eBook Gratuit

APPRENEZ robotframework

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#robotframe work

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: robotframework

It is an unofficial and free robotframework ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official robotframework.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec robotframework

Remarques

Cette section fournit une vue d'ensemble de ce qu'est robotframework et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les sujets importants dans le cadre du robot, et établir des liens avec les sujets connexes. La documentation pour robotframework étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
Framework Robot 3.0.2	2017-02-14
Framework Robot 3.0.1	2017-01-06
Cadre de robot 3.0	2015-12-31
Cadre de robot 2.9.2	2015-10-09
Cadre de robot 2.9.1	2015-08-28
Cadre du robot 2.9	2015-07-30

Examples

Installation ou configuration

Instructions détaillées sur la configuration ou l'installation de Robot Framework.

Framework de robot est un framework d'automatisation de test générique. Il est implémenté à l'aide de Python et est pris en charge sur Python 2 et Python 3 Jython (JVM) et IronPython (.NET) et PyPy. Pour

1. Test d'acceptation
2. Développement sous test d'acceptation (ATDD)

Conditions préalables

1. Installer un interprète
2. Configuration de PATH

3. Définition de https_proxy

Python a les implémentations les plus avancées et il est suggéré d'utiliser Python si vous n'avez pas d'exigences exceptionnelles.

Version du cadre de robot	Version interpréteur prise en charge
Cadre de robot 3.0	Python 2.6
Cadre de robot 3.0	Python 2.7
Cadre de robot 3.0	Python 3.3
Cadre de robot 3.0	Jython 2.7 et Java 7
Cadre de robot 3.0	IronPython 2.7
Cadre de robot 2.5-2.8	Python 2.5
Cadre de robot 2.5-2.8	Jython 2.5
Framework Robot 2.0-2.1	Python 2.3
Framework Robot 2.0-2.1	Python 2.4
Framework Robot 2.0-2.1	Jython 2.2

Installation de Python

La version souhaitée de python peut être téléchargée depuis https://www.python.org/downloads/

Installation de Jython

Un installateur est disponible sur http://jython.org . Vous pouvez exécuter ce package JAR exécutable à partir de la ligne de commande comme javaava -jar jython_installer-.jar.

Installation d'IronPython

Un programme d'installation est disponible à l'adresse http://ironpython.net/download/ pour IronPython 2.7.Lorsque vous utilisez IronPython, une autre dépendance est l'installation du module elementtree 1.2.7.

Configuration de PATH & Setting https_proxy

Ajouter le répertoire d'installation de Python (par défaut C: \ Python27, C: \ Python27 \ Scripts, C: \ jython2.7.0 \ bin etc sur windows) et le répertoire Scripts au début de votre variable de chemin

La valeur de https_proxy devrait être l'URL du proxy. Ceci est requis lorsque ces packages sont installés avec pip et que vous êtes dans un réseau proxy

Installation de Robot Framework avec pip

Installer la dernière version de robotframework

```
pip install robotframework
```

Installer une version spécifique

```
pip install robotframework==2.0
```

Installation de Robot Framework à partir de la source

La distribution source de Robot Framework est disponible à l' adresse https://code.google.com/archive/p/robotframework/downloads.Robot Framework est installée à partir de la source en utilisant le script standard setup.py de Python dans le répertoire des scripts source.

```
python setup.py install
jython setup.py install
ipy setup.py install
```

Installation de Robot Framework 3.0 sur une machine Windows à l'aide de Python 2.7.11

Ceci est un guide rapide pour faire fonctionner Robot Framework 3.0 sur une machine Windows en utilisant Python 2.7.11 - Il ne va pas trop loin sur le pourquoi et le comment, il vous permet simplement de démarrer. Les premières choses sont les premières, n'allons pas installer Python!

1. Téléchargez Python 2.7.11 pour Windows . (Programme d'installation MSI Windows x86-64 ou programme d'installation MSI Windows x86 selon l'architecture)

2. Exécutez l'installation, en vous assurant d'installer "pip" et que vous activez l'option "Ajouter python.exe au chemin" (vous devrez peut-être redémarrer votre ordinateur pour tirer parti du Python PATH. Dans ce guide, vous n'a pas ce luxe)

3. Une fois qu'il est installé, faisons une vérification rapide pour vous assurer qu'il est installé correctement. Exécutez CMD en tant qu'administrateur et naviguez jusqu'à l'emplacement d'installation de Python sur `cd C:\Python27` et tapez `python -V` . Il devrait retourner "Python 2.7.11"

C'est ça, Python est maintenant installé sur votre machine. La partie suivante consiste à installer Robot Framework sur votre machine à l'aide de pip.

1. Tout d'abord, assurons-nous d'avoir la dernière version de pip, en naviguant d'abord dans le

répertoire scripts de Python `cd C:\Python27\Scripts` , puis en entrant `python -m pip install -U pip` . Il faut dire que vous avez la version la plus à jour installée!

2. Ensuite, installons Robot Framework en saisissant `pip install robotframework`

3. Une fois que Pip a fini de télécharger et d'installer les fichiers, entrez `robot --version` pour vous assurer qu'il est correctement installé. Il devrait dire Robot Framework 3.0 (Python 2.7.11 sur win32 / 64)

4. (Facultatif) Si, à l'avenir, il existe une mise à jour pour Robot Framework, vous pouvez exécuter cette commande `pip install --upgrade robotframework`

Lire Démarrer avec robotframework en ligne:
https://riptutorial.com/fr/robotframework/topic/5187/demarrer-avec-robotframework

Chapitre 2: Comment le framework de robot est-il utilisé dans les tests d'automatisation dans les systèmes embarqués?

Introduction

Le framework de robot est largement utilisé dans les tests d'automatisation des produits embarqués. Nous allons prendre un produit Embedded comme exemple et voir comment automatiser les scénarios de test à l'aide de Robot Framework.

Remarques

Abréviation:

- RPS - Alimentation à distance
- RF - Travail sur cadre de robot

Examples

Test de l'alimentation à distance

Simulation d'alimentation à distance

Comme nous n'avons pas de véritable matériel d'alimentation à distance, nous allons le simuler à l'aide du programme python.

Idée de base sur les RPS

- En réalité, l'alimentation à distance est dotée d'un serveur http.
- L'utilisateur peut envoyer des commandes pour activer / désactiver l'alimentation à l'aide de la requête http.

Nous allons simuler une alimentation à distance en utilisant le programme suivant rps-server.py.

```
from flask import Flask, request
from flask_httpauth import HTTPBasicAuth


app = Flask(__name__)
auth = HTTPBasicAuth()

users = {
        'admin': '12345678'
}
```

```
app.url_map.strict_slashes = False

PINS = ['P60', 'P61', 'P62', 'P63']

PINS_STATUS = {'P60':'0', 'P61': '0', 'P62':'0', 'P63':'0'}

@auth.get_password
def get_pw(username):
        if username in users:
                return users.get(username)
        return None

@app.route('/')
@auth.login_required
def index():
        return "Hello, %s!" % auth.username()

def get_html_string():
        html_str = '<html>P60={}P61={}P62={}P63={}</html>'.format(PINS_STATUS['P60'],
                                                PINS_STATUS['P61'],
                                            PINS_STATUS['P62'],
                                            PINS_STATUS['P63'])
        return html_str

def parse_cmd_args(args):
        global current_status
        if str(args['CMD']) == 'SetPower':
                for key in args:
                        if key in PINS:
                                PINS_STATUS[key] = str(args[key])

                return get_html_string()

        if str(args['CMD']) == 'GetPower':
                return get_html_string()


    @app.route('/SetCmd', methods=['GET','POST'])
    def rps():
        if request.method=="GET":
                args=request.args.to_dict()
                ret = parse_cmd_args(args)
                return ret
```

Le code ci-dessus simule en fait le serveur http pour contrôler l'alimentation à distance.

Comment exécuter le serveur RPS?

```
$ export FLASK_APP=rps-server.py
$ flask run
```

Comment envoyer des commandes au serveur rps?

Voici les deux commandes utilisées pour contrôler le RPS

1. SetPower

2. Obtenir du pouvoir

Par défaut, le serveur écoutera au port 5000.

Les ports d'alimentation sont,

1. P60
2. P61
3. P62
4. P64

Les états des ports sont,

1. LE 1
2. OFF - 0

Exigences

Les exigences pour la construction d'une alimentation à distance sont

1. L'alimentation à distance doit pouvoir être activée / désactivée à distance
2. L'état de l'alimentation à distance est accessible à distance.

Dériver des cas de test

Cas de test dérivés d'exigences

1. Allumez l'alimentation 2 à distance.
2. Vérifiez que l'alimentation 2 est activée.
3. Éteignez l'alimentation 2 à distance.
4. Vérifiez que l'alimentation 2 est éteinte.

Test manuel

- Exécutez le serveur rps.
- Pour activer le port 3, ouvrez un navigateur et donnez l'URI suivante

```
http://admin:12345678@localhost:5000/SetCmd?CMD=SetPower&P62=1
```

- Pour obtenir le statut de tous les ports

```
http://admin:12345678@localhost:5000/SetCmd?CMD=GetPower
```

Bibliothèque de test d'écriture

Nous devons écrire une bibliothèque de test en python pour envoyer des commandes http en utilisant la requête http. Plus tard, nous utiliserons cette bibliothèque comme mots-clés dans le

travail sur le cadre du robot.

commands.py

Nous allons utiliser la bibliothèque de commands.py pour envoyer SetPower et GetPower.

```
import requests
import re


class commands(object):

    ROBOT_LIBRARY_SCOPE = 'GLOBAL'
    def __init__(self, ip='localhost:5000'):
        self.ip_address = ip
        self.query = {}
        self.user = 'admin'
        self.passw = '12345678'

    def form_query(self, state, cmd, port):
        port = self.get_port_no(port)
        self.query = {port: state}
        return self.query

    def get_port_no(self, port_no):
        port = 'P6' + str(port_no)
        return port

    def clean_html(self, data):
        exp = re.compile('<.*?>')
        text = re.sub(exp, "", data)
        return text.rstrip()

    def send_cmds(self, cmd, port=None, state=None):
        url = 'http://{}:{}@{}/SetCmd?CMD={}'\
              .format(self.user,
                      self.passw,
                      self.ip_address,
                      cmd)
        print url
        if cmd == 'SetPower':
            self.form_query(state, cmd, port)
            self.req = requests.get(url, params=self.query)
            return True
        elif cmd == 'GetPower':
            self.req = requests.get(url)
            data = self.clean_html(self.req.text)
            return data
        else:
            return False

        return self.req.text


# c = commands('localhost:5000')

# c.send_cmds('SetPower', 2, 1)
# c.send_cmds('SetPower', 3, 1)
# print c.send_cmds('GetPower')
```

Documentation sur les mots clés Python

1. `send_cmds(cmd, port=None, state=None)` est la fonction que nous allons utiliser.
2. Lorsque vous utilisez cette fonction dans le mot-clé Robot, vous n'avez pas besoin de vous soucier de _ , ou de `Lowercaser` ou `Uppercase` dans le nom de la fonction.

La fonction Python ressemblera à ceci en utilisant comme mot-clé,

```
Send Cmds       cmd   port  state
```

Test de rédaction

Nous allons utiliser `Send Cmds` comme mot-clé python dans la suite de tests.

- Les commandes d'envoi RPS utilisent les quatre arguments suivants pour définir la puissance
    - command = SetPower
    - port = 2
    - state = 1 pour ON / 0 pour off Lorsque nous appelons cette commande, il allume / éteint l'alimentation
- RPS get power renvoie le statut de tous les ports d'alimentation

```
*** Keywords ***
RPS send commands
    [Arguments]    ${command}    ${port}    ${state}
    ${output}=    Send cmds    ${command}  ${port}  ${state}
    [return]    ${output}

RPS get Power
    [Arguments]    ${command}
    ${output}=    Send cmds    ${command}
    [return]    ${output}}
```

Algorithme pour tester l'alimentation

1. Mettre le courant sur un port
2. Vérifiez le statut de cmd
3. Obtenez le statut du port et vérifiez s'il est activé / désactivé

Écrire des cas de test en utilisant les mots clés ci-dessus

Nous sommes maintenant prêts à écrire un cas de test en utilisant les deux mots-clés suivants

- Commandes d'envoi RPS - Pour définir et désactiver une alimentation de port
- RPS obtenir le pouvoir - Pour obtenir le statut de tout le port

```
*** Settings ***
Library      commands.py

*** Test Cases ***
Turn on Power supply 2 remotely
     ${out}=    RPS send commands     SetPower  2  1
     Should be equal    ${out}  ${True}

Verify power supply 2 is on
    ${out}=     RPS get power    GetPower
    should contain    ${out}  P62=1

Turn off Power supply 2 remotely
     ${out}=    RPS send commands     SetPower  2  0
     Should be equal    ${out}  ${True}

Verify power supply 2 is off
    ${out}=     RPS get power    GetPower
    should contain    ${out}  P62=0
```

Créez un nom de fichier `remote-power-supply.robot`

Copiez ci-dessus les mots clés et les cas de test dans le fichier.

Comment exécuter le serveur RPS et remote-power-supply.robot?

- Exécutez d'abord l'alimentation à distance
- Exécutez la suite de tests remote-power-supply.robot

```
$ export FLASK_APP=rps-server.py
$ flask run
$ pybot remote-power-supply.robot
```

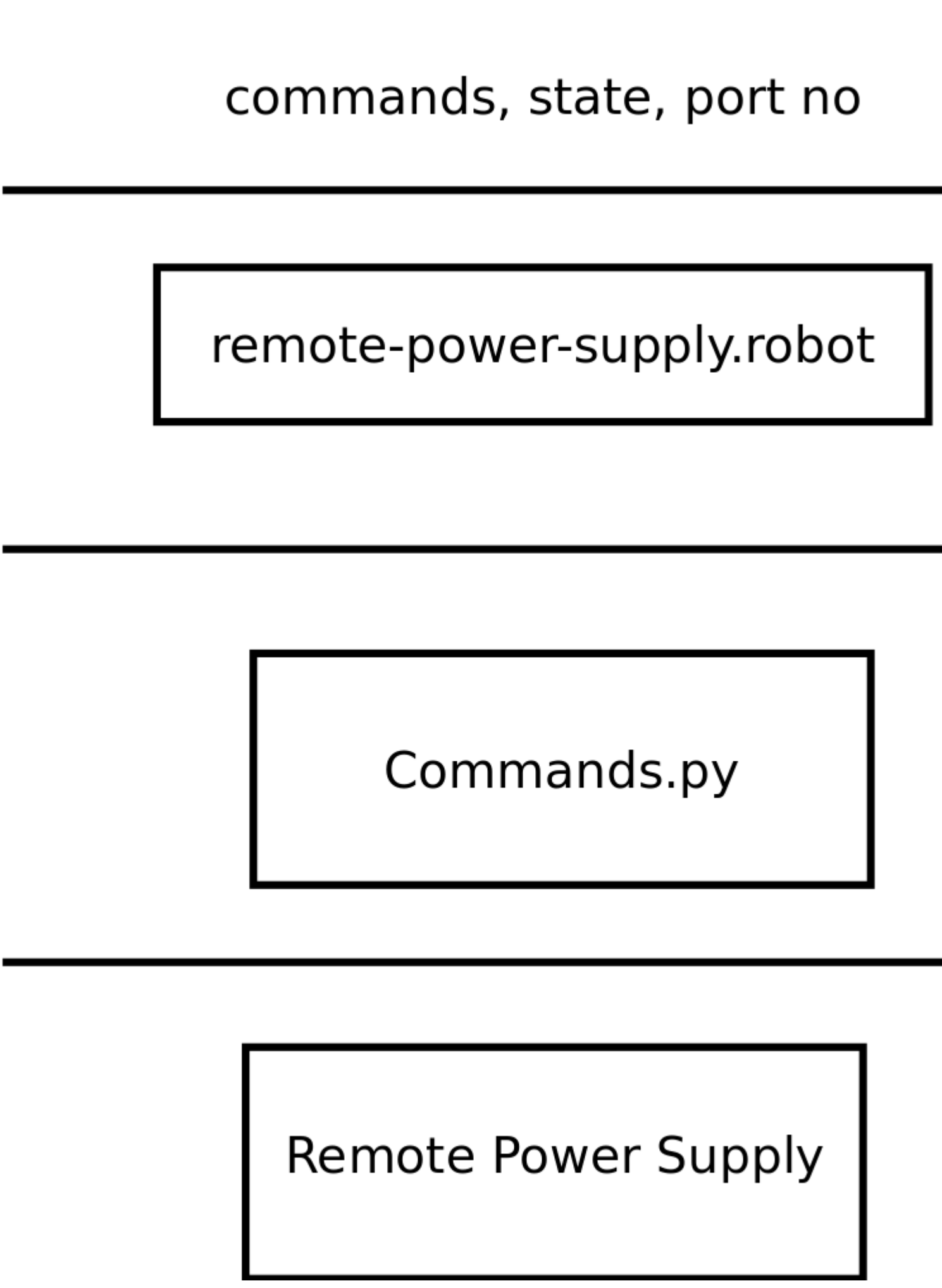
Sortie

```
$ pybot remote-pwer-supply.robot
==============================================================================
Remote-Pwer-Supply
==============================================================================
Turn on Power supply 2 remotely                                       | PASS |
------------------------------------------------------------------------------
Verify power supply 2 is on                                           | PASS |
------------------------------------------------------------------------------
Turn off Power supply 2 remotely                                      | PASS |
------------------------------------------------------------------------------
Verify power supply 2 is off                                          | PASS |
------------------------------------------------------------------------------
Remote-Pwer-Supply                                                    | PASS |
4 critical tests, 4 passed, 0 failed
4 tests total, 4 passed, 0 failed
==============================================================================
Output:  /tmp/talks/robot-framework-intro/test-cases/output.xml
Log:     /tmp/talks/robot-framework-intro/test-cases/log.html
```

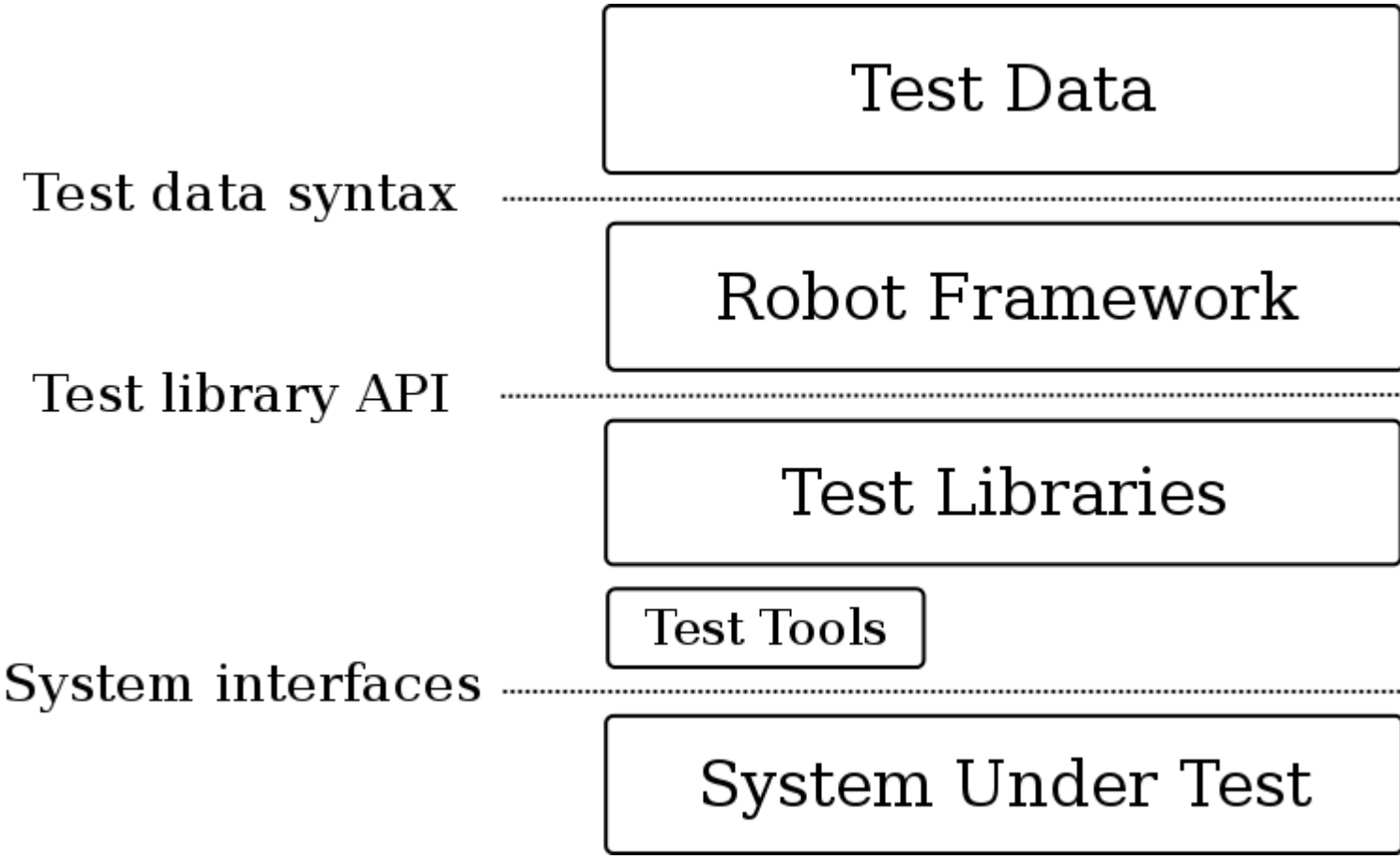
```
Report:  /tmp/talks/robot-framework-intro/test-cases/report.html
```

Les deux diagrammes suivants expliquent l'architecture de test entre RPS et RF

Architecture de test de l'alimentation à distance


commands, state, port no
remote-power-supply.robot
Commands.py
Remote Power Supply

Architecture de travail du cadre robot



Crédits

Merci à la structure du robot pour le diagramme d'architecture.

Le code complet est disponible ici

Alimentation à distance

commands.py

alimentation à distance.robot

Lire Comment le framework de robot est-il utilisé dans les tests d'automatisation dans les systèmes embarqués? en ligne: https://riptutorial.com/fr/robotframework/topic/10672/comment-le-framework-de-robot-est-il-utilise-dans-les-tests-d-automatisation-dans-les-systemes-embarques-

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec robotframework	Community, Goralight, Hariprasad, Shijo
2	Comment le framework de robot est-il utilisé dans les tests d'automatisation dans les systèmes embarqués?	kame, sakthirengaraj