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

2015/8/8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、**

**今回は、RaspberryPi側の環境作成・インストールは、ハンズオンから除外します。**

**環境作成・インストールの手順は別途公開していますので各自でお試しください。**

## **資料公開URL**

**https://github.com/KazukoShikiya/RaspPi\_PWM\_HandsOn**

## 参考URL

Raspberry PiとRaspbianに関する情報：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

Raspberry Piの購入（RS コンポーネンツ）　：　http://jp.rs-online.com/web/

# PCの準備１

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。

この作業は、会場が提供している「会場外接続可能ネットワーク」に接続しておく必要があります。接続設定は、会場提供の資料をご覧ください。

# Firefox環境の準備

ハンズオンの最後に、Firefox OS端末のシミュレータを使います。

FirefoxブラウザやFirefoxシミュレータのインストールが終わっていない場合には、これらをインストールします。

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**

# （任意）FirefoxOS端末接続の準備

ハンズオン後に、FireFoxOS端末を使って動作確認したい方で、PCがWindowsの場合には、以下参照URLにある内容を設定します。

これはハンズオンの範囲外のため、任意になります。

## \_ Windowsをお使いの場合のみ \_

1.USBドライバーのインストール

参照URL：**http://goo.gl/5l3t91**

ここまで終了したら、「会場外接続可能ネットワーク」を切断してください。

# PCの準備２

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。

# Raspberry Pi - PC接続環境の準備

Raspberry PiをHDMIモニタやUSBキーボード無しで利用する方法はいくつかありますが、今回のハンズオンでは、FT232RLやターミナル機能を使って利用します。

USBで配布したデータ内にあるドライバやアプリをインストールしてください。

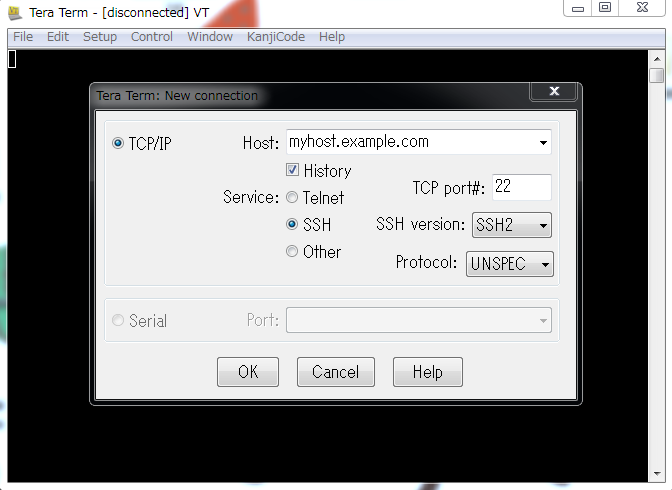
## \_ すべてのPC共通 \_

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。「FT232RL」フォルダの中に、OSごとに分かれたフォルダがありますので、利用環境に合わせて選んで、インストールしてください。

## \_ Windowsをお使いの場合のみ \_

2. Raspberry Piを操作するためのターミナルとしてTeraTermを利用しますので、起動できるか確認します。

「Windows用TeraTerm」と書いてあるフォルダの中を見てください。ショートカットで「\_これを実行します」と書いてあるアイコンがあります。これをクリックして開いてみてください。以下のような画面が出ていれば起動成功です。



起動成功を確認したら、画面右上にある「×」ボタンを押して、アプリを終わらせてください。

# Raspberry Piの接続と起動

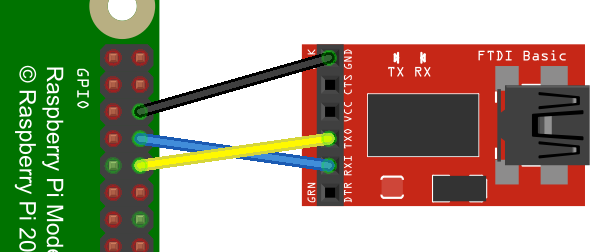
Raspberry PiとPCを接続し、Raspberry Piを起動します。

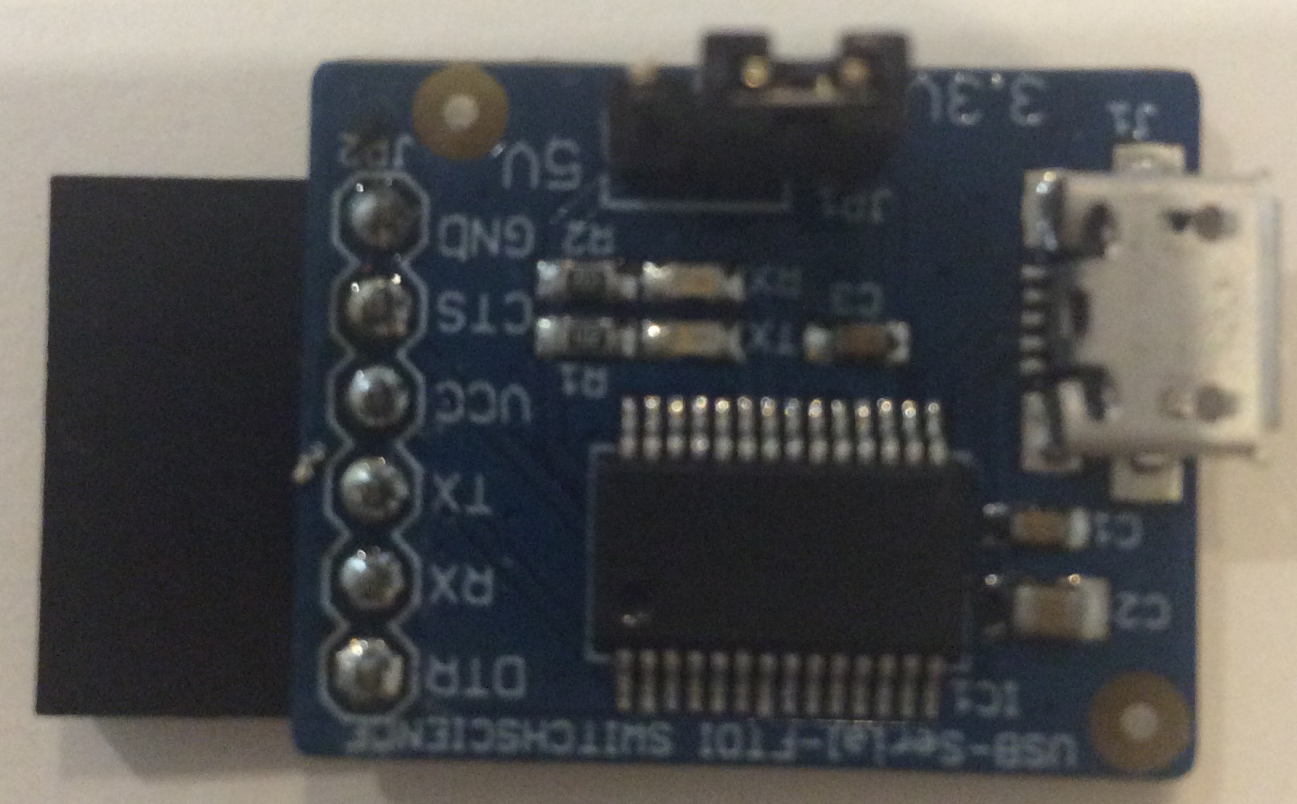
1. シリアル通信用の機器を用意します。

利用するFT232RLが「赤」の場合には回路図の通りに、Raspberry PiとFT232RLをジャンパーケーブルでつなぎます。

利用するFT232RLが「青」の場合には、回路図と下の写真に書いてある「GND」「TXO」「RXI」を参考にして、FT232RLをジャンパーケーブルでつなぎます。

<<<<<回路図：シリアル>>>>>





**TXO**

**RXI**

**GND**

1. ジャンパーケーブルをつないだら、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 \_（Mac OSをお使いの場合には、ここは飛ばします）

1. 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。一瞬なので、注意して見守っていてください。
2. TeraTermを起動します。「Windows用TeraTerm」の「\_これを実行します」を起動します。
3. 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。）
4. 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。
5. Raspberry PiにSDカードを刺してから、Raspberry Pi本体に電源ケーブルを刺します。
6. Raspberry Piが起動します。このときに、本体のLEDが点滅し、TeraTermには起動情報が表示されます。**（Windowsここまで）**

## \_ Mac OS Xをお使いの場合 \_（Windowsをお使いの場合には、ここは飛ばします）

1. Raspberry PiにSDカードを刺してから、Raspberry Pi本体に電源ケーブルを刺します。
2. Raspberry Piが起動します。このときに、本体のLEDが点滅します。
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。一瞬なので、注意して見守っていてください。
4. コンソールを開きます。
5. シリアル接続のパスを調べます。コマンドで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.\***

1. 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**

1. screenコマンド入力後に、一度、Enterキーを押してください。**（MacOSここまで）**

**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へのログイン

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

1. 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）または全てが無点灯状態になったら、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。

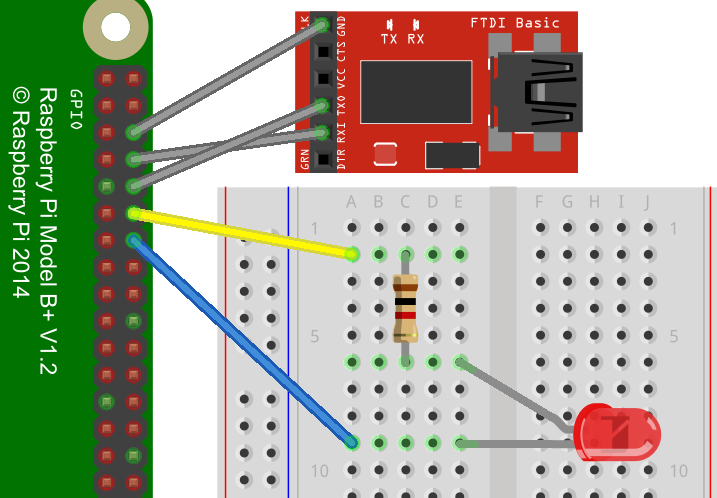
# ## ハンズオン１

LED回路を組み立てます。

1. 「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：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。

この回路は、黄色のジャンパーケーブルからLEDの長い方（プラス側の線）に電流が流れ、Raspberry Piのグラウンドがアースとなります。

（この回路の場合には、抵抗は、プラス側マイナス側のどちらにあっても構いまわない。）

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**

* “Permission Denied” というエラーメッセージが出る場合は、すべてのコマンドを、以下の形式で入力して実行してください。

**$ sudo sh –c “ ここに実行したいコマンドを書きます ”**

(例)

**$ sudo sh –c “echo out > /sys/class/gpio/gpio18/direction”**

* “Resource busy” というエラーメッセージが出る場合は、何らかの原因で前回の処理が正しく終わっていません。5. のGPIO終了処理コマンドを実行後に、1.を初めからやり直してください。

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. 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**

5. 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**

ハンズオンに出てくる「18」は、Raspberry PiのGPIOピンにつけられた番号です。この番号のピンを通じて電流を流します。

# ## 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**

1. 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**を押します。保存時に Y / N を聞いてくるので、Y→Enter を押すと終了します。

<<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 handson3.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。

**$ nano handson4.js**

<<プログラムの内容>>

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

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

**fs.writeFileSync('/sys/class/gpio/gpio18/direction', 'out');**

**fs.writeFileSync('/sys/class/gpio/gpio18/value', 1);**

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

**console.log(value);**

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

2. 実行コマンド

**$ node handson4.js**

“Permission Denied” というエラーメッセージが出る場合は、以下のコマンドを実行してください。

**$ sudo node handson4.js**

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、handson6.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。

**$ nano hanson6.js**

<<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);**

2. 実行コマンド

**$ node handson6.js**

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。）

**$ nano hanson7.js**

（プログラム内容は、次のページ）

<<プログラムの内容>>

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

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

**// change later!!!**

**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() {**

**// change later!!!**

**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。特に何もメッセージが出ない状態であれば、動作成功です。

このプログラムを動かしたまま、次のハンズオンに進んでください。

（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。USBで配布したデータ内にある「websocket\_gpio」フォルダを利用します。

ここからのハンズオンは、

会場が提供している「会場内ネットワーク」に接続しておく必要があります。接続設定は、各自に配られている「PC用会場内ネットワーク設定」をご覧ください。

1. Firefoxブラウザを開きます。
2. WebIDEを開きます。
3. [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。シミュレータが起動したら、シミュレータ内の白丸をマウスでクリック、錠前アイコン側にスライドさせて画面ロックを解除しておきます。
4. [アプリを開く]から[パッケージ型アプリ...]を選びます。
5. [フォルダ]に、「websocket\_gpio」フォルダを選びます。
6. WebIDEに「GPIO」というアプリが表示されます。
7. 「GPIO」アプリのツリーリストの中にある「js/main.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。

RaspberryPiのIPアドレスは、以下のコマンドで確認できます。コマンドを実行後に表示される「wlan0」項目の「inet addr:」と書いてあるところに、IPアドレスがあります。あとで使うので控えておきます。

**$ ifconfig**

1. WebIDEの上のほうにある「実行ボタン（グレーの三角マーク）」をクリックすると、シミュレータに「GPIO」アプリが表示されます。
2. スライダを、セットしたい位置にクリックすると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。)

1. 実行を終えるときには、WebIDEの上のほうにある「停止ボタン（グレーの■マーク）」をクリックします。これを押さずに終わろうとするとRaspberry Piのシステムが壊れることがあります。

# ## 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とハンズオン８で動かしていたhandson7.jsプログラムを停止します。**ctrl + c**でプログラム停止してください。
2. 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。

ファイルのコピーは以下のコマンドで確認できます。

**$ cp handson7.js handson9.js**

1. 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「 // <- here」と書いてある場所です。（全部で５行です。）

（ページをまたいでいますが、１つのファイルとして書いてください。）

<<プログラムの内容>>

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

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

**// change later!!!**

**var port = 0; // <- here**

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

**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!!!**

**port = json.port; // <- here**

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

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

**});**

（プログラムは、次のページにも続く）

（前ページのプログラム続き）

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

**// change later!!!**

**piblaster.setPwm(port,0); // <- here**

**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。

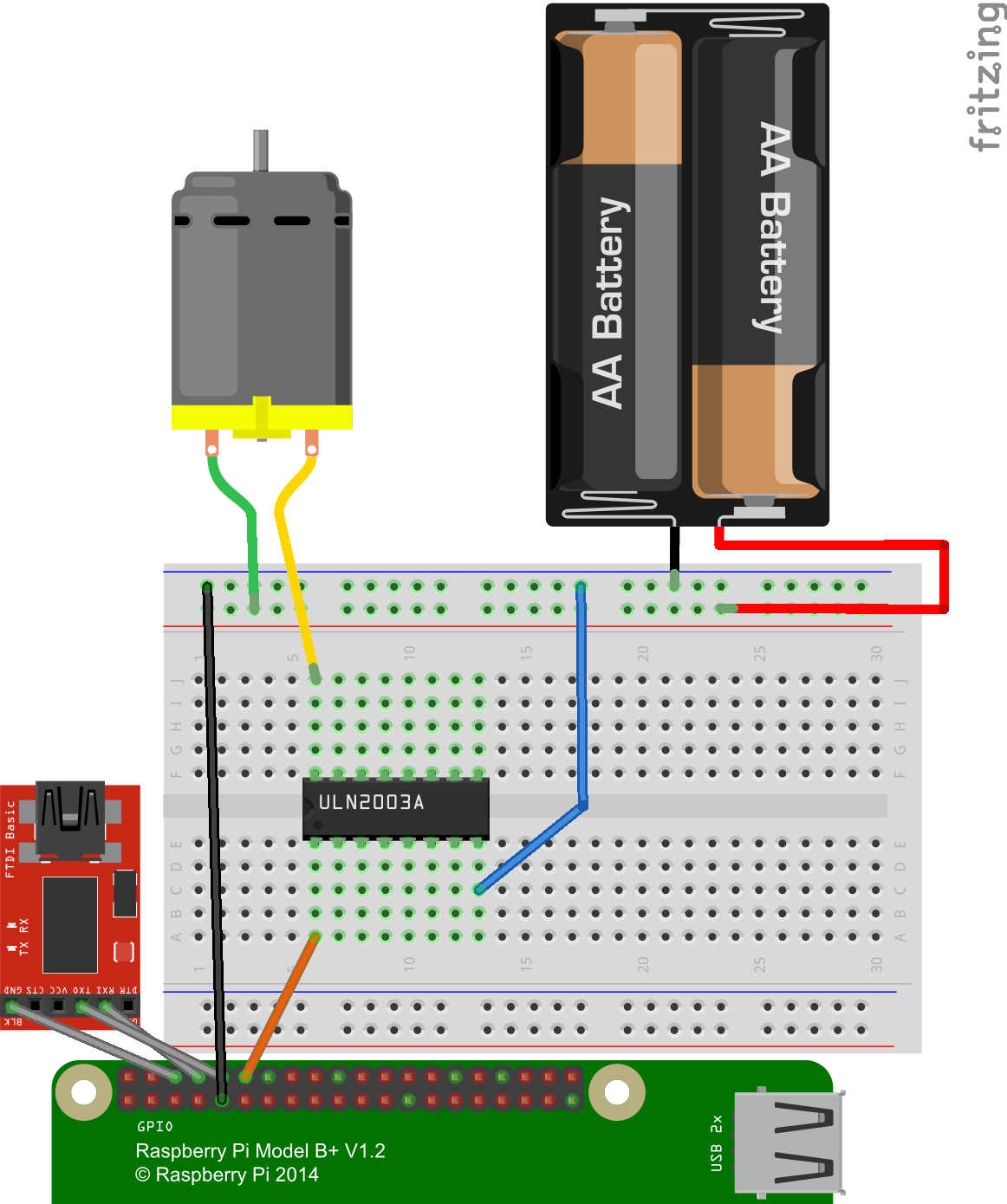
# ## ハンズオン１０

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。

この資料や完成ソースコード等は、以下URLで公開しています。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/