★Raspberry Pi 用実験拡張ボード Apple Pi

■Apple Pi 補足資料

トランジスタ技術 2016 年 8 月号で紹介したアップル・パイの補足資料です。紙面数の関係上省略されたり、わかりにくい部分を補足します。

CQ 出版社の特設ページはこちらです。

http://toragi.cqpub.co.jp/tabid/807/Default.aspx

●機能盛りだくさんボード

Apple Pi とは Raspberry Pi で電子工作を楽しむための拡張ボードです。Raspberry Pi で電子工作を始めると必要になる周辺機能を一枚の基板に集約しました。LED、タクトスイッチをはじめ、温度センサー、湿度センサー、気圧センサー、キャラクタ LCD、赤外線送受信、さらにはハイレゾ再生までも盛り込みました。Apple Pi があれば電子工作を楽しめるでしょう。

●命名の裏話

パイといえば一番に思いつくのはアップル・パイだろうということで名づけました。

食べるパイのスペルは Pie です。Pi とは円周率のπであり、欧米人はそこに数学や科学をイメージします。日本人には理解できないニュアンスがあります。

●ご了承その 1

今回は盛りだくさんの機能を狭い基板に押し込めるため、一部にリード部品ではなく表面実装部品を用いました。手間をおかけしますが、コスト削減と多機能化のためご理解をお願いします。

●ご了承その 2

当初、気温、湿度、気圧を気象データとして取得する予定でしたが、Raspberry Pi のバージョンアップに伴い、消費電力が増え、Raspberry Pi が発熱するようになりました。この影響で Raspberry Pi ケース内の気温、湿度データとなります。センサーが熱せられる影響で気温は高くなり、湿度は乾燥します。気圧も温度補償されるため若干影響を受けます。

室内の気温、湿度を取得したいときは、組み立ての際に延長ケーブルを作りセンサーをケースから出してお使いください。

※センサーモジュール内で I2C のプルアップを行っているため、センサーモジュールを取り付けないと LCD も機能しません。

実行例、ラズベリーパイが冷えているときは正確

```
$ sudo ./getBME
31.19C
 99874Pa
54.64%
```

●I2C の相性問題

I2C 方式のキャラクタ LCD で Raspberry Pi から認識しない問題に直面しました。ある型番は動作し、ある型番は動作しないといった具合で相性問題が発生します。しかもロットによって動作したりしなかったりと不安定です。

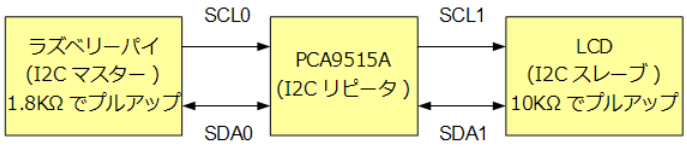
原因を追及すると、I2C スレーブ・デバイス側の電流駆動能力不足であることが判明しました。

Raspberry Pi の I2C 信号は 1.8KΩ で 3.3V にプルアップされています。プルアップ抵抗が 10KΩ であれば発覚しませんでした。

結局 I2C スレーブ・デバイスが I2C の仕様を満たしていないことが原因です。

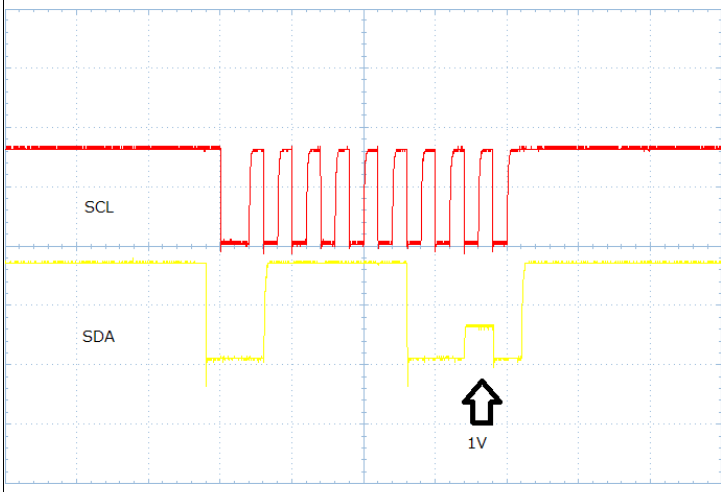
原因がわかれば対処方法は簡単です。今後も同様の問題を抱えた I2C デバイスが登場するとも限りません。そこで I2C リピータ(PCA9515A)を間に入れ、電流駆動能力を補うことにしました。

図 I2C リピータの役割



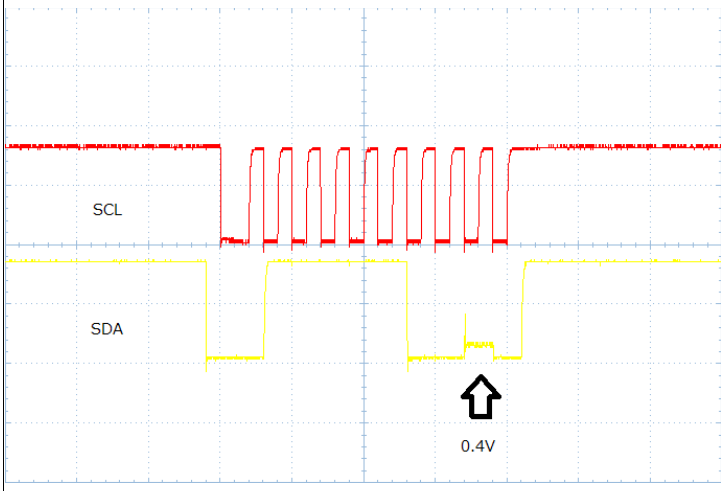
紙面では黄色が薄くなり、確認できないので、改善前と改善後の SDA 信号波形を掲載します。

図 改善前の SCL と SDA 波形



※I2C スレーブの ACK が下がらない

図 改善後の SCL と SDA 波形



※I2C スレーブの ACK が下がった

●R3

R3 は LCD のバックライト用制限抵抗です。これは部品を汎用化するためです。現在は AQM0802 を利用していますが、一時的に在庫切れを起こしました。生産終了する可能性もあります。そこでバックライトありの LCD にも対応しました。

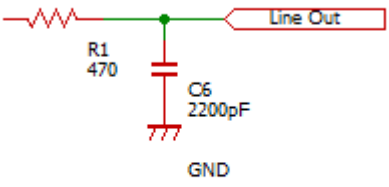
●ハイレゾ DAC

Raspberry Pi 標準のオーディオ出力は特に音量が小さいとノイズが目立ち、音楽として楽しめません。理由はわかりませんがザラザラとノイズが入ります。そこでハイレゾ用 DAC(PCM5102A)を搭載しました。

Raspberry Pi の電源は USB から供給します。USB 電源は激しいスイッチングノイズで知られており、せっかくのハイレゾに電源からノイズを供給したのではもったいありません。そこでノイズフィルタ目的で 3.3V のレギュレータを追加しました。

ラズベリー・パイと PCM5102A は I2S 接続します。ハイレゾ出力はライン出力(2.1Vrms)です。抵抗(470Ω)による分圧効果で、簡易的ではありますがイヤホン(16Ω)やヘッドホン(32Ω)を駆動できます。

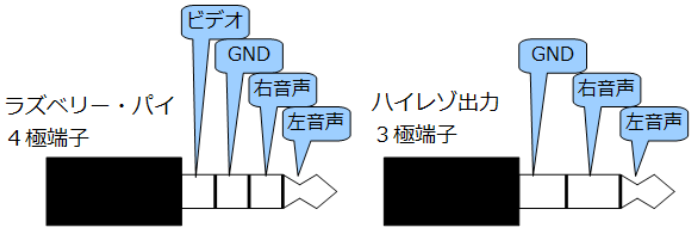
図 ローパスフィルタ



PCM5102Aの出力部

参考までに Raspberry Pi の 3.5mm ステレオ・ミニ・ジャックを解説します。CTIA 規格の４極です。一方でハイレゾ出力は通常の３極です。

図 3.5mm ステレオ・ミニ・ジャック



■あとがき

このたび企画、開発、調整など総合プロデュースしました。CQ 出版社とビット・トレード・ワンさんに多大なご協力をいただき感謝します。

雑誌も製品も発売前にも関わらず、大ヒット作となりました。次回もお楽しみに。

■著作権と免責事項

個人利用に限定され、著作権者の許可なく商用利用できません。直接間接にかかわらず、いかなる損害も筆者は責任を負いません。

■部品セット内容

予告なく変更されることがあります。例えば PCA9515 は NXP や TI 製であったりします。市場の在庫状況によって左右されます。１ピン位置表示はドットであったり、バーであったりします。こうしたことを判断できる上級者向けです。

■組み立て手順

組み立て手順を少し補足します。まずは完成写真をよく確認しましょう。

ケースへ収納のため、部品に高さ制限があります。

ステップ 0

表面実装部品(R1～R6,C1～C9,U5)をピンセットでハンダ付けします。

ステップ 1

LCD をハンダ付けします。

ステップ 2

R7 をハンダ付けします。

ステップ 3

LED1,LED2,IR をハンダ付けします。向きに注意してください。LED の切り欠きを合わせます。

LED の形状は円ではなく、片側が一部平らです。

LED1 は青、LED2 は白です。上から見たとき黄色に見えるのが白です。白 LED は青で発光しており、三原色の原理で、青に黄色を足すことで白になります。

ステップ 4

SW1～SW6 をハンダ付けします。

色の順序は抵抗のカラーコード順です(茶、赤、橙、黄、緑、青)。

ステップ 5

Q1,U1,U2,U3,U4 をハンダ付けします。

U3(赤外線受光)のピンは 90 度に曲げ、寝かせて配置します。

ステップ 6

40 ピンソケットと JK1 を基板の裏にハンダ付けします。

※I2C 端子、UART 端子、電源端子は必要に応じてハンダ付けします。ロープロファイルをご利用ください。

※高さ調節のため右穴だけにスペーサを取り付けます。左穴はケースの突起があるため取り付けできません。ラズベリー・パイと固定はしません。固定するとケースに収まりません。

※高さ制限のため、IC ソケットを使えません。

図 LED の切り欠き

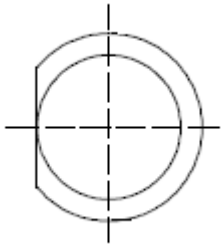
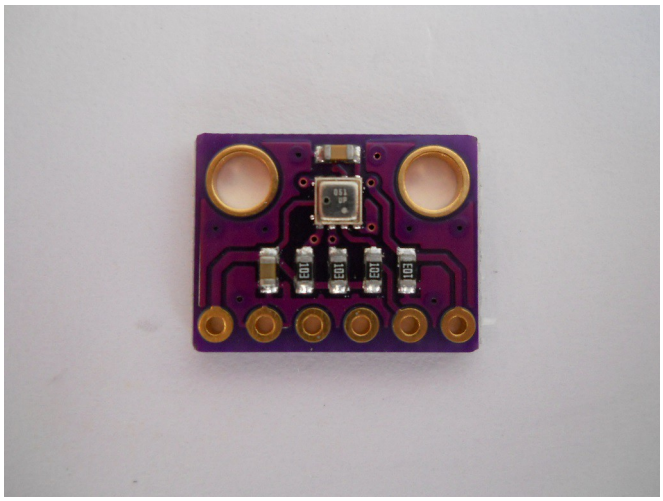


図 BME280 の部品面が表



●ダウンロード方法

記事の ApplePi 用のソースファイル ApplePi.tar のダウンロード URL に誤りがあります。正しくは以下の通りです。

```
$ wget
http://einstlab.web.fc2.com/RaspberryPi/ApplePi.tar
$ tar xvf ApplePi.tar
$ cd ApplePi
$ ./ApplePi.sh
```

setup.sh でソースをコンパイルして実行ファイルを生成します。

●スイッチ制御の注意事項

※スイッチの GPIO 端子を出力設定にしないでください。誤ってボタンを押すとショートしポートを壊すかもしれません。

●aplay コマンドの制限

aplay では 3 バイト単位の 24 ビットファイルを再生できません。4 バイト単位の 24 ビットファイルなら再生できます。

以下はハイレゾ音源を再生できない例です。

```
$ aplay -D hw:1,0 ADAGIO.wav
Playing WAVE 'ADAGIO.wav' : Signed 24 bit Little
Endian in 3bytes, Rate 192000 Hz, Stereo
aplay: set_params:1233: Sample format non available
Available formats:
- S16_LE
- S24_LE
- S32_LE
```

●omxplayer コマンド

omxplayer を使うと 3 バイト単位の 24 ビットファイルも再生できます。omxplayer は mp3 も再生できます。

ただし出力先を PCM5102A に設定できません。omxplayer は ALSA を利用しないためです。

3.5mm ジャック出力する場合

```
$ omxplayer -o local ADAGIO.wav
Audio codec pcm_s24le channels 2 samplerate
192000 bitspersample 24
Subtitle count: 0, state: off, index: 1, delay: 0
```

HDMI 出力する場合

```
$ omxplayer -o hdmi ADAGIO.wav
```

●VLC media player

そこで登場するのが VLC media player です。mp3 を含むさまざなフォーマットに対応しています。ALSA にも対応しています。

インストール方法は簡単です。

```
$ sudo apt-get install vlc
```

再生方法はオーディオ出力先をハイレゾ DAC に変更し、再生ファイルを選択するだけです。

VLC media player にはボリューム機能やイコライザ機能があります。さらにコマンドライン制御もできます。これを利用すればタクトボタンや赤外線受信、あるいはネットワークを使ったリモートコントロール制御が可能です。GUI ではこうした制御をできません。

コマンドラインの例です。

```
$ vlc -I dummy --quiet --aout alsa --alsa-audio-device=hw:1,0 ChistmasEve.mp3 vlc://quit
```

-I dummy でインターフェースをダミーにします。

--quiet で余計な表示を省略します。

--aout alsa --alsa-audio-device=hw:1,0 でオーディオ出力先をハイレゾ DAC にします。

vlc://quit で vlc コマンドを終了します。これを指定しないと vlc コマンドを終了しません。

これをシェル化(play.sh)しました。

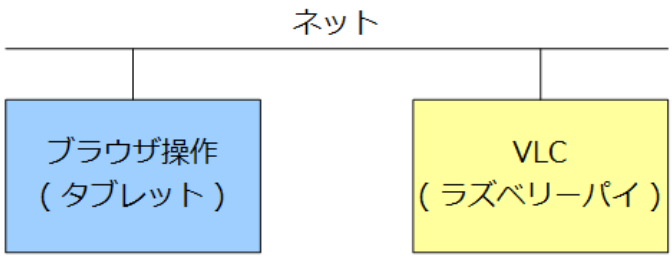
シェルの実行例です。

```
$ ./play.sh ChistmasEve.mp3
```

もちろん VLC の GUI を使ってハイレゾを再生できます。

●VLC のブラウザ・インターフェース

外部の PC やタブレット PC からも VLC を遠隔操作(ネットワーク経由の操作)できます。つまり Raspberry Pi 側にモニタがなくても操作できます。タブレットでリモコン操作します。



ラズベリー・パイ側の設定です。

```
$ vlc --extraintf=http --http-port 8080 --http-password 1234 --aout alsa --alsa-audio-device=hw:1,0
```

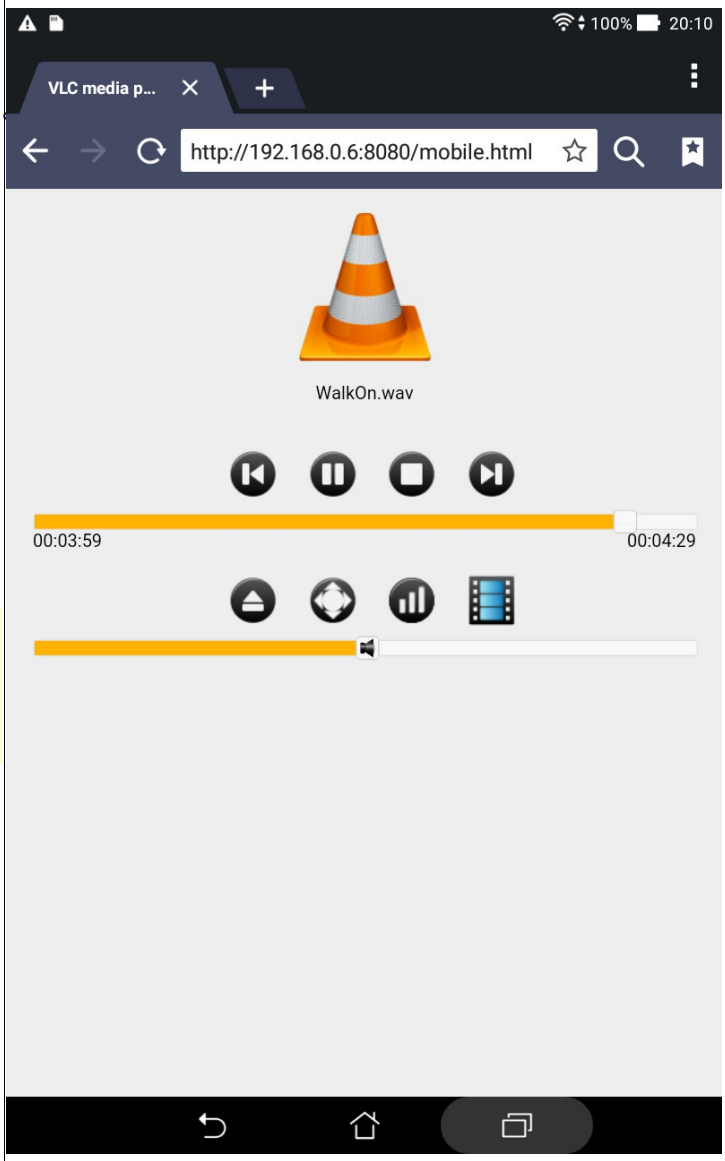
これをシェル化(bvlc.sh)しました。

タブレット側の操作です。

ブラウザでラズベリー・パイの URL(http://192.168.0.6:8080)を指定します。ポート番号は 8080 です。パスワードを聞かれます。名前=なし、パスワード=1234

あとは再生ファイルを選択するだけです。

図 タブレット版ブラウザ・インターフェース



■応用編

Apple Pi の基本的な使い方を理解したところで、応用編です。改良を加えながら、いくつかの事例を紹介します。なお、赤外線コードはリモコン機種によって異なるため、ご自身の環境に合うよう適宜修正してください。

●リモコンでハイレゾ再生

赤外線リモコンを使って Raspberry Pi を制御します。Raspberry Pi が制御される側です。

図 リモコン操作

赤外線送受信のソフトウェアを改良します。bcm2835 ライブラリに割り込み機能はありません。そこで割り込み機能のある WiringPi ライブラリを利用するように書き換えます。赤外線を受信すると割り込みがかかるようにし、永久に待たせます。こうすることで CPU 負荷をかけずにソフトウェアを常駐させることができます。

書き換えたソースが getIR2.c と setIR2.c です。

getIR2.c の抜粋です。

```
int main(int argc, char *argv[])
{

        wiringPiSetup();  // WiringPi の初期化
        pinMode(IROUT,OUTPUT);    // ピンの出力設定
        pinMode(IRIN,INPUT);      // ピンの入力設定
        pullUpDnControl(IRIN,PUD_UP);          // プルアップ設定
        digitalWrite(IROUT,0);

        // to detect falling edge
        wiringPiISR(IRIN,INT_EDGE_FALLING,(void*)getIR);          // 立下りで割り込み設定
        done=0;
        waitForInterrupt(IRIN,-1);    // 割り込み待ち
        while(done==0);// 割り込みスレッド完了待ち
        display(argc);

}
```

WiringPi で割り込みを利用するには少しコツがあります。割り込み関数は別スレッドとして実行されるため、メイン・スレッドと処理を同期させる必要があります。つまり割り込みが発生するとメイン・スレッドと割り込みスレッドが同時に走ります。今回は割り込みスレッドで赤外線データの解析を行い、それが完了してからメイン・スレッドでデータ表示します。メイン・スレッドがデータを受け取る前に割り込みスレッドが終了しても困ります。そのため処理の同期が必要になります。頭を柔らかくし、スレッドが同時並行処理するイメージを思い浮かべましょう。

さらに赤外線コードを判別するため、getIR2 に -h オプションを追加し、ヘキサ形式のコードを出力するようにしました。

getIR2 -h の実行例です。

```
$ sudo ./getIR2 -h
NEC 2038C7906F
```

赤外線リモコンのボタンを判別し、それに応じた処理をする loopIR.sh を作ります。

例えば

(1)ボタン A で WalkOn.wav をハイレゾ再生

(2)ボタン B で One.wav をハイレゾ再生

(3)ボタン C で温度を取得し、温度を LCD 表示

(4)ボタン D で湿度を取得し、湿度を LCD 表示

(5)ボタン E で Raspberry Pi を poweroff

といった具合です。

loopIR.sh の内容です。

```
$ cat loopIR.sh
#!/bin/bash

base=/home/pi/ApplePi

$base/initLCD
$base/locateLCD 0 0
$base/printLCD "PowerOn "



while [ 1 ]
do
S="`sudo $base/getIR2 -h`"
#if [ "${S}" = "AEHA300220800020A0" ]
if [ "${S}" = "NEC 2038C7906F" ]
then
        $base/locateLCD 0 0
        $base/printLCD "WalkOn  "
        aplay -D hw:1,0 $base/../Music/WalkOn.wav &
fi
#if [ "${S}" = "AEHA300220800021A1" ]
if [ "${S}" = "NEC 2038C78F70" ]
then
        $base/locateLCD 0 0
        $base/printLCD "One     "
        aplay -D hw:1,0 $base/../Music/One.wav &
fi
#if [ "${S}" = "AEHA300220800034B4" ]
if [ "${S}" = "NEC 2038C78778" ]
then
        sudo $base/onLED1
        t=`$base/getBME -t`
        $base/locateLCD 0 1
        $base/printLCD $t
        sleep 1
        sudo $base/offLED1
fi
#if [ "${S}" = "AEHA300220800035B5" ]
if [ "${S}" = "NEC 2038C78B74" ]
then
        sudo $base/onLED2
        h=`$base/getBME -h`
        $base/locateLCD 0 1
        $base/printLCD $h
```

```
        sleep 1
        sudo $base/offLED2
fi
#if [ "${S}" = "AEHA30022080003DBD" ]
if [ "${S}" = "NEC 2038C7847B" ]
then
        $base/locateLCD 0 0
        $base/printLCD "PowerOff"
        sudo poweroff
fi
sleep 1
done
```

この loopIR.sh を Raspberry Pi の起動時に自動起動するように仕掛けます。Linux には RC スクリプトと呼ばれる起動の仕組みがあります。

/etc/rc.local の exit 0 直前に loopIR.sh を登録します。

RC スクリプトへの追加場所です。

```
$ sudo vi /etc/rc.local
$ cat /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s¥n" "$_IP"
fi

/home/pi/ApplePi/loopIR.sh &

exit 0
```

これで Raspberry Pi にモニターやキーボード、マウスを接続しなくても赤外線リモコンで操作できます。

USB モバイルバッテリを使えば Raspberry Pi をポータブルにできます。リモコンで poweroff コマンドを発行できるので安心して電源を切れます。

※poweroff 後の USB 電源を切れる状態を示すアクセスランプがバージョンによって異なります。ラズベリー・パイ 2 はアクセスランプが消灯し、ラズベリー・パイ 3 はアクセスランプが点灯します。

●割り込みを使ったスイッチ待ち

getSWx.c ではポーリング待ちしかできませんでした。そこで、割り込み待ちを使った waitSWx.c を作りました。スイッチを押されるまで待ちます。

waitSW6.c の抜粋です。

$ cat waitSW6.c

```
#include <stdio.h>
#include <wiringPi.h>

// SW6 = GPIO27 = J8_13
//#define PIN RPI_BPLUS_GPIO_J8_13
#define PIN 2

void isr()
{
	// nothing to do
}

int main(int argc, char **arg)
{

	wiringPiSetup();
	pinMode(PIN, INPUT);
	pullUpDnControl(PIN, PUD_UP);
	wiringPiISR(PIN,INT_EDGE_FALLING,(void*)isr);
	waitForInterrupt(PIN,-1);

}
```

実行例

```
$ sudo ./waitSW6
```

これを利用すれば、SW6 が押されたら poweroff コマンドを発行するシェルを作れます。

シェル(powerOFF.sh)の例です。

```
$ cat powerOFF.sh
#! /bin/bash
sudo /home/pi/ApplePi/waitSW6
sudo poweroff
```

他にも応用できます。スイッチを押されたら何かをするという場合に役に立ちます。

●demo.sh のスイッチ反応速度改善方法

demo.sh ではスイッチを 0.1 秒間隔でポーリングしています。そのため、0.1 秒の反応遅れを生じることがあります。ポーリング間隔を小さくすると反応速度が改善します。ただしあまり短くすると CPU 負荷が高くなります。さらにチャタリングによる誤動作も生じます。ポーリング間隔はチャタリング防止の意味もあります。

ポーリング間隔は 0.1 秒から 10ms の間で選択します。

```
# wait 0.1sec
sleep 0.1
```

●log.sh の訂正

温度湿度気圧情報を記録し、index.html を生成する log.sh に誤りがありました。

誤:sudo $base/getBME >> $/base/log.txt

正:sudo $base/getBME >> $base/log.txt

●getSW5.sh の訂正

誤:port=27

正:port=26

誤:if [! -e /sys/class/gpio/gpio27]; then

正:if [! -e /sys/class/gpio/gpio26]; then

●HAT

Arduino では拡張ボードをシールド(Shield)と呼ぶように Raspberry Pi では拡張ボードをハット(HAT:Hardware Attached on Top)と呼びます。しかし細かい HAT 規格があり、これに準拠しないと HAT と名乗ることができません。

特にボードを識別するために EEPROM を搭載しなければなりません。また HAT 規格が Raspberry Pi 2 の時に決定されたため、一部 Raspberry Pi 3 で問題になる箇所があります。