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

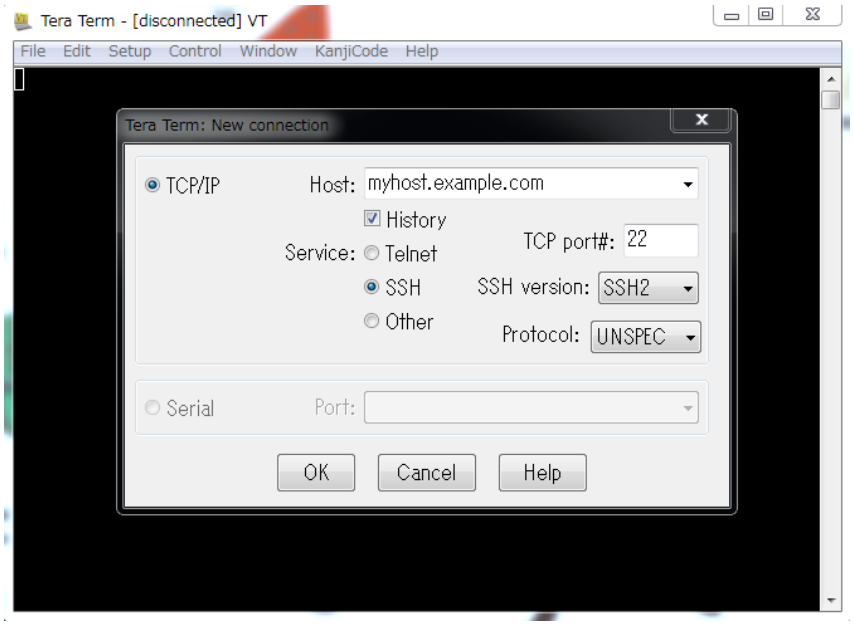
Raspberry Pi を HDMI モニタや USB キーボード無しで利用する方法はいくつかありますが、今回のハンズオンでは、FT232RL やターミナル機能を使って利用します。
USB で配布したデータ内にあるドライバやアプリをインストールしてください。

_ 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。



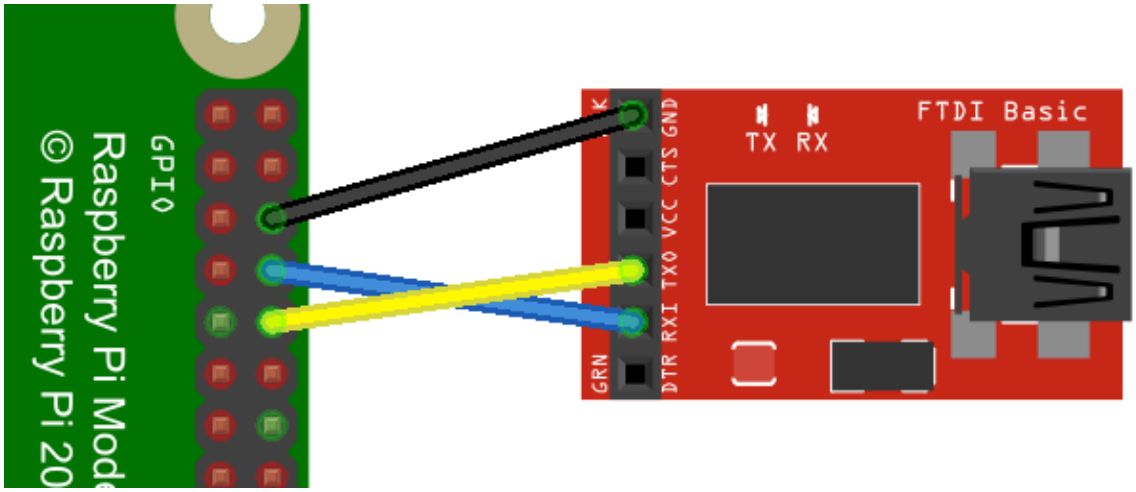
起動成功を確認したら、画面右上にある「×」ボタンを押して、アプリを終わらせてください。

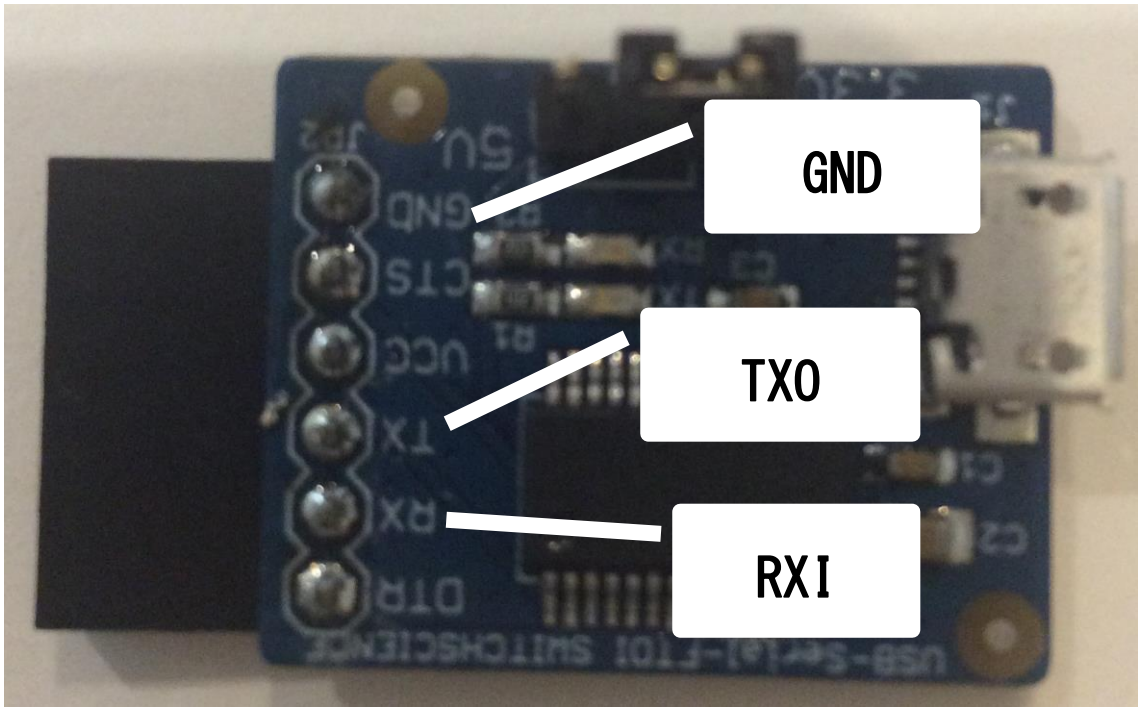
Raspberry Pi の接続と起動

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>>>>>





2. 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）

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. 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。

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 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。

8. 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）

3. 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。

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。一瞬なので、注意して見守っていてください。

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。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.*
```

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. 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。

10. 「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
```

11. 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。

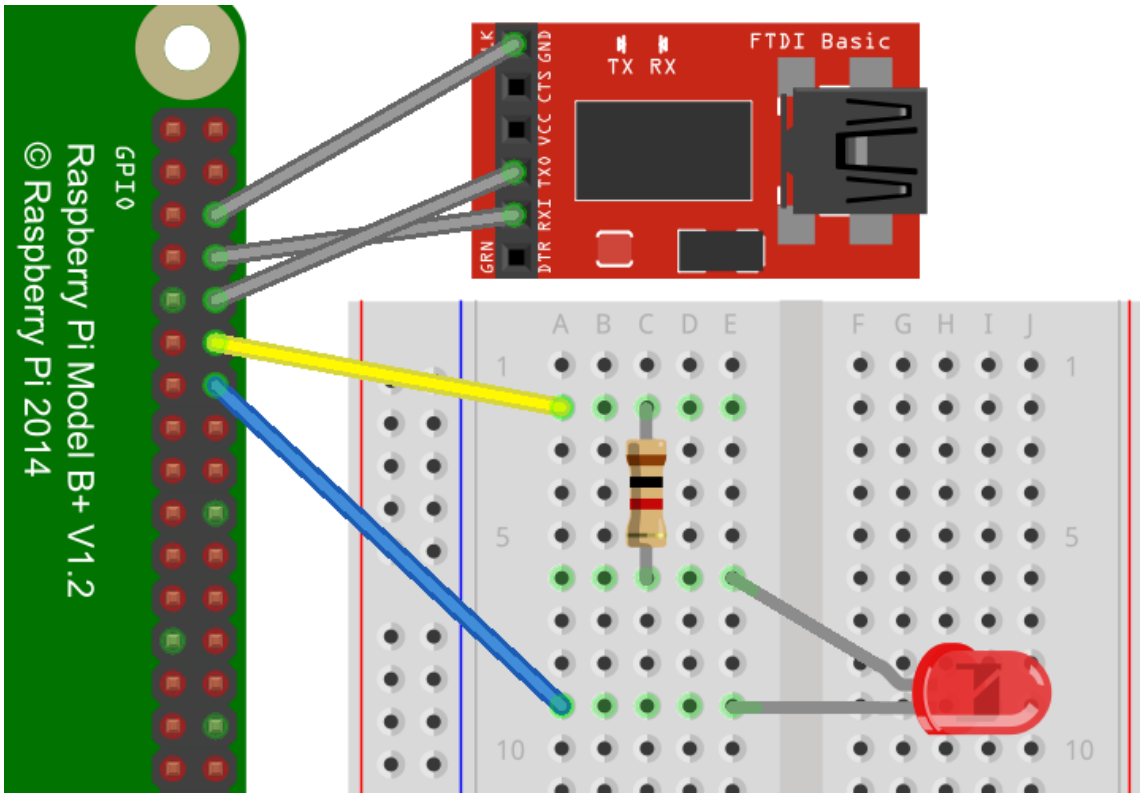
新しい回路に組み替えるときは、Raspberry Pi を停止したほうが安全です。

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
```

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 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
```

8. 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。

9. 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。）

10. 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
   ```

3. 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');
  });
});
```

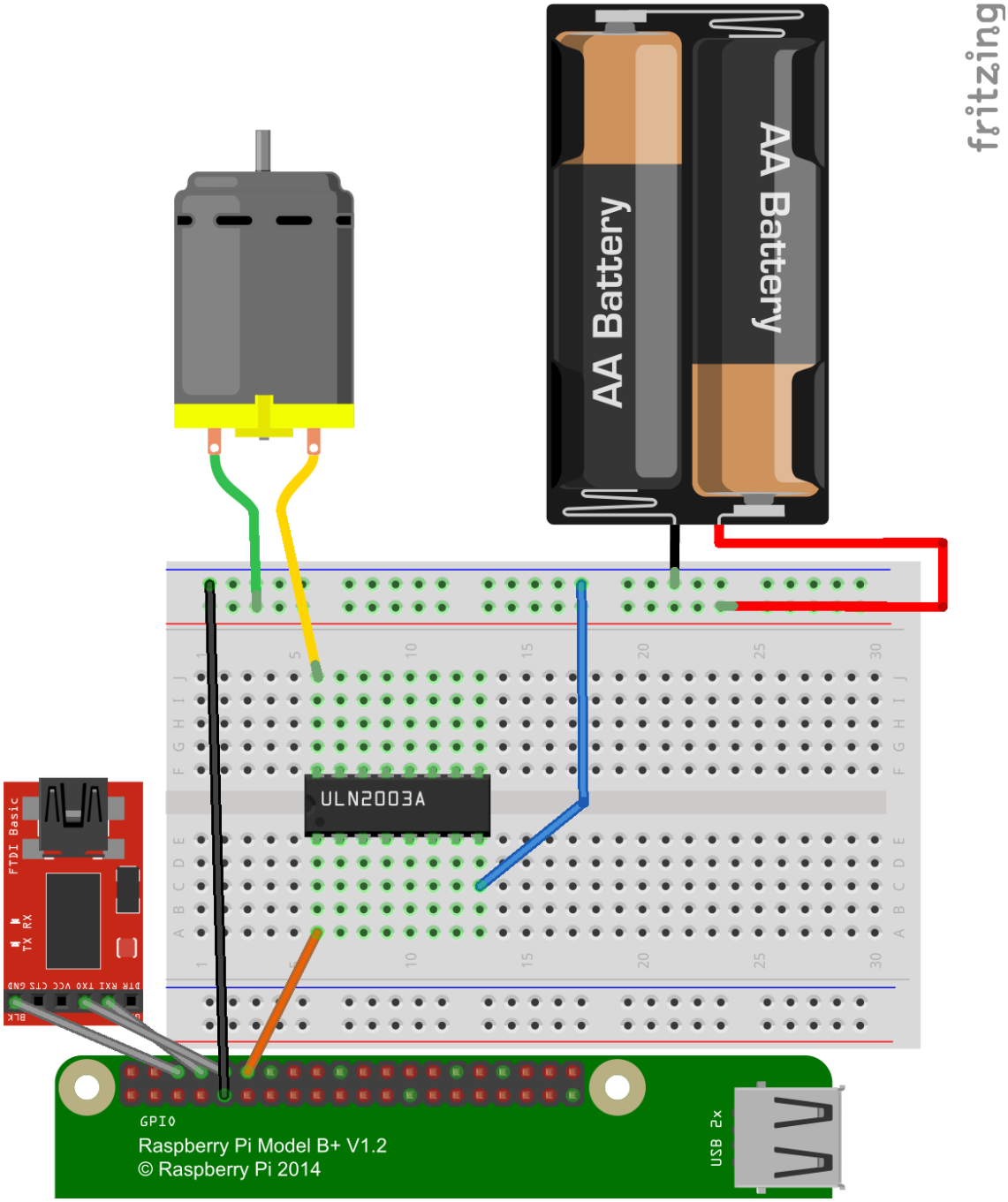
ハンズオン８を参考にして、シミュレータもしくは Firefox OS 端末から FirefoxOS アプリから呼び出します。

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/