Raspberry Pi Hands-on

LEDの明るさやDCモーターの速度を変える

&

Firefox OSアプリからの制御

2015/4/26版

author : Kazuko Shikiya @ UniversalGravitation.jp

# はじめに

このハンズオンでは、

Raspberry PiにRaspbian（Debian Wheezy）セットアップした環境にNode.js + pi-blaster をインストールし、LEDやDCモーターをPWM制御します。

また、

Firefox OSアプリからも制御が行えるようWebSocket（WebSocket-Nodeを利用）通信で動くようにします。

**ただし、**

**2015/4/26のハンズオン環境では、環境作成・インストールは、ハンズオンから除外します。**

## 参考URL

[Raspberry Pi : ](https://www.raspberrypi.org/)

[Node.js : ](http://nodejs.jp/)

[pi-blaster.js : ](https://github.com/sarfata/pi-blaster.js/)

[WebSocket-Node : ](https://github.com/theturtle32/WebSocket-Node)

[RS コンポーネンツ : ](http://jp.rs-online.com/web/)

# PCの準備

このハンズオンで利用するPC環境のセットアップをします。

## \_ すべてのPC共通 \_

1. Raspberry PiとPCを接続するためのFT232RL用ドライバをインストールします。

## \_ Windowsをお使いの場合のみ \_

（Mac OS Xは、OSにターミナル機能が付いているので、この手順は不要です。）

2. Raspberry Piを操作するためのターミナルとしてTeraTermをインストールします。

# Firefox OS端末接続の準備

Firefox OSを利用した端末にハンズオン用アプリを入れて動作確認します。アプリをそれぞれがお使いの環境に合わせて変更したり、Firefox OS端末をPCと接続したりするため、PC側の準備をします。

参照URLにある内容を設定します。

## \_ すべてのPC共通 \_

1.Firefoxブラウザのインストール

参照URL：**http://goo.gl/2kWWMw**

2.Firefoxシミュレーター(Ver 2.0)のインストール

参照URL：**http://goo.gl/gnddhV**

## \_ Windowsをお使いの場合のみ \_

3.USBドライバーのインストール

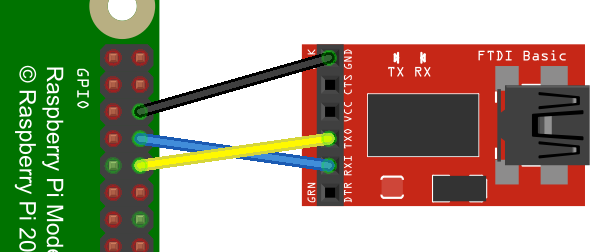
参照URL：**http://goo.gl/5l3t91**

# Raspberry Piの起動

Raspberry PiとPCを接続し、Raspberry Piを起動します。

1. シリアル通信用の機器を用意します。回路図の通りに、Raspberry PiとFT232RLをつなぎます。

<<<<<回路図：シリアル>>>>>



2. FT232RLにデータ転送用USBケーブルをつなぎます。

## \_ Windowsをお使いの場合 \_

3. データ転送用USBとPCをつなぎます。このとき、FT232RLについている小さなLEDが光ります。

4. TeraTermを起動します。ttermpro.exeを使います。

5. COMポート経由でRaspberry Piとつなぎます。TeraTermの[New connection]ウィンドウが開いているので、[Serial]側のラジオボタンをクリックし、[OK]ボタンをクリックしてください。（[Serial]側のラジオボタンがクリックできない場合は、FT232RLが認識されていないかドライバがインストールできなかった時です。回路図、USBケーブル、ドライバを確認してください。）

6. シリアル通信のボーレートを指定します。メニューの[Setup]->[Serial port...]->[Baud Rate:]のプルダウンリストから[115200]を選びます。

7. Raspberry Piの電源を入れます。SDカードを刺してから、Raspberry Pi本体のマイクロUSBに電源ケーブルを刺します。

8. Raspberry Piが起動します。このときに、本体のLEDが点滅します。

## \_ Mac OS Xをお使いの場合 \_

3. Raspberry Piの電源を入れます。SDカードを刺してから、Raspberry Pi本体のマイクロUSBに電源ケーブルを刺します。

4. Raspberry Piが起動します。このときに、本体のLEDが点滅します。

5. データ転送用USBとPCをつなぎます。このとき、FT232RLについている小さなLEDが光ります。

6. コンソールを開きます。

7. シリアル接続のパスを調べます。以下コマンド「ls /dev/tty.\*」を入力してtty番号やttyprintk以外のものを探してください。（ttyA??0になっていることが多いです。?は何かの文字。）

**$ ls /dev/tty.\***

8. コマンドでシリアル接続します。以下コマンド「screen シリアル接続のパス 115200」を入力してください。

**$ screen シリアル接続のパス 115200**

9. 一度、Enterキーを押してください。

これで、Raspberry Piが起動し、PCの端末（ターミナル）とつながりました。

Raspberry PiのIDとpasswordは、全員共通で、以下の通りに設定されています。IDとpasswordを使ってログインしてください。

**[ID :] pi**

**[password :] raspberry**

# Raspberry Piのネットワーク設定

ハンズオンの後半で、Firefox OS端末からRaspberry Piに接続します。そのため、Firefox OS端末とRaspberry Piが接続できるように、Raspberry Piにネットワーク設定します。

1. 以下のコマンドで、ネットワーク接続用のファイルを開きます。

**$ sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf**

2. ファイルの内容を、指定された接続先にします。（SSID値とpsk値は、別途お知らせします。）

<< wpa\_supplicant.confファイルの内容>>

**ctrl\_interface=DIR=/var/run/wpa\_supplicant GROUP=netdev**

**update\_config=1**

**network={**

**ssid="機器のSSID値"**

**psk="機器のpsk値"**

**}**

3. ファイルの保存は、**ctrl + x**でできます。保存時に Y / N を聞いてくるので、Yで終わってください。

この設定は、Raspberry Piの再起動後に有効になります。

（参考URL：**http://goo.gl/YOkDIH**）

# Raspberry Piの停止

Raspberry Piを停止するときは、コマンドでshutdownしてから、電源ケーブルを抜きます。順序を間違えると機材やOSを破壊してしまうことがあります。

1. 停止のためのコマンドを実行する。以下コマンドを入力します。

**$ sudo shutdown -h now**

2. Raspberry Pi側の小さいLED（ACT側）が連続点滅し終わって、小さいLEDが１つだけが点灯している状態（PWR側）になったら、電源ケーブルを抜きます。

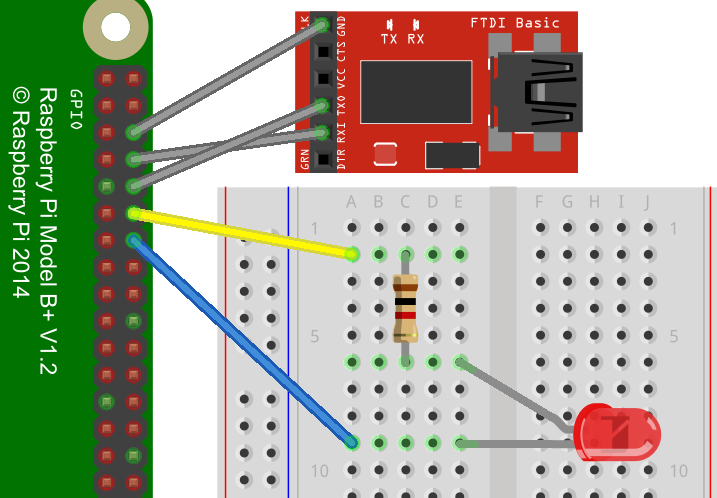
新しい回路に組み替えるときは、Raspberry Piを停止したほうが安全です。

## ハンズオン１

LED回路を組み立てます。

1. 回路図の通りに組み立てます。

<<<<<回路図：LED>>>>>



**ジャンパーケーブルを刺すところで位置を間違えると、LEDが点灯しなかったり、回路やRaspberry Piが破壊されたりします。注意してください。**

1.1 LEDの線は長い線のほうが、抵抗と同じライン上に刺さっています。

1.2 抵抗は、左右どちら向きでも構いません。

できあがった回路を、次のハンズオン２で動作確認します。

# ## ハンズオン２

LED回路をshellで制御します。

Raspberry Piを起動し、shellコマンドを入力して、LEDを点灯/消灯します。

1. 以下のコマンドで、shellコマンドでLEDに給電するGPIOを初期化処理をします。

**$ echo 18 > /sys/class/gpio/export**

**$ echo out > /sys/class/gpio/gpio18/direction**

2. 以下のコマンドで、LEDを点灯します。

**$ echo 1 > /sys/class/gpio/gpio18/value**

3. 以下のコマンドで、LEDを消灯します。

**$ echo 0 > /sys/class/gpio/gpio18/value**

4. 以下のコマンドで、GPIOの終了処理をします。終了処理が行われないと、次回のGPIOアクセスでエラーが発生します。

**$ echo 18 > /sys/class/gpio/unexport**

5. LEDが点灯しているかどうかをコマンドで確認することも可能です。以下のコマンドを実行し、1が表示されたら点灯中、それ以外は消灯またはエラー中です。

**$ cat /sys/class/gpio/gpio18/value**

# ## ハンズオン３

Node.jsでプログラムを書いてみます。

Raspberry Pi上にインストールされているNode.jsを利用したプログラムを作り、実行します。

1. 以下コマンドで、これから作るプログラムのファイルを作ります。

**$ nano handson3.js**

2. 以下の命令を、ファイルに書き込み、ファイルを保存します。ファイルの保存は、**ctrl + x**でできます。保存時に Y / N を聞いてくるので、Yで終わってください。

<<プログラムの内容>>

**console.log('Hello Node & Raspberry Pi');**

3. Node.jsで実行する場合には、nodeコマンドを使います。以下コマンドで、作ったプログラムを実行してください。

**$ node hanson3.js**

画面上に「Hello Node & Raspberry Pi」が表示されます。

# ## ハンズオン４

LED回路をNode.jsで制御します。

1. ハンズオン３を参考にして、handson4.jsというファイルに、以下プログラムを記入して、実行してください。

<<プログラムの内容>>

**var fs = require('fs');**

**fs.writeFileSync('/sys/class/gpio/export', 18);**

**fs.writeFileSync('/sys/class/gpio/gpio17/direction', 'out');**

**fs.writeFileSync('/sys/class/gpio/gpio17/value', 1);**

**var value = fs.readFileSync('/sys/class/gpio/gpio18/value','ascii');**

**console.log(value);**

**fs.writeFileSync('/sys/class/gpio/unexport', 18);**

LEDが一瞬点灯したあと消灯します。画面上に「1」が表示されます。

# ## ハンズオン５

LED回路の明るさをpi-blasterとshellで制御します。GPIOの初期化処理は必要ありません。GPIOの終了処理はありますが、実行しなくても、次回も正しく点灯できます。

1. 以下のコマンドで、LEDを点灯します。

**$ echo "18=1" > /dev/pi-blaster**

2. １よりも暗くLEDを点灯します。

**$ echo "18=0.4" > /dev/pi-blaster**

3. 以下のコマンドで、LEDを消灯します。

**$ echo "18=0" > /dev/pi-blaster**

4. 以下のコマンドで、GPIOの終了処理をします。

**$ echo "release 18" > /dev/pi-blaster**

「=」の後ろにある値は、0<= x <=1の範囲が有効値です。小数点値を使って明るさが指定できます。

# ## ハンズオン６

LED回路の明るさをNode.jsとpi-blasterで制御します。

1. ハンズオン３を参考にして、handson5.jsというファイルに、以下プログラムを記入して、実行してください。

<<プログラムの内容>>

**var piblaster = require('pi-blaster.js');**

**for(x=1;x<=10;x++){**

**for(y=1;y<=10;y++){**

**piblaster.setPwm(18, (x \* y) / 100);**

**}**

**}**

**piblaster.setPwm(18, 0);**

LEDが暗い光で点灯して、だんだん明るくなり、最後に消灯します。

# ## ハンズオン７

Node.jsでWebSocketを使ったプログラムを書いてみます。

1. ハンズオン３を参考にして、handson7.jsというファイルに、以下プログラムを記入して、実行してください。（「// change later!!!」と書いてある行は、ハンズオン８のためのコメントです。この行は実行には影響がないので、省いてもかまいません。）

<<プログラムの内容>>

**var fs = require('fs');**

**var httpServer = require('http').createServer().listen(8000);**

**var WSServer = require('websocket').server;**

**var webSocketServer = new WSServer({**

**httpServer: httpServer**

**});**

**webSocketServer.on('request', function (request) {**

**var connection = request.accept(null, request.origin);**

**connection.on('message', function(message) {**

**var json = JSON.parse(message.utf8Data);**

**// change later!!!**

**console.log('port : '+ json.port + ', slider : ' + json.brightness);**

**});**

**connection.on('close', function() {**

**console.log('connection closed');**

**});**

**});**

実行すると、プログラムはクライアントからの要求を待っている状態になります。

プログラムを停止するときには、**ctrl + c**を使ってください。

# ## ハンズオン８

ハンズオン７で作ったプログラムをFirefoxOSアプリから呼び出してみます。

1. ハンズオン用電子データ資料から「websocket\_gpio」フォルダをPCにダウンロードします。

2. Firefoxブラウザを開きます。

3. WebIDEを開きます。

4. [ランタイムを選択]から[シミュレータ]にある[Firefox OS 2.0]を選びます。

5. [アプリを開く]から[パッケージ型アプリ...]を選びます。

6. [フォルダ]に、さきほどコピーした「websocket\_gpio」フォルダを選びます。

7. WebIDEに「GPIO」というアプリが表示されます。

8. 「GPIO」アプリのツリーリストの中にある「js/maim.js」を選びます。ソースコード内のIPアドレスをRaspberry Piが利用しているIPアドレスに変更します。

9. WebIDEの上のほうにある「グレーの三角マーク」をクリックします。

10. シミュレータに「GPIO」アプリが表示されます。

11. スライダーを動かすと、Raspberry Piのターミナル上に、そのときの値が表示されます。

シミュレータではなく、Firefox OS端末を利用することもできます。その場合には、手順4.のところで[シミュレータ]->[Firefox OS 2.0]ではなく[USBデバイス]に表示されているFirefox OS端末を選んでください。

# ## ハンズオン９

LED回路の明るさをFirefoxOSアプリで制御します。

1. ハンズオン７で作ったプログラムhandson7.jsをhandson9.jsにコピーします。その後handson9.jsの中身を以下のプログラムになるように改造し、実行してください。改造箇所は「// change later!!!」と書いてある場所のすぐ下の行です。

<<プログラムの内容>>

**var fs = require('fs');**

**var httpServer = require('http').createServer().listen(8000);**

**var WSServer = require('websocket').server;**

**var webSocketServer = new WSServer({**

**httpServer: httpServer**

**});**

**webSocketServer.on('request', function (request) {**

**var connection = request.accept(null, request.origin);**

**connection.on('message', function(message) {**

**var json = JSON.parse(message.utf8Data);**

**// change later!!!**

**piblaster.setPwm(json.port, json.brightness / 100);**

**});**

**connection.on('close', function() {**

**console.log('connection closed');**

**});**

**});**

ハンズオン８を参考にして、シミュレータもしくはFirefox OS端末からFirefoxOSアプリから呼び出します。

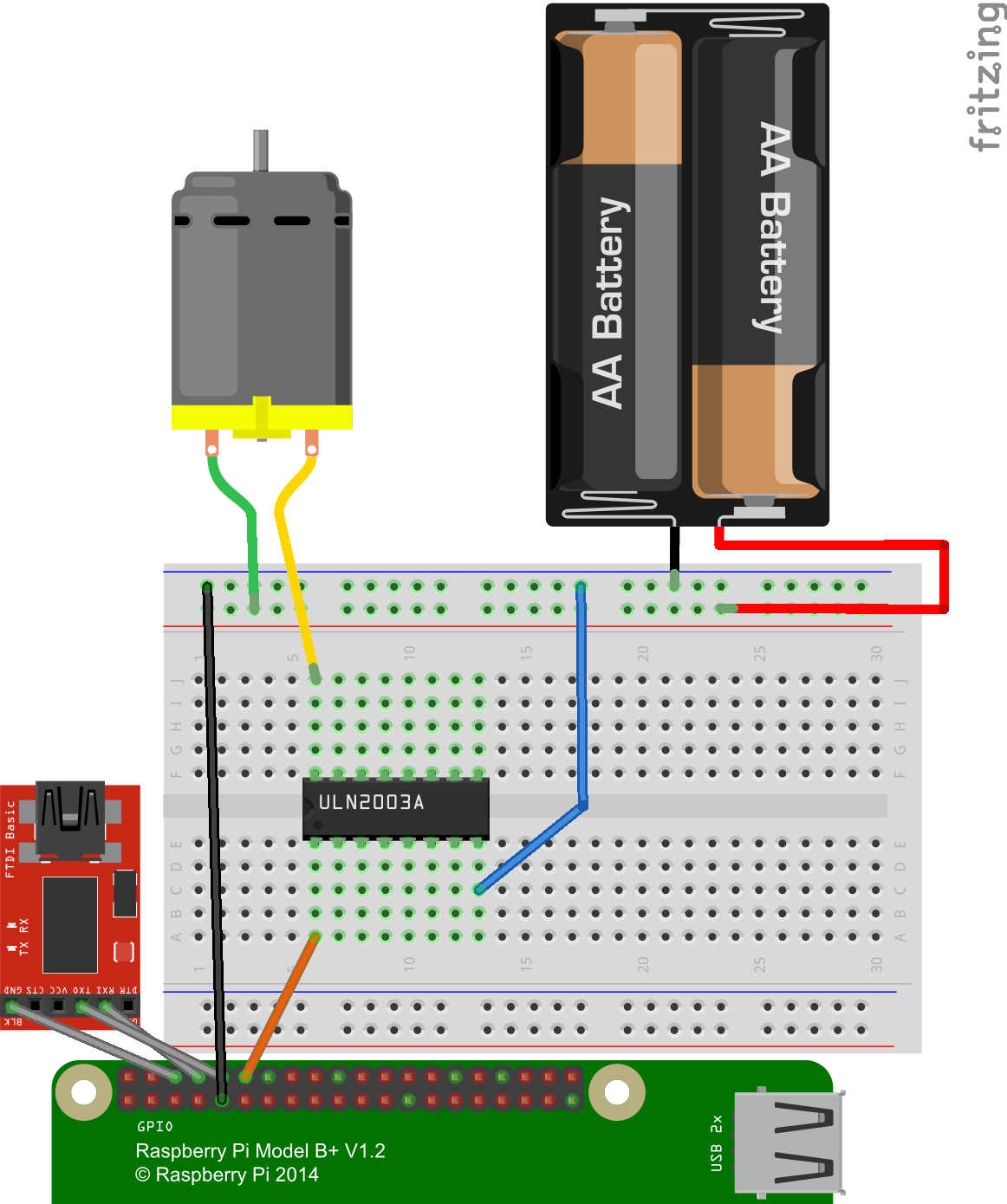
# ## ハンズオン１０

DCモーター回路を組み立てます。

1. 「Raspberry Piの停止」を参考に、手順通りにRaspberry Piの電源を切ります。

2. 回路図の通りに組み立てます。

<<<<<回路図：DCモーター>>>>>



**各部品を刺すところで位置や向きを間違えると、モーターが回らなかったり、回路やRaspberry Piが破壊されたりします。注意してください。**

# ## ハンズオン１１

DCモーター回路のスピードをFirefoxOSアプリで制御します。

Raspberry Pi側、Firefox OSアプリ側、どちらも、ハンズオン９のものが、そのまま使えます。試してみてください。

# さいごに

ハンズオンはこれで終了です。Raspberry Piで作成したソースコードを削除してください。

**$ rm \*.js**

お疲れさまです。お楽しみいただけましたでしょうか？

Firefox OSアプリ側は、別ブースにてご説明しておりますので、ぜひそちらにもお立ち寄りください。

利用した機材については、すべて返却をお願いいたします。

元の絶縁ビニール袋等にきちんと入れてご返却ください。細かい部品もありますので、紛失のないよう、気を付けて片付けてください。

ソースコード等は、以下URLで公開していますので、そちらから入手をお願いいたします。

<https://github.com/KazukoShikiya/RaspPi_PWM_HandsOn>

Raspberry Piのユーザー組織が、全国にいくつかございます。ぜひ、ネットで探してご参加ください。

ご参加ありがとうございました。

敷矢　和子

　Japanese Raspberry Pi Users Groupメンバー

http://raspi.jp/

　ゆるふあ女子のためのRaspPi勉強会 主催

https://raspijoshi.doorkeeper.jp/