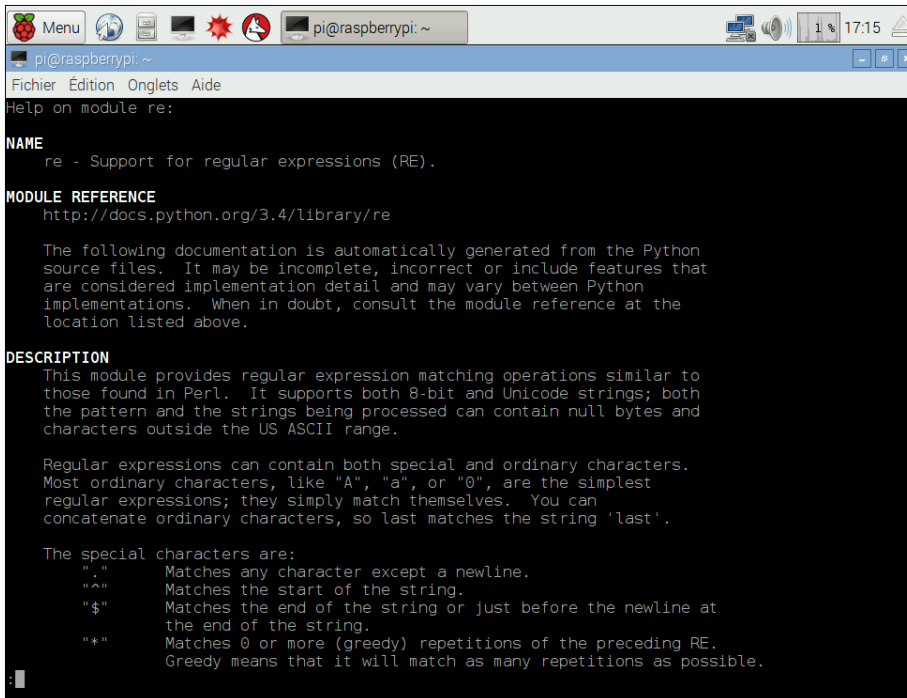
```
pi@raspberrypi: ~
Fichier Edition Onglets Aide
pi@raspberrypi:~ $ pydoc3 with
Related help topics: CONTEXTMANAGERS, EXCEPTIONS, yield
pi@raspberrypi:~ $
```

En plus de chercher de la documentation pour pratiquement n'importe quel sujet touchant au langage, `pydoc3` recherche aussi la documentation associée aux modules de la bibliothèque standard ainsi que les modules installés avec `pip3`, si l'auteur du module en question a évidemment pris le soin de l'écrire.

Pour y parvenir, il suffit de passer en paramètre de `pydoc3` le nom du module dont vous souhaitez consulter la documentation :

```
pi@raspberrypi:~ $ pydoc3 re
```

Ce qui a pour effet d'afficher la documentation associée au module `re` :



```

pi@raspberrypi: ~
Fichier  Edition  Onglets  Aide
Help on module re:

NAME
  re - Support for regular expressions (RE).

MODULE REFERENCE
  http://docs.python.org/3.4/library/re

  The following documentation is automatically generated from the Python
  source files. It may be incomplete, incorrect or include features that
  are considered implementation detail and may vary between Python
  implementations. When in doubt, consult the module reference at the
  location listed above.

DESCRIPTION
  This module provides regular expression matching operations similar to
  those found in Perl. It supports both 8-bit and Unicode strings; both
  the pattern and the strings being processed can contain null bytes and
  characters outside the US ASCII range.

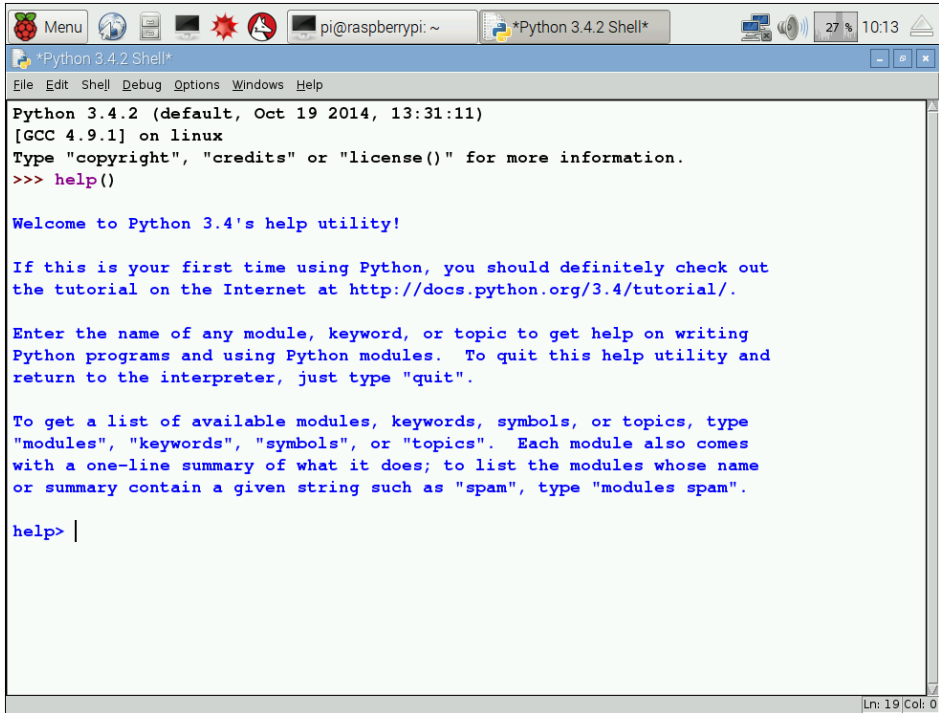
  Regular expressions can contain both special and ordinary characters.
  Most ordinary characters, like "A", "a", or "0", are the simplest
  regular expressions; they simply match themselves. You can
  concatenate ordinary characters, so last matches the string 'last'.

  The special characters are:
  "." Matches any character except a newline.
  "^" Matches the start of the string.
  "$" Matches the end of the string or just before the newline at
  the end of the string.
  "*" Matches 0 or more (greedy) repetitions of the preceding RE.
  Greedy means that it will match as many repetitions as possible.
  
```

`pydoc3` est aussi disponible depuis le REPL d'IDLE grâce à la fonction `help()`. En effet, cette fonction fait indirectement appel à `pydoc3` pour la recherche de documentation. `help()` s'utilise de deux manières : la première consiste à appeler la fonction une fois dans le REPL en ne donnant aucun paramètre. L'utilisateur bascule alors dans une seconde console interactive spécialisée dans la recherche de documentation.

## 214 Python et Raspberry Pi - Apprenez à développer sur votre nano-ordinateur

Tapez un terme et laissez la console effectuer la recherche afin de présenter l'aide associée au terme :



```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> help()

Welcome to Python 3.4's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.4/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> |
```

La deuxième consiste à passer en paramètre un objet. `help()` va alors analyser l'objet en question et afficher la documentation associée. Dans ce cas de figure, vous ne pouvez pas passer autre chose que des objets déjà instanciés ou importés dans le contexte courant.

Vous pouvez aussi afficher la documentation d'une fonction en particulier provenant d'un module :

```
>>> help(re)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 're' is not defined
>>> import re
>>> help(re)
(affiche la documentation du module re)
>>> s = str
>>> help(s)
(affiche la documentation de la classe str)
>>> help(s.replace)
(affiche la documentation de la fonction replace de la classe str)
```

## Chapitre 4

# Les objets ESP8266

## 1. Informations pratiques

### 1.1 Prérequis et configurations

Ce chapitre se concentre sur le développement des objets IoT et nécessite quelques prérequis abordés dans les précédents chapitres avant de se lancer dans l'aventure.

1. Les ESP8266 sont reflashés avec MicroPython pour ESP8266 (version 1.9.1 minimum, elle apporte le support de `asyncio` utilisé dans les objets), cf. ESP8266 sous MicroPython – Charger le firmware MicroPython.
2. La copie de scripts Python sur l'ESP8266 à l'aide d'un utilitaire comme RShell (ou équivalent) est un point maîtrisé, cf. ESP8266 sous MicroPython – Prise de contrôle.
3. La mise en place du fichier `boot.py` avec authentification sur le réseau Wi-Fi domestique ainsi que la fonctionnalité RunApp, cf. ESP8266 sous MicroPython – Séquence de démarrage MicroPython.
4. Le broker MQTT Eclipse Mosquitto est installé sur le Raspberry Pi et configuré avec une **authentification avec le login `pusr103` et le mot de passe `21052017`**, cf. Le broker MQTT – Installation de Mosquitto, cf. Le broker MQTT – Configurer le login du broker MQTT.
- 5- Le Raspberry Pi exécutant le broker MQTT est configuré avec l'adresse IP fixe **192.168.1.210** dans le cadre de cet ouvrage. Toute modification d'adresse IP du Raspberry Pi implique une modification des scripts Python avant de les téléverser sur les objets.

### 1.2 LED de statut

Un élément important de tout projet est la possibilité d'informer l'utilisateur sur son état de fonctionnement. La LED #0 disponible sur le feather ESP8266 est utilisée pour indiquer ce statut.



Utilisation de la LED #0 comme LED de statut

La LED de statut utilise plusieurs motifs de clignotement pour informer l'utilisateur sur le fonctionnement interne de la plateforme.

Motif de la LED	Description
Éteinte	À l'arrêt. Vérifier interrupteur RunApp (ou le fichier <code>boot.py</code> ) puis presser Reset pour redémarrer. À noter que la LED est également éteinte après l'arrêt de l'objet.
Allumée (après démarrage)	Début d'exécution (dans le script <code>main.py</code> , juste après le test RunApp).
Heartbeat	Allumée fixe avec une extinction de 200 ms toutes les 10 secondes. En cours de fonctionnement.
Erreur	Successions de clignotements rapides entrecoupés de séquences de clignotements lents. Le nombre de clignotements lents (de 1 à 6) indique un code d'erreur. En cas d'erreur, l'ESP8266 est redémarré automatiquement ( <code>machine.reset()</code> ) après une heure. Voir détail des codes d'erreurs ci-dessous.
Erreur 1	MQTTClient retourne un code d'erreur, code renvoyé par le broker MQTT.



Motif de la LED	Description
Erreur 2	Erreur lors de la connexion MQTT. Vérifier les constantes MQTT_SERVER, MQTT_USE, MQTT_PSWD. Le contenu du message d'exception est renvoyé sur la console REPL.
Erreur 3	Erreur durant le chargement des bibliothèques senseurs. Vérifier la présence des différentes bibliothèques nécessaires et le fait qu'elles se chargent en mémoire sans erreur (par ex. en chargeant la bibliothèque dans une session REPL). Le contenu du message d'exception est renvoyé sur la console REPL.
Erreur 4	Erreur durant la création des objets destinés à la lecture des différents senseurs. Le contenu du message d'exception est renvoyé sur la console REPL.
Erreur 5	Erreur durant la publication de l'adresse MAC sur le topic connect/<clientID> lors du démarrage de l'objet. Le contenu du message d'exception est renvoyé sur la console REPL.
Erreur 6	Erreur durant le traitement des tâches (capture de données sur les senseurs et publication MQTT). Le contenu du message d'exception est renvoyé sur la console REPL.

### 1.3 Les topics MQTT

Les topics MQTT et publications des informations sont détaillés dans le chapitre de la mise en place du broker MQTT, cf. Le broker MQTT - Topics du projet.

## 1.4 Télécharger et préparer le code des objets IoT

Le code source des différents objets IoT est disponible sur le dépôt GitHub du projet : <https://github.com/mchobby/la-maison-pythonic>

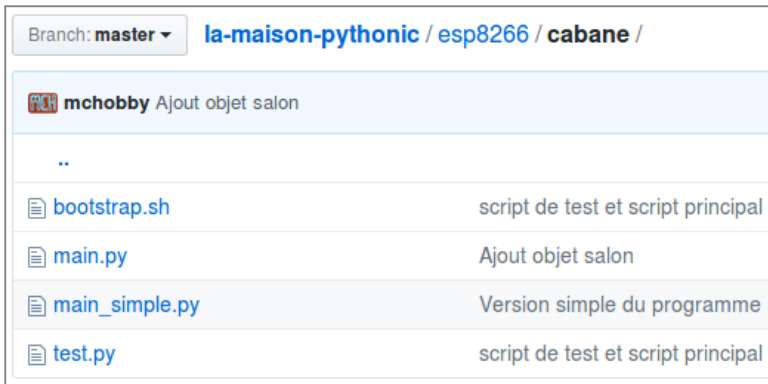
Une copie des sources est également disponible dans les ressources de cet ouvrage.

Les sources peuvent être facilement téléchargées sur le Raspberry Pi à l'aide de l'utilitaire `git`.

```
cd ~  
git clone https://github.com/mchobby/la-maison-pythonic.git
```

Un nouveau répertoire `la-maison-pythonic` est disponible dans le répertoire utilisateur (`/home/pi`). Ce dernier contient les sources des différents objets dans le sous-répertoire `esp8266`. Le répertoire `esp8266` contient lui-même un sous-répertoire par objet.

Par exemple, le code destiné à la cabane est disponible dans le répertoire `/home/pi/la-maison-pythonic/esp8266/cabane/`.



Code source de l'objet Cabane tel que disponible sur GitHub

Le script principal est bien entendu `main.py`, mais d'autres versions sont également disponibles comme `test.py` permettant de tester les senseurs et `main_simple.py` proposant une version intermédiaire, mais simplifiée, du code.

### Bootstrap.sh

Le fichier `bootstrap.sh` permet de télécharger les dépendances (les bibliothèques Python) de l'objet cabane. Les bibliothèques des senseurs sont publiées sur un autre dépôt GitHub, le fichier `bootstrap.sh` fait le nécessaire pour les rapatrier dans le répertoire courant. Ces fichiers devront aussi être copiés sur l'ESP8266.

```

Branch: master ▾ la-maison-pythonic / esp8266 / cabane / bootstrap.sh
mchobby script de test et script principal
1 contributor

Executable File | 11 lines (8 sloc) | 350 Bytes
1  #!/bin/bash
2  # Collecter les pilotes nécessaires
3  rm ts12561.py
4  rm bme280.py
5  rm am2315.py
6  wget https://raw.githubusercontent.com/mchobby/esp8266-upy/master/ts12561/ts12561.py
7  wget https://raw.githubusercontent.com/mchobby/esp8266-upy/master/bme280-bmp280/bme280.py
8  wget https://raw.githubusercontent.com/mchobby/esp8266-upy/master/am2315/am2315.py
9
10

```

Contenu du fichier bootstrap de l'objet cabane

Pour télécharger les dépendances :

1. Ouvrir un terminal.
2. Se placer dans le répertoire cabane.
3. Exécuter le script bootstrap.sh.

Si le dépôt GitHub a été cloné dans le répertoire utilisateur alors les dépendances (bibliothèques) peuvent être téléchargées comme suit :

```

cd ~/la-maison-pythonic/esp8266/cabane/
./bootstrap.sh

```

```

pi@pythonic:~ $ cd ~/la-maison-pythonic/esp8266/cabane/
pi@pythonic:~/la-maison-pythonic/esp8266/cabane $ ./bootstrap.sh
rm: cannot remove 'tsl2561.py': No such file or directory
rm: cannot remove 'bme280.py': No such file or directory
rm: cannot remove 'am2315.py': No such file or directory
--2017-12-31 17:15:59-- https://raw.githubusercontent.com/mchobby/esp8266-upy/master/tsl2561/tsl2561.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.36.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.36.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6997 (6.8K) [text/plain]
Saving to: 'tsl2561.py'

tsl2561.py          100%[=====] 6.83K  --.-KB/s  in 0.001s

2017-12-31 17:15:59 (10.2 MB/s) - 'tsl2561.py' saved [6997/6997]

--2017-12-31 17:15:59-- https://raw.githubusercontent.com/mchobby/esp8266-upy/master/bme280-bmp280/bme280.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.36.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.36.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8409 (8.2K) [text/plain]
Saving to: 'bme280.py'

```

Téléchargement des bibliothèques

## 2. Fonctionnement général d'un objet IoT

Tous les scripts des différents objets développés dans ce chapitre suivent une même structure de code.

Le code ci-dessous reprend la structure générale d'un objet.

```

01: # coding: utf8
02: """ La Maison Pythonic - Object Cabane v0.2
03:
04:     Envoi des données toutes les heures + 30 minutes
05:     vers serveur MQTT
06:     """
07:
08: from machine import Pin, I2C, reset
09: from time import sleep, time
10: from ubinascii import hexlify
11: from network import WLAN
12:
13: CLIENT_ID = 'cabane'
14: MQTT_SERVER = "192.168.1.210"
15:
16: # Mettre à None si pas utile
17: MQTT_USER = 'pusr103'
18: MQTT_PSWD = '21052017'
19:
20: # redémarrage auto après erreur
21: ERROR_REBOOT_TIME = 3600 # 1 h = 3600 sec
22:
23: # --- Démarrage conditionnel ---
24: runapp = Pin( 12, Pin.IN, Pin.PULL_UP )

```