

ラボのスペースとコストを節約 Raspberry Pi 3 Model Bでシリアルと イーサネットのテスト機器を制御する

APPLICATION NOTE



KEITHLEY
A Tektronix Company

Tektronix[®]

テストラボの設立と運営にかかるコストの一部は、ルーチンや手順を自動化するために必要なコンピュータリソースのコストです。シングルステーションであれ、マルチステーションであれ、高品質のテスト機器はコンピュータよりも良い投資となります。コンピュータのコストは、試験装置のコストに比べればわずかなものであると言えますが、コンピュータに費やす数百ドルは、試験装置に機能やオプションを追加することに充てた方がよいかもしれません。こうしたことから、オンラインで多くのユーザーコミュニティが存在し、多くの事例がある低コストの代替品を活用することは検討に値するでしょう。

Raspberry Piは、クレジットカードサイズのシングルボードコンピュータで、ARM互換の統合CPUとオンチップGPUを搭載しています。Raspberry Piは、その小型サイズ、低消費電力、適応性の高さから、ロボット工学、データロギング、その他の電子機器プロジェクトなど、さまざまな用途で人気を博しています。Raspberry Piの推奨ディストリビューションはRaspbianと呼ばれる、DebianベースのLinuxオペレーティングシステムです。Raspbianの最新版であるStretchには、さまざまなソフトウェア開発ツールが搭載されており、Pythonをはじめとする多くのプログラミング言語に対応しています。

Pythonは読みやすく、使いやすいプログラミング言語として知られています。このドキュメントでは、Raspberry Pi 3 Model Bを使った遠隔計測とデータ収集のための最適な方法を詳しく説明しています。

このドキュメントでは、Raspberry Piの一般的なセットアップと、テスト自動化のための推奨ソフトウェアツールのインストールについて説明しています。 **ケースレー DAQ6510 データロガー・データ収集マルチメータシステム**のサンプルコードでは、PyVISAの実装と、オペレータがラボのテストツールと通信する方法を示しています。また、ソケットベースの通信（Ethernet/LAN対応の計測器用）の例では、運用に必要なソフトウェアプラグインの数を最小限に抑えています。

Raspberry Pi セットアップとコンフィグレーション

Raspberry Piを使用するには、HDMI対応のモニター、またはHDMIアダプターケーブルを使用した別のモニタータイプ、5Vで2 Aを供給できるマイクロUSB電源、そして標準的なキーボードとマウスが必要です。Piの電源を入れるには、電源に接続するだけで自動的に起動します。

Installing Raspbian

Raspberry Pi Foundationでは、NOOBS（New Out of Box Software）がプリインストールされた、クラス4の8 GBマイクロSDカードの使用を推奨しています。NOOBSは、Raspberry Pi用のOSインストールマネージャーです。

事前に設定されたRaspberry Pi用のSDカードを使用しない場合は、まず、8GB以上の容量を持つ空のマイクロSDカードを入手します。8GB以上のSDカードの場合、カードをFATで再フォーマットする必要があります。選んだSDカードの記憶容量が32GB以上（例えば64GB以上）の場合は、FAT32に再フォーマットする必要があります。

カードのフォーマットが完了したら、NOOBSをカードにインストールします。NOOBSのZIPファイルは、Raspberry Piの公式サイトからダウンロードできます。ダウンロードしたファイルの内容を、フォーマットしたSDカードのルートにコピーします。マイクロSDをRaspberry Piの底面にあるスロットに挿入します

Initial Boot

Piの電源を初めて入れたとき、「インストール」タブで「Raspbian（フルデスクトップ版）」を選択します。フルバージョンにはRaspbianが含まれているので、インターネットに接続しなくてもSDカードから直接インストールすることができます。カード上のOSイメージが古く、新しいバージョンがリリースされている場合は、Raspberry Piがインターネットに接続されると、最新バージョンをダウンロードするオプションが利用できるようになります。

インターネット接続を確立するには、LANポートを介してPiをネットワークに接続するか、Raspberry Pi 3 Model BのオンボードWiFi機能を使用して、NOOBSインストーラーウィンドウのWiFiタブから利用可能なワイヤレスネットワークに接続するだけです。コマンドターミナルからアップデートや新しいPythonパッケージをダウンロードしてインストールするには、インターネット接続が必要です。

Raspbianがインストールされると、デスクトップが表示され、左上のアプリケーションメニューにアクセスできるようになります。アプリケーションメニューには、同梱されているすべてのプログラミングIDE、Raspberry Piの設定オプション、シャットダウンオプションがあります。Raspbianのインストールは、最新版のRaspbianイメージのみをPiにインストールするように更新することができます。オペレーティングシステムを別のRaspbianイメージにアップグレードすることはできません。

アップデートを適用するには、コマンドターミナルを開いて入力します。

```
sudo apt-get update
sudo apt-get dist-upgrade
```

その後、Raspberry Piを再起動します。

セットアップの詳細については、Raspberry Pi Foundationの公式ドキュメントページをご覧ください。

VISA

Raspberry Piは、PyVISAライブラリを使って、Python経由でVISAを介して計測器と通信することができます。PyVISAはVISAのフロントエンドで、複数のバックエンドに接続できるPythonのAPIを提供します。デフォルトのバックエンドは、従来のNational Instruments社のNI-VISAライブラリです。しかし、NI-VISAは、Raspberry PiのRaspbianと互換性がありません。代わりに、PyVISA-pyバックエンドが使われています。PyVISA-pyは、VISAライブラリの純粋なPython実装で、最も一般的な属性とメソッドをサポートしています。

Python2

PyVISAをラズベリーパイにインストールするには、ラズベリーパイのコマンドプロンプトを開いて入力します。

```
pip install -U pyvisa
```

ラズベリーパイにPyVISA-pyをインストールするには、ラズベリーパイのコマンドプロンプトを開き、入力してください。

```
pip install pyvisa-py
```

Python3

ラズベリーパイにPyVISAをインストールするには、ラズベリーパイのコマンドプロンプトを開いて入力します。

```
pip3 install -U pyvisa
```

ラズベリーパイにPyVISA-pyをインストールするには、ラズベリーパイのコマンドプロンプトを開いて、次のように入力します。

```
pip3 install pyvisa-py
```

PyVISA-pyバックエンドは、VISAリソースマネージャのインスタンスを作成する際に、'@py'引数を渡すことで選択できます。この'@py'引数により、PyVISAはデフォルトのNational Instruments社のバックエンドをバイパスすることができます。

以下は、Python3でPyVISA TCP/IPリソース接続を開き、閉じ、検証する基本的な例で、測定器のID文字列を照会し、コンソールウィンドウに表示します。Piと計測器をLANポートを介してイーサネットケーブルで接続します。

TCPIP Example

```
import visa

rm = visa.ResourceManager('@py')
address = "TCPIP0::169.254.153.137::inst0::INSTR"
inst = rm.open_resource(address)

inst.write("*RST")
print(inst.query("*IDN?"))

inst.close()
rm.close()

#Use pyvisa library
#Use the pyvisa-py backend
#Keithley DAQ6510 IP address TCP/IP string

#Sends the reset command to the instrument
#Prints the instrument ID string to the
command line

#Close the connection
```

PyVISAを使用する利点は、PySerialによるRS-232など、イーサネット以外の通信のための様々なインターフェイスを容易にすることができることです。PyVISA-pyのバックエンドの性質上、TCP/IP以外のリソースタイプを使用するためには、追加のPythonライブラリをインストールする必要があります。

Python2

PySerialをラズベリーパイにインストールするには、ラズベリーパイのコマンドプロンプトを開いて入力します。:

```
python -m pip install pyserial
```

使用可能なリソースタイプと、それに依存するライブラリがインストールされているかどうかを確認するには、コマンドプロンプトを開いて次のように入力します。

```
python -m visa info
```

Python3

PySerialをラズベリーパイにインストールするには、ラズベリーパイのコマンドプロンプトを開き、次のように入力します

```
python3 -m pip install pyserial
```

どのようなリソースタイプが利用できるか、またそれらが依存するライブラリがインストールされているかを確認するには、コマンドターミナルを開いて次のように入力します。

```
python3 -m visa info
```

Raspberry Piと測定器の間でシリアル通信を確立するには、USB-シリアル変換ケーブルを使用して、USB側をPiに、シリアルピンを測定器のRS-232ポートに接続します。最良の結果を得るためには、機器のボーレートを9600に設定してください。

Python2

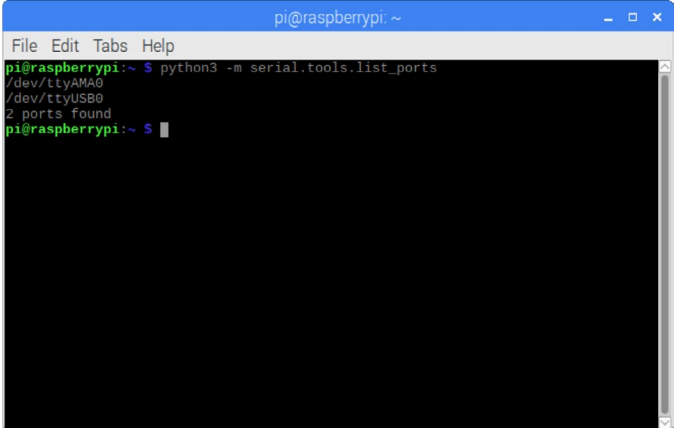
Piに接続されている利用可能なシリアルポートやアダプタを特定するには、Raspberry Piのコマンドプロンプトを開いて入力します。

```
python -m serial.tools.list_ports
```

Python3

Raspberry Piに接続されているシリアルポートやアダプタを特定するには、Raspberry Piのコマンドプロンプトを開いて、次のように入力します。

```
python3 -m serial.tools.list_ports
```



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ python3 -m serial.tools.list_ports  
/dev/ttyAMA0  
/dev/ttyUSB0  
2 ports found  
pi@raspberrypi:~$
```

以下の例では、Python3でPyVISAシリアルRS-232リソース接続による長期温度スキャンをオープン、クローズ、実行する方法を詳細に説明しています。測定器のID文字列とスキャンデータは、.csvファイルに送信される前にコンソールに表示されます

RS-232 Example

```
import visa #Use pyvisa library
import time

rm = visa.ResourceManager('@py') #Use the pyvisa-py backend
address = "ASRL/dev/ttyUSB0::INSTR" #Raspberry Pi USB Serial Port ASRL String
inst = rm.open_resource(address)
inst.write("*RST")
print(inst.query("*IDN?"))

inst.write("FORM:DATA ASCII") #Format scan data into ascii

inst.write("FUNC 'TEMP', (@101, 102)") #Measure Temperature
inst.write("TEMP:TRAN FRTD, (@101, 102)") #Set transducer to four wire RTD
inst.write("TEMP:RTD:FOUR PT3916, (@101, 102)") #Set the RTD type to PT3916
inst.write("TEMP:OCOM ON, (@101, 102)") #Turn on offset compensation
inst.write("TEMP:ODET ON, (@101, 102)") #Turn on open lead detector

inst.write("ROUT:SCAN:COUN:SCAN 25") #Set scan count to 25
inst.write("ROUT:SCAN:INT 300") #Execute a scan every 5 minutes
inst.write("ROUT:SCAN:CRE (@101, 102)") #Scan channels 101 and 102
inst.write("INIT") #Initiate the scan

# monitor for scan completion...

time.sleep(5.0)
trigState = inst.query(':TRIG:STAT?') # query the state of the scan activity which is either
# "running" when the measurements are being made
# or "waiting" when the interval (delay) between scans
# is active

while("RUNNING" in trigState) | ("WAITING" in trigState):
    print("Running...")
    time.sleep(5.0)
    trigState = inst.query(':TRIG:STAT?')

data = inst.query('TRAC:DATA? 1, 50, "defbuffer1", CHAN, READ, UNIT')
print(data) #Print readings to console

inst.close() #Close the connection

#send data to csv file
with open("/home/pi/4Wire_RTD_Temperature.csv", "a") as log:
    log.write(data)

rm.close()
```

For more information, see the official [PyVISA](#), [PyVISA-py](#), and [PySerial](#) documentation.

Ethernet Sockets

Raspberry Piでのソケットプログラミングによる通信は、Python標準ライブラリのソケットモジュールを介して可能です。ソケットプログラミングにより、同じネットワーク上の2つのデバイスが通信できるようになります。遠隔地の計測器の場合、計測器はサーバーとして動作してリスンし、Piはクライアントとしてプログラムされ、接続を形成するためにリーチアウトします。

ソケットプログラミングは、ユーザーが追加のPythonパッケージをインストールする必要がなく、単純なイーサネットネットワーク接続で容易に実現できるため、Raspberry Piで計測器と通信するための最も簡単な方法です。

次の例では、Python3のソケット接続でACパラメータスキャンを開き、閉じ、実行する方法を詳しく説明しています。測定器のID文字列とスキャンデータがコンソールに表示され、その後.csvファイルに送信されます。

Example

```
import socket
import time

TCP_IP = "169.254.153.137"           #Instrument IP Address
TCP_PORT = 5025

def instsend(s, command):
    command += "\n"
    s.send(command.encode())
    return

def instrecv(s):
    return s.recv(1024).decode()

def instquery(s, command):
    instsend(s, command)
    return instrecv(s)

s = socket.socket()
s.connect((TCP_IP, TCP_PORT))       #Pi will be client to server instrument
instsend(s, "*RST")
print(instquery(s, "*IDN?"))

instsend(s, "FORM:DATA ASCII")     #Format scan data into ascii

instsend(s, "FUNC 'VOLT:AC', (@101)") #Measure AC Volts on channel 1
instsend(s, "FUNC 'FREQ', (@102)")  #Measure Frequency on channel 2
instsend(s, "FUNC 'PER', (@103)")  #Measure Period on channel 3
instsend(s, "FUNC 'CURR:AC', (@121)") #Measure AC Current on channel 21

instsend(s, "ROUT:SCAN:COUN:SCAN 10") #Set scan count to 10
instsend(s, "ROUT:SCAN:CRE (@101:103, 121)") #Scan channels 101-103 and 121
instsend(s, "INIT")                #Initiate the scan

time.sleep(0.5)
trigState = instquery(s, ':TRIG:STAT?') # query the state of the scan activity which is either
# "running" when the measurements are being made
# or "waiting" when the interval (delay) between scans
# is active

while("RUNNING" in trigState) | ("WAITING" in trigState):
    print("Running...")
    time.sleep(5.0)
    trigState = instquery(s, ':TRIG:STAT?')

data = instquery(s, 'TRAC:DATA? 1, 40, "defbuffer1", CHAN, READ, UNIT')
print(data)                          #Print readings to console
s.close()                             #Close the connection

with open("/home/pi/AC_Parameters.csv", "a") as log: #send data to csv file
    log.write(data)
```

Contact Information:

Australia* 1 800 709 465
Austria 00800 2255 4835
Balkans, Israel, South Africa and other ISE Countries +41 52 675 3777
Belgium* 00800 2255 4835
Brazil +55 (11) 3759 7627
Canada 1 800 833 9200
Central East Europe / Baltics +41 52 675 3777
Central Europe / Greece +41 52 675 3777
Denmark +45 80 88 1401
Finland +41 52 675 3777
France* 00800 2255 4835
Germany* 00800 2255 4835
Hong Kong 400 820 5835
India 000 800 650 1835
Indonesia 007 803 601 5249
Italy 00800 2255 4835
Japan 81 (3) 6714 3086
Luxembourg +41 52 675 3777
Malaysia 1 800 22 55835
Mexico, Central/South America and Caribbean 52 (55) 56 04 50 90
Middle East, Asia, and North Africa +41 52 675 3777
The Netherlands* 00800 2255 4835
New Zealand 0800 800 238
Norway 800 16098
People's Republic of China 400 820 5835
Philippines 1 800 1601 0077
Poland +41 52 675 3777
Portugal 80 08 12370
Republic of Korea +82 2 565 1455
Russia / CIS +7 (495) 6647564
Singapore 800 6011 473
South Africa +41 52 675 3777
Spain* 00800 2255 4835
Sweden* 00800 2255 4835
Switzerland* 00800 2255 4835
Taiwan 886 (2) 2656 6688
Thailand 1 800 011 931
United Kingdom / Ireland* 00800 2255 4835
USA 1 800 833 9200
Vietnam 12060128

* European toll-free number. If not accessible, call: +41 52 675 3777

Rev. 02.2018



Find more valuable resources at TEK.COM

Copyright © Tektronix. All rights reserved. Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specification and price change privileges reserved. TEKTRONIX and TEK are registered trademarks of Tektronix, Inc. All other trade names referenced are the service marks, trademarks or registered trademarks of their respective companies.

090618 SBG 1KW-61463-0

