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
改定ドラフト 2015/07/03(作業中)
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。お渡しした USB メモリの OS ごとのフォルダにある Driver フォルダ内プログラムを実行してください。ダウンロードしてインストールする場合には以下の URL からダウンロードできます。

参照 URL : http://www.ftdichip.com/Drivers/VCP.htm

_ 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。お渡しした USB メモリの OS ごとのフォルダにある TeraTerm フォルダ内プログラムを実行してください。ダウンロードしてインストールする場合には以下の URL からダウンロードできます。

参照 URL : http://goo.gl/r2IaaA

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 にある内容を全て設定します。

_ 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

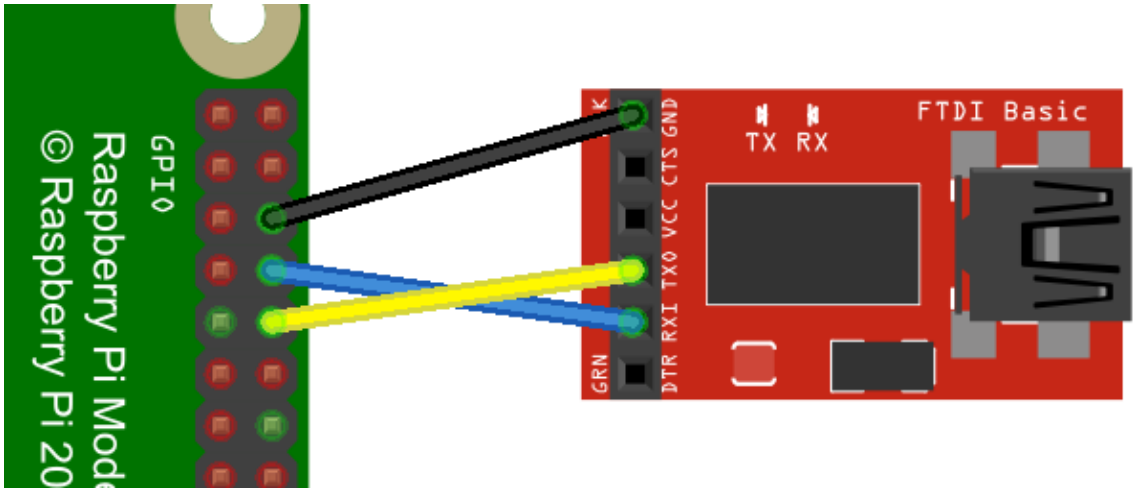
参照 URL : http://goo.gl/5l3t91

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 カードと無線 USB 子機を刺してから、

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 カードと無線 USB 子機を刺してから、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/Y0kDIH**）

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。

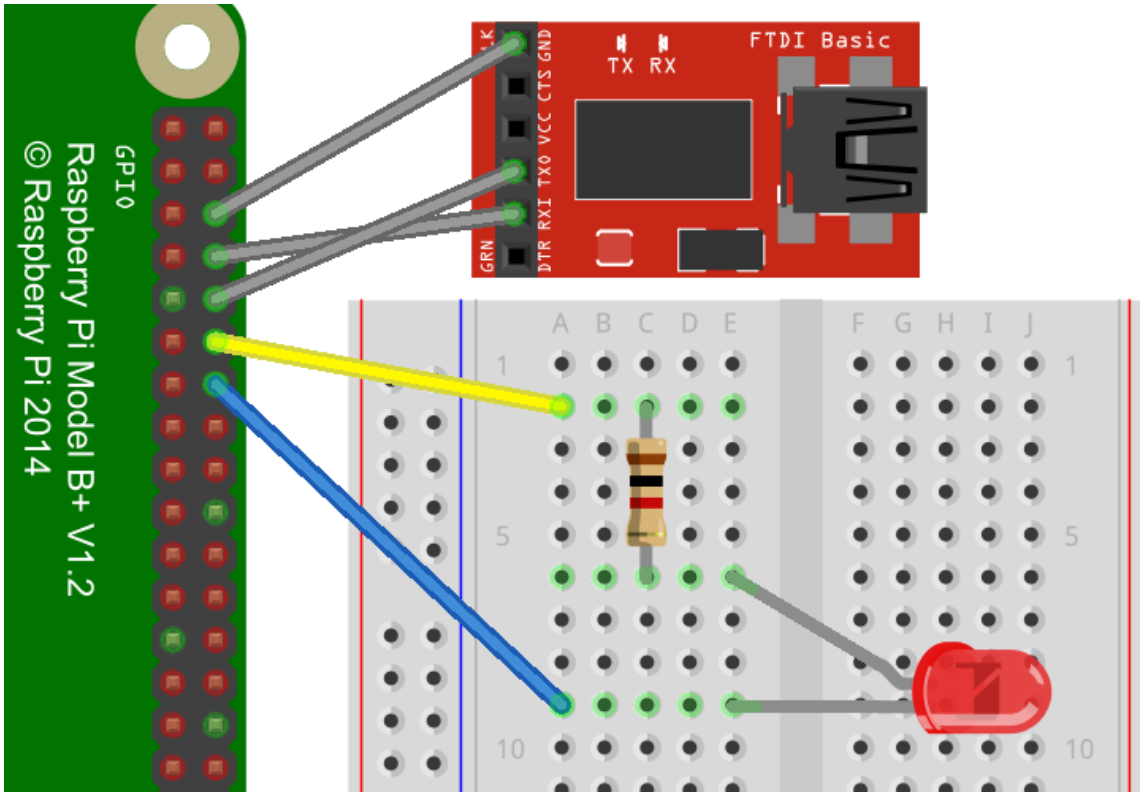
```
$ sudo shutdown -h now
```

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。プログラムを作らず、Raspberry Pi のターミナルに直接打ち込んでください。

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  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/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. Node.js で実行する場合には、OS の特権者として 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。

```
$ sudo  node  handson5.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。

Node.js 内から、writeFileSync を利用して GPIO にアクセスする場合には、OS の特権者の権限が必要です。

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。プログラムを作らず、Raspberry Pi のターミナルに直接打ち込んでください。

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);
```

2. pi-blaster を利用したプログラムを Node.js で実行する場合には、OS の特権者権限は必要ありません。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  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. 今までのハンズオンを参考にして、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。特に何か表示されることはありません。Raspberry Pi をこのままの状態にして、ハンズオン８を行ってください。

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 handson7.js 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。

```
$ cp  handson7.js  handson9.js
```

2. 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');
  });
});
```

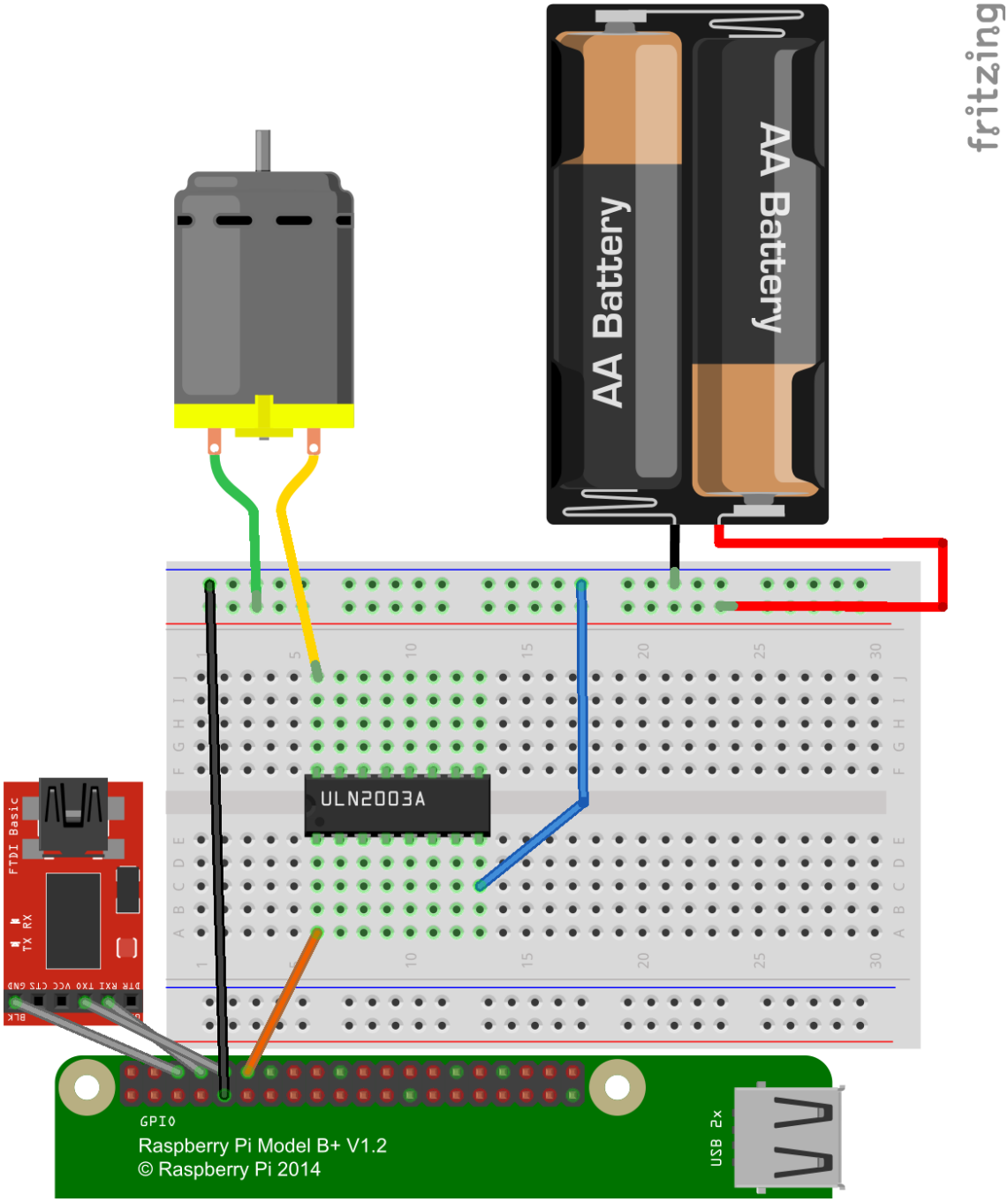
ハンズオン８を参考にして、シミュレータもしくは Firefox OS 端末から FirefoxOS アプリから呼び出します。

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/