# **Raspberry Pi Mouse**

# 取扱い説明書

2.0版
 2016年6月15日
 株式会社アールティ



# 内容

1	安全について
1.1	安全にご使用いただくために
2	ハードウェアについて
2.1	パーツー覧
2.2	名称について
3	RasPiMouseの組立について
4	<b>OS</b> のインストール方法について
5	デバイスドライバのインストールについて14
6	デバイスドライバの使い方について17
6.1	LED の操作17
6.2	ブザの操作17
6.3	モータの操作18
6.4	センサの読み取り19
6.5	タクトスイッチの読み取り19
7	デバイスドライバの使用例21
7.1	Step1 LED を光らせよう21
7.1.	1 Shell バージョン21
7.1.	2 C 言語 バージョン
7.1.	3 Python バージョン25
7.2	Step2 ブザを鳴らそう
7.2.	1 Shell バーション
7.2.	2 C 言語バージョン
7.2.	3 Python バージョン
7.3	Step3 スイッチを使おう31
7.3.	1 Shell バージョン31
7.3.	2 C 言語 バージョン
7.3.	3 Python バージョン
8	備考
8.1	参考文献
8.2	著作権
9	問い合わせ
10	改版履歴



# 1 安全について

### 1.1 安全にご使用いただくために

- Raspberry Pi Mouse をお買い上げいただきましてありがとうございます。ご使用になる前 にこの説明書をよくお読みになり、十分理解した上で作業を始めてください。
- ② 初めてロボットを触る方は、経験者と一緒に作業することをお勧めします。

#### ③ 【注意】ショートについて

センサ、モータ、基板、電池などの端子同士など接触によるショート時にはロボットを壊 すだけではなく、発火の恐れがあります。ケーブルの取り回しの際には各部品に挟まれな いよう余裕をもたせ、ケーブルに傷がある場合は絶対に使用しないでください。また、電 源とグラウンド線の扱いには細心の注意を払ってください。

#### ④ 【重要】電池の関するご注意

Raspberry Pi Mouse は、リチウムイオンポリマ充電池(以下Li-Po電池と表記)を採用してい ます。Li-Po電池を使用するにあたり、必ず正しい知識で取扱いをお願いします。Li-Po電池 は、小さくて軽く、瞬間的に流せる電流もほかの電池に比べて大きく、さらにメモリ効果も ほとんどないためロボット用途には向いています。しかし、他の電池に比べるとやや高価で す、過充電やショートを起こすと発火・爆発することもあります。一般的なLi-Po電池は保 護回路が入っていますが、取扱いには細心の注意が必要です。Li-Po電池はそのCellの数で電 圧の大きさが決まります。1Cell平均3.7V、2Cellで7.4V、3Cellで11.1Vと電圧が大きくなっ ていきます。また、全ての電池の放電に関する単位にCを使います。Cは容量に対する放電 の比率を表し、1Cですと1倍、2Cですと2倍、3Cですと3倍の電流を放電するという意味 になります。

充電する際には専用の充電器を使ってください。(Li-Po電池は1Cで充電するのが良いでしょ う)保管する際には充電容量の約90~100%の状態で保管するのが良いと言われています。ま た、1Cell 3.3V を切ると過放電となり、使えなくなってしまいます。Raspberry Pi Mouseで 使われているLi-Po電池は3Cell なので、10V 以下には絶対しないでください。目安としてモ ータに電源が入っている場合、フル充電から20分ぐらいで10V程度まで電圧が下がります。 10V以下になるとピー・ピーと警告音のブサがなります。ブザがなりましたら、早急に電源 をOFFしてください。Li-Po電池を利用するときは、注意事項をよく守って、過放電、過充 電にならないように取り扱ってください。

万が一、誤った使用により怪我をしたり、火災を起こしたりした場合でもメーカ販売店は責 任を持ちません。

#### 充電について

必ず専用充電器を使用してください。充電中は燃えやすいものが近くにない場所で、目を離 さないように行って下さい。充電完了後は、必ず電池のコネクタを充電器からはずしてくだ さい。またLi-Po電池はメモリ効果がほとんどないため、追加充電可能です。必ず完全に放 電してしまう前に充電してください。



#### <充電中に電池がふくれ始めたら>

万一、充電中に異臭やふくれ始めたのを確認したら、直ちに充電を中止し、コネクタを充電 器からはずしてください。(充電し続けると発火、または爆発の危険があります。)その場 合、電池を外して燃え移るものがない安全な場所で1日程度様子をみてください。再使用は 絶対にしないで廃棄してください。(廃棄方法の項を参照して廃棄処分してください。)

#### 放電について

Raspberry Pi Mouseに使用しているLi-Po電池は過放電をすると電池自体が使用不可能になります。過放電は絶対にしないよう注意してください。また、Raspberry Pi Mouseをご使用後は必ずコネクタを抜いてください。

#### 使用について

本説明書をよくお読みになり、正しくご利用ください。ショート、衝撃、釘刺し等による破 損でも発火、爆発の危険があります。たとえば、電池と鋭利な工具とを一緒に工具箱に入れ る、汗をかいた手でコネクタを触る、誤って水に落とすなど、いろいろな場面が想定されま すが、そのような行為は、絶対にしないでください。

Li-Po電池は、定格電流が決められています。Raspberry Pi Mouseのみでご使用の場合では 最大電流を超えないように設計していますが、各ユニットパーツを自作等のものに取り替え たときやマニュアルに無いような応用回路にしたとき、電池の放電容量を超えないように設 計してください。定格電流以上に電流を流すと、爆発の可能性があります。

#### 保管方法について

保管する際十分な充電を行い、周りに導通するものや、燃えやすいものがないところに保管 してください。

#### 廃棄方法について

海水と同程度の濃度の塩水に2~3日漬けて完全に放電させてから、燃えないごみとして廃 棄して下さい。具体的には水1Lに対して塩30gを入れると海水と同程度の塩水になりま す。

#### Li-Po電池の使用における保証について

Li-Po電池を安全に使用するのはユーザの責任です。メーカおよび販売店は、Li-Po電池の誤 使用によって起こるいかなる対人・対物事故、損害、破損について一切の責任を負いませ ん。安全に管理すれば非常に使いやすい電池です。正しい知識をもってお取り扱いくださ い。



# 2 ハードウェアについて

ここでは、Raspberry Pi Mouseの内容物と各部の名称を確認します。

### 2.1 パーツ一覧

組み立てられた状態で出荷されますので、パーツとしては本体、Raspberry Piと本体をつなぐ コネクタ、RaspberryPi(フルキットのみ)、Debian系OSのRaspbian 入りMicroSDカード(フルキ ットのみ)になります。





# 3 RasPiMouse の組立について

**OS** が入った **SD** カードを図の向きで挿入します。RasberryPi3 では、ロック機能がなくなっています。



RasberryPiをつなぐコネクタと RasberryPiを固定するネジを外し、スペーサの上に RaspberryPi を置きます。



Raspnberry Pi を置いたら、先ほど外したネジで固定してください。



コネクタで RaspberryPi と Raspberry Pi Mouse を接続します。ピンがずれないように入れてください。





# 4 OS のインストール方法について

Raspberry Pi は Raspbian、Ubuntu(ROS)、RTM にて動作確認ができていますが、ここでは Raspberry Pi 公式の OS Raspbian のインストールを行う手順を示します。Raspbian の OS は https://www.raspberrypi.org/downloads/

からダウンロードして SD カードに書き込む必要があります。

Windows 上での作業方法を掲載します。

DOWNLOADS		
Raspbian is the Foundation's offi here, or use NOOBS, our easy ins	cial supported Operating Systen taller for Raspbian and more.	n. Download it
NOOBS	RASPEIAN	

ダウンロードサイトには上記の2種類(NOOBSとRASPBIAN)あります。RASPBIANの方は、イ メージファイルになります。ダウンロードしてSDカードに書くにはイメージを書くソフトが必要 になります。NOOBSは、ダウンロードし、展開したデータをそのままSDカードにコピするだけ です。初心者にはNOOBSをお勧めします。ここでは、NOOBSでのRASPBINのインストール方 法を示します。NOOBSをクリックしていただくと下記の画面になります。





NOOBS(NOOBS LITE ではない方)の Download.ZIP をクリックしローカルに保存します。

この ZIP を展開(windows7 以降なら右クリックし"すべて展開"で展開できます。)し、SD カード にコピします。SD カードは 16GB 以上をお勧めします。デバイスドライバを RaspberryPi 上でコ ンパイルする環境を整備すると 8GB 必要です。



SD カードに OS の基となるデータを書き終えたら、3 章に示したように Raspberry Pi Mouse を 組み立てます。組み立てましたら、Raspberry Pi に USB のマウスと USB のキーボードと HDMI の モニタを接続します。





上記の接続例では、電池の代わりにACアダプタで接続しています。(フルキットに含まれていま す。RaspberryPiMouse 単品で購入しACアダプタがほしい方は、オプションで別途購入できま す。Lipo 充電器 LBC-010[AC アダプタ・変換ケーブル付)6500 円(税抜き)] 電池で知期環境を作る際はフル本電した電池をご使用ください。マウストキーボードト HDMLのコ

電池で初期環境を作る際はフル充電した電池をご使用ください。マウスとキーボードと HDMI のモ ニタを接続しましたら、MAIN SW を ON にしてください。MOTOR SW は OFF にしてください。 OS が起動する前で MOTOR\_SW を ON にするとモータに電流が流れます。電池で環境整備をして いる場合、環境の設定が終わる前に電圧が 10V 以下になる可能性があるため、環境整備の時は必 ず、MOTOR\_SW は OFF にしてください。

デバイスドライバは github からダウンロードしますので、LAN ケーブルを挿入するか無線 LAN で接続できるようにしておいてください。



RECOMMENDED				
ntv-created Dort	of Debian jess	ie for the Ra	spberry Pi	
Rasphian	をクリック	7		
raopolari				
			1 heading	Ath All
	Raspbian	Raspbian をクリック	Raspbian をクリック	Raspbian をクリック

MAIN SW を ON し、しばらくすると上記の画面が表示されます。Raspbian をクリックし、Install ボタンをクリックします。Raspbian をクリックしないと Install ボタンが押せません。

127		NOOBS v1.9 - E	Built: Mar 18 20	16			
Install (i)	Edit config (e)	U Wifi networks (w)	Online help (h)	Exit (Esc)			
×C	Raspbian [RECOMMENDED] A community-created port of Debian jessie for the Raspberry Pi						
8		Co	nfirm		×		
Warning: this will install the selected Operating System(s). All existing data on the SD card will be overwritten, including any OSes that are already installed.							
	an eady miscain	Yes	] <u>№</u> о				
Disk spa		Yes	) <u>N</u> o				

Install ボタンを押すと確認の画面が表示されますので"Yes"をクリックしてください。





"Yes"を押すとインストールが始まりますのでしばらくお待ちください。Raspberry Pi3 で 15 分ぐらいです。



インストールが終わると上記が表示されますので、**"OK"**ボタンをクリックします。クリックすると 自動的に再起動します。



再起動すると GUI が立ち上がります。パスワードや user の問い合わせはありません。



**Raspberry Pi Mouse** は SPI を使用するため SPI の機能をイネーブルにします。 Menu の Preferences の Raspberry Pi Configuration をクリックします。





Raspberry Pi Configuration が立ち上がりますので、Interfaces のタブをクリックします。SPIの Enable をクリックし、"OK"ボタンをクリックします。



OS 入りの SD カートには、ここまで設定されたものが入っています。環境整備が面倒な方は Raspberry Pi Mouse フルキットをお勧めします。

# 5 デバイスドライバのインストールについて

デバイスドライバはGitを使って「GitHub」からダウンロードします。Gitはターミナルにコマンドを打ち込んで使用します。GUIからターミナルの起動は以下のようにして起動します。 Terminalをクリックすると右の画面が表示されます。



デバイスドライバのインストールはSSHでログインしても可能です。SSHでログインするには、 初期ユーザ名 pi とパスワード raspberryを入力してください。

デバイスドライバは、GitHubからダウンロードしたファイルの「./lib/RaspberryPiのバージョン/ カーネルのバージョン」にあるので、「uname –r」でカーネルのバージョンを調べ、カーネルの バージョンを合わせてご使用ください。



Gitのクローンをします。(もしRaspberryPiMouseのディレクトリが既にある場合削除しておく)

git clone https://github.com/rt-net/RaspberryPiMouse.git

Gitからパスワードを聞かれた場合は、raspberryと入力してください。予め「sudo su -」でrootに なるとパスワードが聞かれなくなります。



クローンすると以下のファイル構造が存在していると思います。

RaspberryPiMouse
<b>H</b> lib
│
└3.18.14+など
III Lrtmouse.ko
I └Pi2B+
┃ <b>└</b> 4.4.13-v7+など
Lrtmouse.ko
- LICENSE
- HREADME.md
– Fsrc
I HMakefile
Hakefile.rasbian
Makefile.ubuntu14
L Lrtmouse.c
I = 1201507
I I +Drive circuit4 ipg
$    L = 1000 \pm 1000 \pm 1000 \pm 10000 \pm 10000 \pm 10000 \pm 10000 \pm 10000 \pm 100000 \pm 100000000$
<b>I -</b> 201508
I BPiMouse-schematic pdf
L LMEMO raspi-config rst

build\_install.raspbian.bash
 build\_install.ubuntu14.bash

RaspberryPiMouse/lib/RaspberryPiのバージョン/カーネルバージョン/rtmouse.koがドライバ本体です。このファイルは「カーネルモジュール」といって、カーネルの一部として動作するプログラムの集まりです。また、SPIをセンサで使用するのでSPI機能を有効にしてください。4章参照。

カーネルバージョン4.1.19以降のデバイスドライバでは、RaspberryPi2とRaspberryPi3共通で使用できます。RaspberryPi3でデバイスドライバの組み込みはPi2+のデバイスドライバをご使用ください。

デバイスドライバの組み込みは "insmod"を使用します。"Ismod"というコマンドを打つと組み込 み済みのカーネルモジュールがリストされるので、その中にrtmouseあることを確認してくださ い。デバイスドライバが組み込まれたときにブザかなり、センサが光ります。



インストール
\$sudo insmod rtmouse.ko쉬
メッセージが出なければ成功
RaspberryPi のバージョン
または、
\$sudo insmod /home/pi/RaspberryPiMouse/lib/Pi?B+/`uname -r`/rtmouse.ko⊲⊐
インストールされているかの確認 カーネルのバージョン
\$Ismod   grep_rtmouse⊲ <sup></sup>
rtmouse 12664 0

インストールがうまくいったら、下記のように、/dev/のディレクトリの下にrt<なんとか>と いうファイルができているはずです。/dev/の下にあるファイルは「デバイスファイル」とい われるもので、機械と入出力をつかさどるファイルです。

\$ls /dev/rt*⊲⊐	
/dev/rtbuzzer0	/dev/rmotor_raw_r0
/dev/rtled0	/dev/rtmotoren0
(略)	

パーミッションを変更しておくことでルートにならずにデバイスドライバを扱うことができます。参考までに、ドライバをアンインストールする場合は、「rmmod」を使用します。

\$sudo chmod 666 /dev/rt\*纪 \$ls -I /dev/rt\*纪 crw-rw-rw- 1 root root 245, 0 4月 27 14:38 /dev/rtbuzzer0 crw-rw-rw- 1 root root 246, 0 4月 27 14:38 /dev/rtled0 crw-rw-rw- 1 root root 246, 1 4月 27 14:38 /dev/rtled1 (以下略)



## 6 デバイスドライバの使い方について

デバイスドライバでできることは、LEDの点灯、ブザの発信、光反射型距離センサの値の取得、モータを回す、スイッチの on/off の確認ができます。

### 6.1 LED の操作

LED は 0~3 まであり、下記に示すコマンドは例として LED0 の点灯/消灯を示します。 LED1 の場合は、rtled1 で点灯/消灯できます。最後の数字を 0~3 に変更することで LED の 点灯個所が変わります。

点灯	
	\$ echo 1 > /dev/rtled0⊲ <sup>_</sup>
消灯	
	\$echo 0 > /dev/rtled0<┚

図 6-1 LED 操作

### 6.2 ブザの操作

echo で周波数(単位:Hz)を渡します。

440Hz を鳴らす \$ echo 440 > /dev/rtbuzzer0型

消音する

\$ echo 0 > /dev/rtbuzzer0々┘

図 6-2 Buzzer 操作

参考までに音階の周波数を表1に示します。

小数点が使えません(0と正の整数のみ)ので、切り上げ、切り下げはユーザの判断でお願いします。

表1 音名(階名)と周波数一覧



オクターブ	音名 <b>(</b> 階	名)										
	C(ド)	ド#	D(レ)	レ#	E(ミ)	<b>F(</b> フ	ファ#	G(ソ)	ン#	A(ラ)	ラ#	B(ジ)
						ア)						
0	16.35	17.32	18.35	19.45	20.60	21.83	23.12	24.50	25.96	27.50	29.14	30.87
1	32.70	34.65	36.71	38.89	41.20	43.65	46.25	49.00	51.91	55.00	58.27	61.74
2	65.41	69.30	73.42	77.78	82.41	87.31	92.50	98.00	103.8	111.0	116.5	123.5
3	130.8	138.6	146.8	155.6	164.8	174.6	185.0	196.0	207.7	220.0	233.1	246.9
4	261.6	277.2	293.7	311.1	329.6	349.2	370.0	392.0	415.3	440.0	466.2	493.9
5	523.3	554.4	587.3	622.3	659.3	698.5	740.0	784.0	830.6	880.0	932.3	987.8
6	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976
7	2093	2217	2349	2489	2637	2794	2960	3136	3322	3520	3729	3951
8	4186	4435	4699	4978	5274	5588	5920	6272	6645	7040	7459	7902
9	8372	8870	9397	9956	10548	11175	11840	12544	13290	14080	14917	15804

### 6.3 モータの操作

モータを回すには、ハードウェアスイッチ(MOTOR SW を ON)と、ソフトウェアスイッチ (/dev/rtmotoren0)を両方 ON にして初めてモータを回すことができます。ブログラムがミス なくソフトウェアスイッチを ON/OFF できればハードウェアスイッチを付ける必要はない のですが、Raspberry Pi の暴走、ソフトウェアのバグなどによりソフトウェアスイッチで OFF できない場合、強制的に停止できるよう、ハードウェアスイッチを付けてあります。

ソフトウェアスイッチを ON \$ echo 1 > /dev/rtmoteren0 左のモータのみを順方向に 400Hz で回す \$ echo 400 > /dev/rtmotor\_raw\_I0 逆方向に 400Hz で回す \$ echo -400 > /dev/rtmotor\_raw\_I0 右のモータのみを順方向に 250Hz で回す \$ echo 250 > /dev/rtmotor\_raw\_r0 回転を止める \$ echo 0 > /dev/rtmotor\_raw\_r0

\$echo 0 > /dev/rtmotor\_raw\_lo

図 6-3 モータの操作(1/2)



左右のモータを与えた周波数で与えた時間回す 左 500Hz、右 10Hz、1 秒間回す。時間の単位は ms 左右時間 \$echo 500 10 1000 > /dev/rtmotor0 ソフトウェアスイッチを OFF \$ echo 0 > /dev/rtmoteren0↔

図 6-3 モータの操作(2/2)

入力できる数字は整数のみです。

Raspberry Pi Mouse に使用されているモータはステッピングモーターというもので、1パルスで 0.9[deg]回り、400パルスで一回転します。echoの後の数字は周波数を入力するようになっており、400の場合は、1秒間に400パルス入力され、丁度1秒間に一回転することになります。

#### 6.4 センサの読み取り



\$ cat /dev/rtlightsensor0↩
187 189 184 191
第 1、第 2、第 3、第 4 の順番でセンサの
値が表示されます。

図 6-4 基板についている四つの可視光 LED

### 6.5 タクトスイッチの読み取り

タクトスイッチは0~2まであり、下記に示すコマンドは例として SW0 が押されたか押されて いないかを示します。

タクトスイッチ1の場合は、rtswitch1で最後の数字を0~2に変更することで各々のタクトス



イッチを状態を取得できます。

押されている状態 \$ cat /dev/rtswitch0ゆ 0 押されていない状態 \$cat /dev/rtswitch0ゆ 1

図 6-5 スイッチの操作



# 7 デバイスドライバの使用例

6章でデバイスドライバの簡単な使い方を記述しましたが、この章では、Shell、C言語、 pythonの3種類で具体的なプログラムを提示し、解説します。

### 7.1 Step1 LED を光らせよう

Step1 は表示用の LED を光らせてみます。LED を 0.5 秒ごとに点滅させるプログラム例を以下に示します。

### 7.1.1 Shell バージョン

#!/bin/ while t do	bash rue	
	echo 1   tee /dev/rtled?	#点灯
	sleep 0.5	#0.5 秒待ち
	echo 0   tee /dev/rtled?	#消灯
	sleep 0.5	#0.5 秒待ち
done		

図 7-1 step1 shell バージョン

#### 動作確認と解説

上記のプログラムを step1.sh というファイルに保存し、実行の権限を追加(chmod +x step1.sh)し「\$ ./step1.sh」で実行すると LED が点滅します。終了するには「Ctrl + C 🕘」で終了します。

無限に繰り返すコマンドとして while を使用しています。 while 文の書式

while	条件式
do	
	処理
done	

図 7-2 while の書式



条件が真の場合 do~done の間の処理を繰り返し実行します。"true"の代わりに「:」のヌルコマンドを指定しても無限ループになります。

デバイスドライバにデータを送るコマンドとして echo を使用しています。echo コマンドは、 引数に指定された文字列や変数を表示します。

#### echo 1 > /dev/rtled0

とすれば、デバイスドライバ LED0 に"1"が送られ、LED が点灯します。"0"を送ると消灯しま す。「>」はリダイレクションと呼ばれるもので、左のデータを右のファイル(デバイスドライ バ)に新規に書き込みます。追加で書き込む場合は「>>」になります。ここの例題では「>」を 使わずに tee コマンドと「?」(任意の1文字のワイルドカード、文字列の場合は「\*」)を使用 して LED0 から LED3 まで書き込むようにしています。tee を使用するには、コマンドを連結 するパイプ「|」を使用します。パイプを使用すると、コマンドは左から実行され、コマンド の実行結果を次のコマンドに渡されます。「echo 1」で発行された「1」が「tee /dev/rtled?」 に渡されます。「tee」はコマンドの結果をファイル(今回はデバイスドライバ)に出力しながら 標準出力をするときに使用します。

sleep は指定した時間だけ停止します。記号なしの指定時間の単位は「秒」です。



```
7.1.2 C 言語 バージョン
```

```
#include "fcntl.h"
void main(void)
{
int led[4];
int i;
led[0] = open("/dev/rtled0",O_WRONLY);
led[1] = open("/dev/rtled1",O WRONLY);
led[2] = open("/dev/rtled2",O_WRONLY);
led[3] = open("/dev/rtled3",O_WRONLY);
while(1)
{
       for(i=0;i<4;i++)
       {
               write(led[i],"1",1);
       }
       usleep(500*1000);
       for(i=0;i<4;i++)
       {
               write(led[i],"0",1);
       }
       usleep(500*1000);
}
       for(i=0;i<4;i++)
       {
               close(led[i]);
       }
}
```

#### 図 7-3 step1 C 言語バージョン

動作確認と解説

上記のプログラムを step1.c というファイルに保存し、「\$gcc step1.c -o step1」でコンパイ ルします。「-o」をしない場合は、「a.out」になります。コンパイルで作った実行ファイルを 「\$./step1」で実行すると LED が点滅します。終了するには「Ctrl + C 句」で終了します。 C 言語でファイルの読み書きに open、close 関数を使用します。ファイルのアクセスに限ら ず、デバイスドライバにも使用できます。open、close 関数を使用するには "fcntl.h" が必要に



なるため include します。 無限に繰り返すコマンドして while を使用しています。

while 文の書式



図 7-4 while の書式

条件が真の場合 { }の間の処理を繰り返し実行します。似たようなコマンドで「do { } while(条 件式);」がありますが、「while(条件式) { }」とは違い条件式が false でも一度は実行されると いう仕様になります。つまり、「while(false) { }」では { } 内を一度も実行されないが、「do { } while(false);」では一度点滅が実行されます。

while(0){	do{
write (led0,"1",1);	write (led0,"1",1);
usleep(500 * 1000);	usleep(500 * 1000);
write (led0,"0",1);	write (led0,"0",1);
usleep(500 * 1000);	usleep(500 * 1000);
}	} while(0)

図 7-5 LED0 が一度も点滅しない。 図 7-6 LED0 が一度だけ点滅する。

デバイスドライバに出力するコマンドして write コマンドを使用しています。 write の書式は

int write(int fd, void \*buf, unsigned int n);

図 7-7 wrtie の書式

int fd:データを書き込むファイルを指定します。この例では、デバイスドライバになります。 void \*buf:書き込むデータのアドレスを指定します。この例では、直接データを書き込んでいます。 unsigned n:データの大きさを指定します。この例では1文字を出力するので1としています。

fwrite でもデバイスドライバに書き込むことが可能ですが、fwrite の場合、内部の buffer を経由 するため、buffer に溜まる または close されるまで出力されません。

C 言語には sleep 関数がありますが、小数点が使えず、マイクロ秒[ $\mu$  s]の usleep を使って 0.5 秒の sleep を実現しています。単位は、10<sup>-3</sup>が ms、10<sup>-6</sup>が $\mu$ s で $\mu$ s から ms にするには 1000 倍する必要があり、0.5秒 = 500ms なので、500 \* 1000 で 0.5 秒となります。



### **7.1.3 Python** バージョン

#!/usr/bin/python
import time
import time
import sys

files = [ "/dev/rtled0", "/dev/rtled1", "/dev/rtled2", "/dev/rtled3"]

while 1:
 for filename in files:
 with open(filename, 'w') as f:
 f.write("1")

 time.sleep(0.5)
 for filename in files:
 with open(filename, 'w') as f:
 f.write("0")
 time.sleep(0.5)

図 7-8 step1 python バージョン

動作確認と解説

上記のプログラムを step1.py というファイルに保存し、「\$python step1.py」で実行すると LED が点滅します。終了するには「Ctrl + C 🕘」で終了します。

デバイスドライバに書き込むコマンドとして write を使用しています。Write 単体で使用すると バッファに入るため、ある程度データの書き込みをしないとデバイスドライバに書いてくれま せん。with を使うことで自動的に close をしてくれるので、バッファに溜まらずデータがすぐ に反映されます。Sleep 関数は time に含まれているため time を import しています。

C 言語では while 内で処理するものとして「{}」を使用していましたが、Pythson は「{}」を 使用せず、インデントで制御します。



# 7.2 Step2 ブザを鳴らそう

デバイスドライバを使用すると周波数をデバイスドライバに送るだけで音が鳴ります。キー ボードをたたいてピアノのように音階を出すプログラム例を以下に示します。 キーボードと音階の配置の仕様は以下のようになっています。



図 7-9 step2 音階

# 7.2.1 Shell バーション

step2.sh

図 7-10 step2 shell バージョン(1/2)



SCALE ファイル

off 0 0
ド <b>261 a</b>
ド <b># 277 w</b>
∠ 293 s
∨# 311 e
ે 329 d
ファ 349 f
ファ# 370 t
ン 392 g
ン# 415 y
ラ 440 h
ラ# 466 u
シ 493 j
ド <b>523 k</b>

図 7-11 step2 shell バージョン(2/2)

動作確認と解説

上記のプログラムを1つ目を step2.sh、2つ目を SCALE というファイル名で保存し、実行 の権限を追加し「\$./step2.sh」で実行すると、キーボートを叩くと音が鳴ります。

Shell には連想配列がないため、awk と組み合わせて連想配列を実現しています。While の後の「read – N 1 b」はキーボードから 1 文字読んで変数 b に代入しています。「-N 数字」のオプションで変数に入れる文字数を指定できます。「-N 数字」のオプションがない場合は、リターンを押すまで文字数が変数に入力されます。



### **7.2.2 C**言語バージョン

```
#include "fcntl.h"
int _Getch(void)
{
Int ch:
       system("stty -echo -icanon min 1 time 0");
       ch = getchar();
       system("stty echo icanon");
       return ch;
}
void main(void){
int buzzer= open("/dev/rtbuzzer0" , O_WRONLY );
int c=1;
while(c){
     switch(_Getch()){
          case '0'://off
               write (buzzer,"0",2);
               break:
          case 'a'://do
               write(buzzer,"261",4);
               break;
          case 'w'://do#
               write(buzzer,"277",4);
               break;
          case 's'://re
               write(buzzer,"293",4);
               break;
          case 'e'://re#
               write(buzzer,"311",4);
               break;
          case 'd'://mi
               write(buzzer,"329",4);
               break;
          case 'f'://fa
               write(buzzer,"349",4);
               break;
```

図 7-12 step2 C 言語バージョン(1/2)





図 7-12 step2 C 言語バージョン(2/2)

動作確認と解説

上記のプログラムを step2.c というファイル名で保存し、「\$gcc step2.c –o step2」でコン パイルします。「\$ step2」で実行し、キーボードを叩くと音がなります。

C言語のgetchar コマンドはリターンキーを叩いたときに入力データが変数に更新されるため、音階に指定したキーボードを叩くだけでは音がなりません。そこで、system 関数を使って Linux上でエコーバックをしない(-echo)、特殊文字を無効にする(-icanon)、-canon が設定され ているとき、最小文字数が読み込まれなかった場合に時間切れにするタイム(time N)、-canon が設定されているとき、最低 N 文字(min N)を設定しています。これにより、一文字入力された らすぐに変数に反映されます。

#### RT CORPORATION

### **7.2.3 Python** バージョン

```
#!/usr/bin/python
import sys
class _Getch:
  def __init__(self):
     import tty, sys
  def __call__(self):
     import sys, tty, termios
     fd = sys.stdin.fileno()
     old_settings = termios.tcgetattr(fd)
     try:
        tty.setraw(sys.stdin.fileno())
        ch = sys.stdin.read(1)
     finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
     return ch
dict= { "0":"0", "a":"261", "w":"277", "s":"293","e":"311","d":"329","f":"349","t":"370",
"g":"392","y":"415","h":"440","u":"446","j":"493","k":"523"}
while 1:
     getch = _Getch()
     d = getch()
     if d == "c" :
          break
     if d in dict :
           with open('/dev/rtbuzzer0','w') as f:
                   f.write(dict[d])
```

図 7-13 step2 python バージョン

動作確認と解説

上記のプログラムを step2.py というファイル名で保存し、「\$python step2.py」で実行し、 キーボードを叩くと音がなります。終了するには「Ctrl + C 🕘」または、"c"を入力すると終了 します。

python には、連想配列(ディクショナリ)があり、それを使用することですっきりしたプログ ラムになります。Python にはキーボードの入力を取り込む「raw\_input()」コマンドがあります が、「raw\_input()」コマンドはキーボードをたたいた後リターンキーのを押さないとデータの 取り込みができない仕様となっているため、\_Getch 関数を作り、キーボードをたたいた後すぐ にデータの取り込みができるようにしています。



# 7.3 Step3 スイッチを使おう

スイッチ(SW0)を押すと LED3 を点灯/消灯を繰り返し、スイッチ(SW1)を押すと LED1、 LED2 の点灯/消灯を繰り返し、スイッチ(SW2)を押すと LED0 を点灯/消灯を繰り返すプログラ ムを以下に示します。

#!/bin/ba	sh	
state0=0 state1=0 state2=0		
while true	e; do	
if gr	ep -q 0 /dev/rtswitch0 ; then	
-	sleep 0.1	
	while grep -q 0 /dev/rtswitch0 ; do	
	sleep 0.1	
	done	
	state0=`expr \$state0 + 1`	
	state0=`expr \$state0 % 2`	
	echo \$state0 > /dev/rtled3	
fi		
if grep -q 0 /dev/rtswitch1 ; then		
	sleep 0.1	
	while grep -q 0 /dev/rtswitch1 ; do	
	sleep 0.1	
	done	
	state1=`expr \$state1 + 1`	
	state1=`expr \$state1 %2`	
	echo \$state1 > /dev/rtled2	
	echo \$state1 > /dev/rtled1	
fi		

### 7.3.1 Shell バージョン

図 7-14step3 shell バージョン(1/2)



```
if grep -q 0 /dev/rtswitch2 ; then
sleep 0.1
while grep -q 0 /dev/rtswitch2 ; do
sleep 0.1
done
state2=`expr $state2 + 1`
state2=`expr $state2 % 2`
echo $state2 > /dev/rtled0
fi
done
```

図 7-14step3 shell バージョン(2/2)

動作確認と解説

上記のプログラムを step3.sh というファイル名で保存し、実行の権限を追加し 「\$./step3.sh」で実行すると、タクトスイッチ(SW0~SW2)を押すと LED が光ります。タク トスイッチが押されると"0"が入力されます。押されていない状態では"1"が入力されます。if grep -q 0 /dev/rtswitch??で 0 が入力されたあと sleep 0.1 で少し待っています。これは、機械 的スイッチを押したとき/離したときに内部の金属が震えている状態になります。この状態をチ ャタリングといいます。チャタリング時にスイッチの状態を調べると"0"か"1"のいずれの値も取 れてしまうため、押したのか離したのかが分かりません。しかし、チャタリングは、数μsか ら数 ms 待つことによって固定されますので、sleep 0.1 でチャタリングが終わった後スイッチ の状態を確認しに行っています。

### **7.3.2 C**言語 バージョン

```
#include "fcntl.h"
char get_SW0(void){
    char buf[2];
    int SW0;
    SW0 = open("/dev/rtswitch0",O_RDONLY);
    read(SW0,buf,2);
    close(SW0);
    return buf[0];
}
char get_SW1(void){
    char buf[2];
```

```
int SW1;
```

図 7-15 step3 C 言語バージョン(1/3)



```
SW1 = open("/dev/rtswitch1",O_RDONLY);
    read(SW1,buf,2);
    close(SW1);
    return buf[0];
}
char get_SW2(void){
    char buf[2];
    int SW2;
    SW2 = open("/dev/rtswitch2",O_RDONLY);
    read(SW2,buf,2);
    close(SW2);
    return buf[0];
}
void main(void){
    int state0, state1, state2;
    int LED0, LED1, LED2, LED3;
    LED0 = open("/dev/rtled0",O_WRONLY);
    LED1 = open("/dev/rtled1",O_WRONLY);
    LED2 = open("/dev/rtled2",O_WRONLY);
    LED3 = open("/dev/rtled3",O_WRONLY);
    state0 = state1 = state2 = 0;
    while(1){
         if (get_SW0() =='0'){
              usleep(10000);
              while (get_SW0() == '0');
              usleep(10000);
              state0=(state0+1)&0x01;
              if(state0==0){
                   write(LED3,"0",1);
              }else{
                   write(LED3,"1",1);
              }
         }
```

図 7-15 step3 C 言語バージョン(2/3)





図 7-15 step3 C 言語バージョン(3/3)

動作確認と解説

上記のプログラムを step3.c というファイル名で保存し、「\$gcc step3.c –o step3」でコン パイルします。「\$ Step3」で実行し、スイッチを押すと LED が光ります。 /dev/rtswitch??からは、2 バイトの情報が送られてきます。1 つ目はスイッチの状態の"0"また は"1"、2 つ目は改行コードです。read で 2 バイト受信し、1 つのデータを使用しています。 main 関数内で、スイッチの入力を記述すると見にくくなるため関数化しています。



### **7.3.3 Python** バージョン

```
#!/usr/bin/python
import time
state0=state1=state2=0
while 1:
     with open("/dev/rtswitch0","r") as f:
          if f.readline() == "0\n" :
                time.sleep(0.1)
                while 1:
                     with open("/dev/rtswitch0","r") as f:
                           if f.readline() != "0 \ n":
                                break
                time.sleep(0.1)
                state0 = (state0 + 1 ) & 1
                with open("/dev/rtled3","w") as f:
                     f.write(str(state0))
     with open("/dev/rtswitch1","r") as f:
          if f.readline() == "0\n" :
                time.sleep(0.1)
                while 1:
                     with open("/dev/rtswitch1","r") as f:
                           if f.readline() != "0 \ :
                                break
                time.sleep(0.1)
                state1 = (state1 + 1 ) & 1
                with open("/dev/rtled2","w") as f:
                     f.write(str(state1))
                with open("/dev/rtled1","w") as f:
                     f.write(str(state1))
     with open("/dev/rtswitch2","r") as f:
          if f.readline() == "0\n":
                time.sleep(0.1)
                while 1:
                     with open("/dev/rtswitch2","r") as f:
                           if f.readline() != "0\n" :
                                break
                time.sleep(0.1)
                state2 = (state2 + 1 ) & 1
                with open("/dev/rtled0","w") as f:
                     f.write(str(state2))
```

図 7-16 step3 python バージョン



動作確認と解説

上記のプログラムを step3.py というファイル名で保存し、「\$python step3.py」で実行し、 スイッチを押すと LED が光ります。Python は数値を文字列に変換する"str"の組み込み関数が あり、それを使用して state の値をそのまま LED に出力しています。



## 8 備考

### 8.1 参考文献

日経 Linux 2015 年 6 月号~10 月号「Raspberry Pi で始めるかんたんロボット製作」 日経 BP ラズパイマガジン 2016 年春号

### 8.2 著作権

本取扱い説明書で紹介、または記載されている会社名、製品名は、各社の登録商標または商標です。

本取扱い説明書に掲載されている文章、写真、イラストなどの著作物は、日本の著作権法及び 国際条約により、著作権の保護を受けています。インターネット等の公共ネットワーク、構内 ネットワーク等へのアップロードなどは株式会社アールティの許可無く行うことはできませ ん。

# 9 問い合わせ

本製品に関するお問い合わせは、下記までお願いします。

株式会社 アールティ 〒101-0021 東京都千代田区外神田 3-2-13 山口ビル 3F Email:shop@rt-net.jp URL: http://www.rt-net.jp



# 10改版履歴

版数	改版内容	改版日
1.0	-	2015年8月4日
1.1	・図 <b>3-1</b> のパスを修正	2015年9月15日
	・図 3-2 にアクセスの方法を追加	
	・表1音階を追加	
2.0	・名称の図を RaspberryPiMouseV2 に変更	2016年6月15日
	・RasPiMouseの組み立て手順を追加	
	・Raspbianの OS のインストール手順を追加	
	・OS のインストールのところで初期ユーザとパスワードを記載	
	・デバイスドライバのインストールのところでデバイスドライバ	
	が RasberryPi2 と RaspberryPi3 共通で使えることを追加	
	・モータの操作で追加された機能 "echo 左 Hz 右 Hz 時間 ms >	
	/dev/rtmotor0"のサンプルを記載	
	・デバイスドライバの使用例を追加	

