Kostenloses eBook

LERNEN robotframework

Free unaffiliated eBook created from
Stack Overflow contributors.

#robotframework

work

Inhaltsverzeichnis

Über

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

Kapitel 1: Erste Schritte mit robotframework

Bemerkungen

In diesem Abschnitt erhalten Sie einen Überblick über das, was robotframework ist und warum ein Entwickler es verwenden möchte.

Es sollte auch alle großen Themen in robotframework erwähnen und mit den verwandten Themen verknüpfen. Da die Dokumentation für robotframework neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Versionen

Ausführung	Veröffentlichungsdatum
Roboter-Framework 3.0.2	2017-02-14
Robot Framework 3.0.1	2017-01-06
Robot Framework 3.0	2015-12-31
Roboter-Framework 2.9.2	2015-10-09
Roboter-Framework 2.9.1	2015-08-28
Roboter-Framework 2.9	2015-07-30

Examples

Installation oder Setup

Detaillierte Anweisungen zum Einrichten oder Installieren von Robot Framework.

Robot Framework ist ein generisches Testautomatisierungs-Framework. Dieses wird mit Python implementiert und von Python 2 und Python 3 Jython (JVM) und IronPython (.NET) sowie PyPy unterstützt. Zum

1. Abnahmeprüfung
2. Akzeptanztestgesteuerte Entwicklung (ATDD)

Voraussetzungen

1. Installieren Sie einen Dolmetscher
2. PATH konfigurieren

3. Https_proxy einstellen

Python verfügt über die fortschrittlichsten Implementierungen. Es wird empfohlen, Python zu verwenden, wenn Sie keine außergewöhnlichen Anforderungen haben.

Roboter-Framework-Version	Unterstützte Version des Interpreters
Robot Framework 3.0	Python 2.6
Robot Framework 3.0	Python 2.7
Robot Framework 3.0	Python 3.3
Robot Framework 3.0	Jython 2.7 & Java 7
Robot Framework 3.0	IronPython 2.7
Robot Framework 2.5-2.8	Python 2.5
Robot Framework 2.5-2.8	Jython 2.5
Robot Framework 2.0-2.1	Python 2.3
Robot Framework 2.0-2.1	Python 2.4
Robot Framework 2.0-2.1	Jython 2.2

Python-Installation

Die gewünschte Version von Python kann von https://www.python.org/downloads/ heruntergeladen werden.

Jython-Installation

Ein Installationsprogramm kann unter http://jython.org gefunden werden . Sie können dieses ausführbare JAR-Paket über die Befehlszeile wie javaava -jar jython_installer-.jar ausführen.

IronPython-Installation

Ein Installationsprogramm ist unter http://ironpython.net/download/ für IronPython 2.7 zu finden. Wenn Sie IronPython verwenden, wird das Elementbaum- Modul 1.2.7 zusätzlich installiert

Konfigurieren von PATH und Einstellen von https_proxy

Fügen Sie das Python-Installationsverzeichnis (standardmäßig C: \ Python27, C: \ Python27 \ Scripts, C: \ jython2.7.0 \ bin usw. unter Windows) und das Scripts-Verzeichnis am Anfang Ihrer

Pfadvariablen hinzu

Der Wert von https_proxy sollte die URL des Proxy sein. Dies ist erforderlich, wenn diese Pakete mit pip installiert werden und Sie sich in einem Proxy-Netzwerk befinden

Robot Framework mit pip installieren

Installieren Sie die neueste Version von robotframework

```
pip install robotframework
```

Installieren Sie eine bestimmte Version

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

Robot Framework von der Quelle installieren

Die Quelldistribution von Robot Framework finden Sie unter https://code.google.com/archive/p/robotframework/downloads.Robot Framework wird vom Quellcode aus mit dem Standard-Skript setup.py von Python im Quellenskriptverzeichnis installiert

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

Installieren von Robot Framework 3.0 auf einem Windows-Computer mit Python 2.7.11

Dies ist eine Kurzanleitung, um Robot Framework 3.0 auf einem Windows-Computer mit Python 2.7.11 zum Laufen zu bringen - Das Warum und Wie wird nicht allzu detailliert beschrieben, es bringt Sie einfach zum Laufen. Die ersten Dinge sind zuerst, lasst uns nicht gehen und Python installieren!

1. Laden Sie Python 2.7.11 für Windows herunter. (Windows x86-64 MSI-Installationsprogramm oder Windows x86 MSI-Installationsprogramm je nach Architektur)

2. Führen Sie die Installation durch und stellen Sie sicher, dass Sie "pip" installieren und "Python.exe zum Pfad hinzufügen" wählen diesen Luxus nicht haben)

3. Nach der Installation überprüfen wir kurz, ob die Installation korrekt ist. Führen Sie CMD als Administrator aus und navigieren Sie zu dem Ort, an dem Python installiert wurde, um `cd C:\Python27` und geben Sie `python -V` . Es sollte "Python 2.7.11" zurückgeben

Python ist jetzt auf Ihrem Rechner installiert. Der nächste Teil ist das Installieren des Roboter-Frameworks mithilfe von pip auf Ihrer Maschine.

1. `cd C:\Python27\Scripts` wir zunächst sicher, dass wir über die neueste Version von pip

verfügen, indem Sie zunächst zum Verzeichnis scripts in Python `cd C:\Python27\Scripts` und dann `python -m pip install -U pip` . Es sollte sagen, dass Sie die aktuellste Version installiert haben!

2. Als Nächstes können Sie Robot Framework installieren, indem Sie `pip install robotframework`

3. Nachdem pip die Dateien heruntergeladen und installiert hat, geben Sie `robot --version` , um sicherzustellen, dass sie ordnungsgemäß installiert wurde. Es sollte Robot Framework 3.0 sagen (Python 2.7.11 unter win32 / 64)

4. (Optional) Wenn es in Zukunft ein Update für Robot Framework gibt, können Sie diesen Befehl `pip install --upgrade robotframework`

Erste Schritte mit robotframework online lesen:
https://riptutorial.com/de/robotframework/topic/5187/erste-schritte-mit-robotframework

Kapitel 2: Wie wird das Roboter-Framework bei Automatisierungstests in eingebetteten Systemen verwendet

Einführung

Roboter-Frameworks werden häufig bei Automatisierungsprüfungen für eingebettete Produkte eingesetzt. Wir werden ein Embedded-Produkt als Beispiel nehmen und sehen, wie man die Testfälle mit Robot Framework automatisiert.

Bemerkungen

Abkürzung:

- RPS - Fernspeisung
- RF - Roboterrahmenarbeit

Examples

Test der Fernspeisung

Simulation der Fernspeisung

Da wir keine echte Hardware für die Fernspeisung haben, werden wir sie mit dem Python-Programm simulieren.

Grundidee zu RPS

- Eigentlich hat die Fernspeisung einen http-Server.
- Der Benutzer kann Befehle zum Ein- und Ausschalten der Stromversorgung über die http-Anfrage senden.

Wir simulieren die Fernspeisung mit dem folgenden Programm 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
```

Der obige Code simuliert tatsächlich einen http-Server zur Steuerung der Remote-Stromversorgung.

Wie führe ich einen RPS-Server aus?

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

Wie sende ich Befehle an den RPS-Server?

Nachfolgend sind die beiden Befehle zur Steuerung des RPS aufgeführt

1. SetPower
2. Macht erlangen

Standardmäßig überwacht der Server den Port 5000.

Die Stromversorgungsanschlüsse sind

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

Die Zustände der Ports sind

1. AUF 1
2. AUS - 0

Bedarf

Voraussetzungen für den Aufbau einer Fernspeisung sind

1. Die Fernspeisung sollte aus der Ferne ein- und ausgeschaltet werden können
2. Auf den Status der Fernspeisung kann remote zugegriffen werden.

Testfälle ableiten

Von der Anforderung abgeleitete Testfälle

1. Schalten Sie die Stromversorgung 2 aus der Ferne ein.
2. Stellen Sie sicher, dass das Netzteil 2 eingeschaltet ist.
3. Schalten Sie die Stromversorgung 2 aus der Ferne aus.
4. Stellen Sie sicher, dass das Netzteil 2 ausgeschaltet ist.

Manuelle Prüfung

- Führen Sie den RPS-Server aus.
- Um Port 3 zu aktivieren, öffnen Sie einen Browser und geben Sie den folgenden URI an

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

- Um den Status aller Ports zu erhalten

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

Testbibliothek schreiben

Wir müssen eine Testbibliothek in Python schreiben, um http-Befehle mit der http-Anfrage zu

senden. Später werden wir diese Bibliothek als Schlüsselwörter in der Roboterarbeit verwenden.

Befehle.py

Wir werden die Bibliothek von befehl.py verwenden, um SetPower und GetPower zu senden.

```
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')
```

Python-Schlüsselwortdokumentation

1. `send_cmds(cmd, port=None, state=None)` ist die Funktion, die wir verwenden werden.
2. Während Wort mit dieser Funktion in Roboter - Taste, keine Notwendigkeit , die Mühe über _ oder `Lowercaser` oder `Uppercase` in Funktionsnamen.

Die Python-Funktion sieht so aus, wenn sie als Schlüsselwort verwendet wird.

```
Send Cmds       cmd   port  state
```

Test-Schlüsselwörter schreiben

Wir werden in der `Send Cmds` als Python-Schlüsselwort verwenden.

- RPS-Sendebefehle verwenden die folgenden vier Argumente, um die Leistung einzustellen
    - Befehl = SetPower
    - Port = 2
    - state = 1 für ON / 0 für Off Wenn wir diesen Befehl aufrufen, wird die Stromversorgung ein- / ausgeschaltet
- RPS get power liefert den Status aller Stromanschlüsse

```
*** 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}}
```

Algorithmus zum Testen der Stromversorgung

1. Stellen Sie die Stromversorgung für einen Port ein
2. Überprüfen Sie den Status von cmd
3. Rufen Sie den Status des Ports ab und prüfen Sie, ob er EIN / AUS ist

Schreiben von Testfällen mit den obigen Schlüsselwörtern

Jetzt können wir Testfälle mit den folgenden zwei Schlüsselwörtern schreiben

- RPS-Sendebefehle - Zum Einstellen und Deaktivieren einer Portstärke
- RPS get power - Um den Status aller Ports zu erhalten

```
*** 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
```

Erstellen Sie einen Dateinamen `remote-power-supply.robot`

Kopieren Sie die Schlüsselwörter und den Testfall in die Datei.

Wie führt man RPS-Server und Remote-Power-Supply.robot aus?

- Führen Sie zuerst die Fernspeisung aus
- Führen Sie die Testsuite remote-power-supply.robot aus

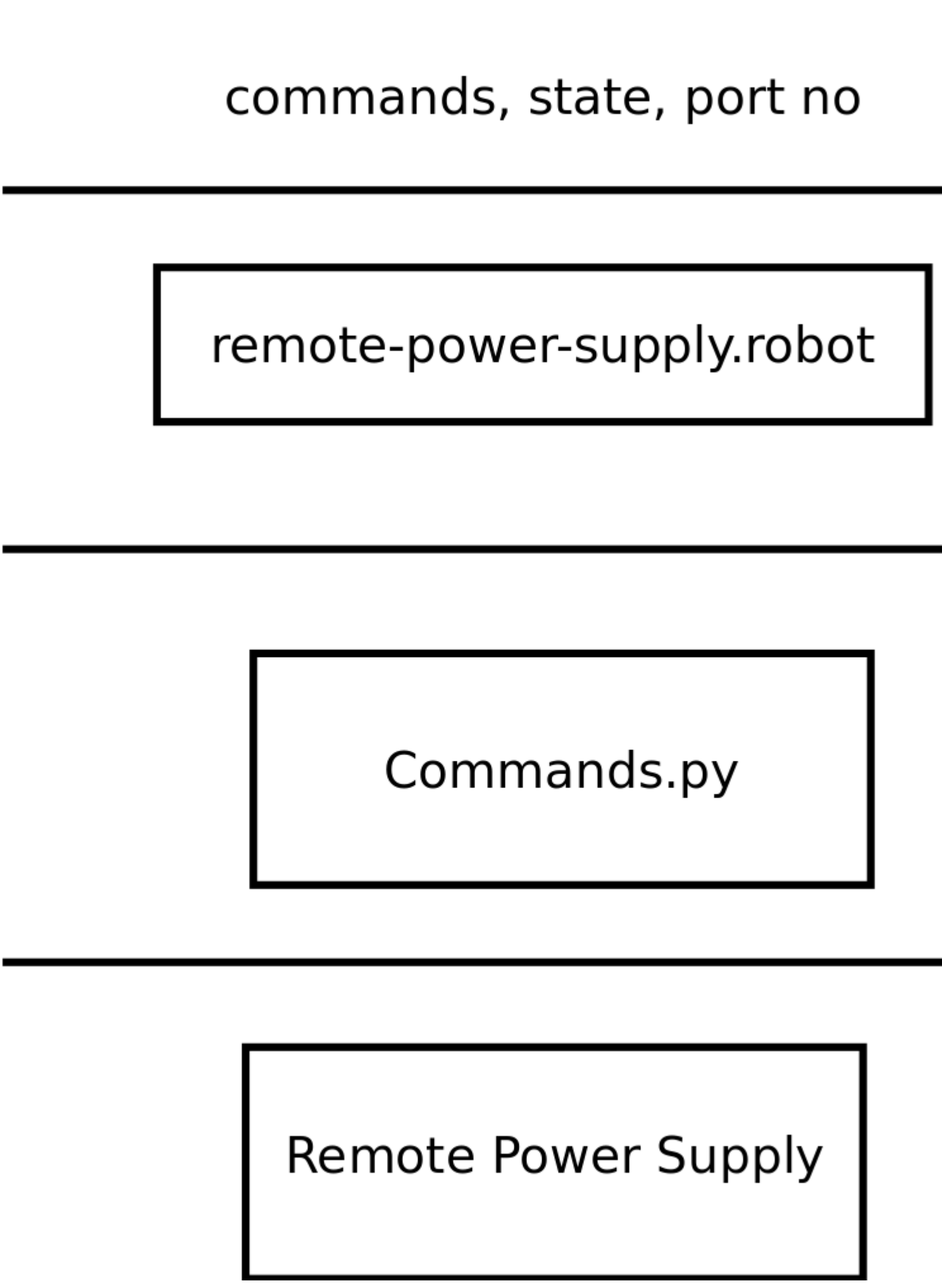
```
$ export FLASK_APP=rps-server.py
$ flask run
$ pybot remote-power-supply.robot
```

Ausgabe

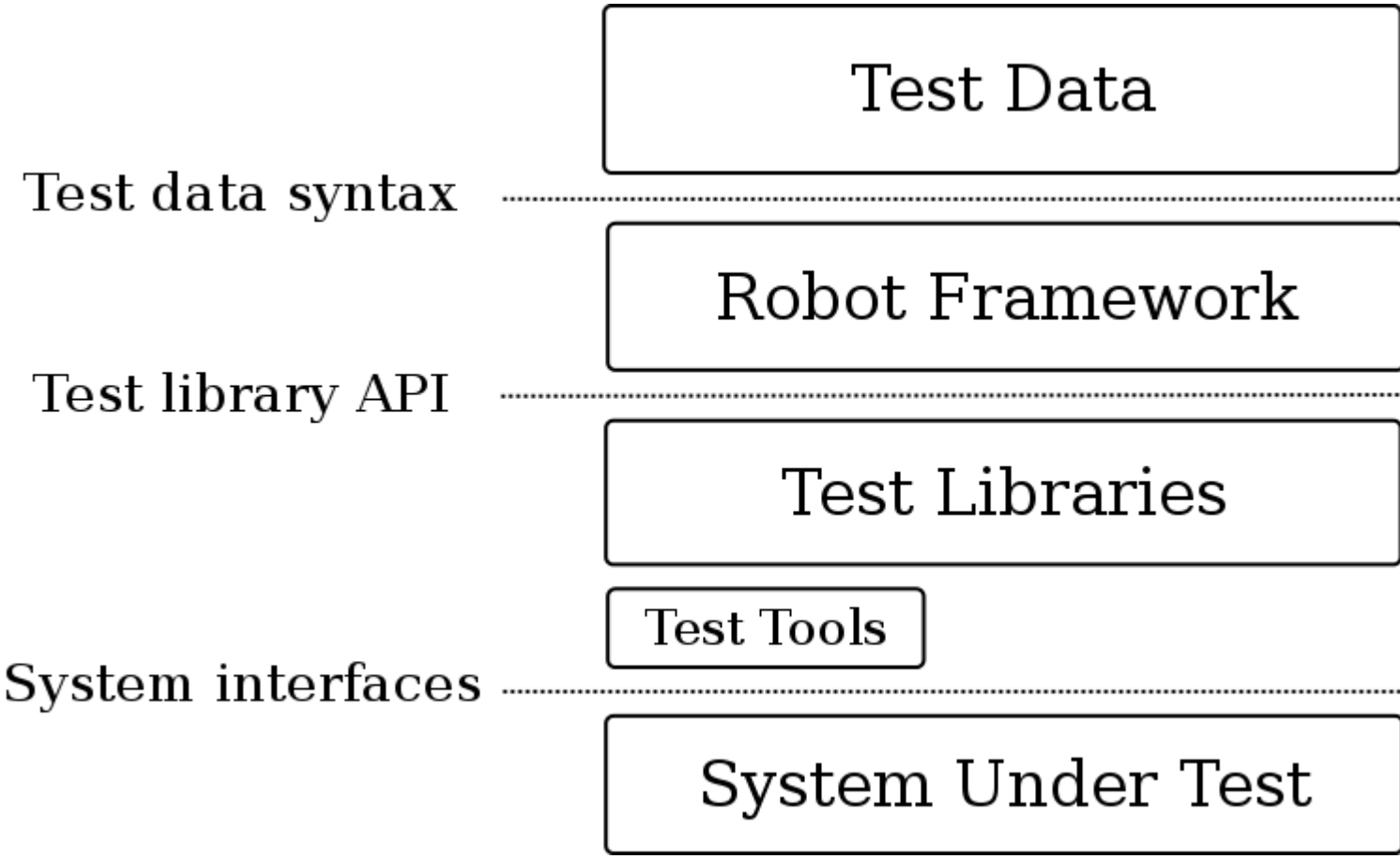
```
$ pybot remote-pwer-supply.robot
==============================================================================
Remote-Pwer-Supply
==============================================================================
Turn on Power supply 2 remotely                                       | PASS |
------------------------------------------------------------------------------
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
```

Die folgenden zwei Diagramme erläutern die Testarchitektur zwischen RPS und RF

Testarchitektur für Remote-Stromversorgung


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

Roboter-Rahmenarbeitsarchitektur



Credits

Dank Roboter-Framework für Architekturdiagramm.

Den vollständigen Code finden Sie hier

Fernspeisung

Befehle.py

Fernspeisung.Roboter

Wie wird das Roboter-Framework bei Automatisierungstests in eingebetteten Systemen verwendet online lesen: https://riptutorial.com/de/robotframework/topic/10672/wie-wird-das-roboter-framework-bei-automatisierungstests-in-eingebetteten-systemen-verwendet

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit robotframework	Community, Goralight, Hariprasad, Shijo
2	Wie wird das Roboter-Framework bei Automatisierungstests in eingebetteten Systemen verwendet	kame, sakthirengaraj