Gathering on Raspberry Pi

—Hack on Real World with GPIO—

2014 年 7 月 5 日

1 GPIO とは

GPIO は General Purpose Input/Output（汎用入出力）の略語である。

マイクロプロセッサ、マイクロコントローラ、インタフェースデバイスは、外界と接続するための一つまたは複数の GPIO インタフェースを持っている。

入力として動作した場合は電気回路のほかの部分からのデジタル信号を読み取り、出力として動作した場合は他デバイスの制御や信号の通知を行う。

しばしば GPIO はピンのグループ（典型的には 8 ピン）である GPIO ポートで扱われることがある。通常は個別の GPIO ピンごとに入力または出力に個別に設定することが出来るが、GPIO ポートはグループごとの入出力設定となる。

個別の GPIO の読み書きや入出力の設定は、一つまたは複数の制御レジスタを読み書きすることで行う。

場合によっては、GPIO は割り込みを生成したり、大量のデータのデバイスへの出力／デバイスからの入力を効率的に行うために、DMA を使用することが出来る。

GPIO デバイスの種類は非常に広範囲である。時には、デバイスは非常に単純であり、入出力を切り替えることの出来るピンのグループである。

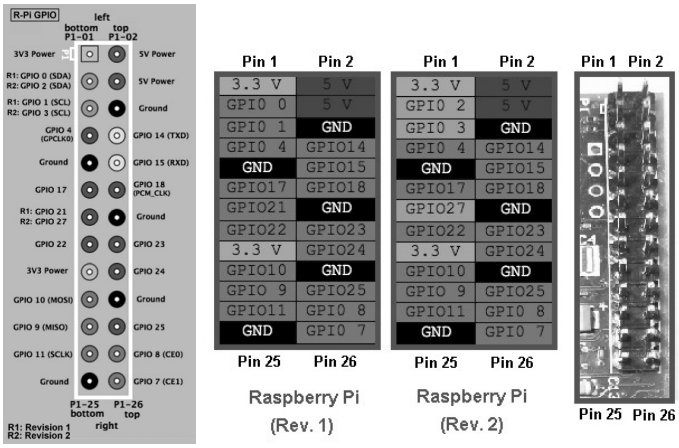
一方では、それぞれのピンが異なるロジック電圧の入出力が出来るよう柔軟に設定可能であり、ドライブ能力やプルアップ／プルダウンも設定可能である。

普遍的ではないが、典型的には入出力電圧は、GPIO を持つデバイスの供給電圧に制限される。制限を越えた電圧により、デバイスが損傷を受けることがある。

いくつかの GPIO には 5V 耐性の入力ピンがある。低い供給電圧 (2V) であっても、デバイスは損傷を受けることなく、5V の入力を受け入れることが出来る。

1.1 Raspberry Pi の GPIO

Raspberry Pi には、外部入出力用のピンが２６本あり、下記のように構成されている。

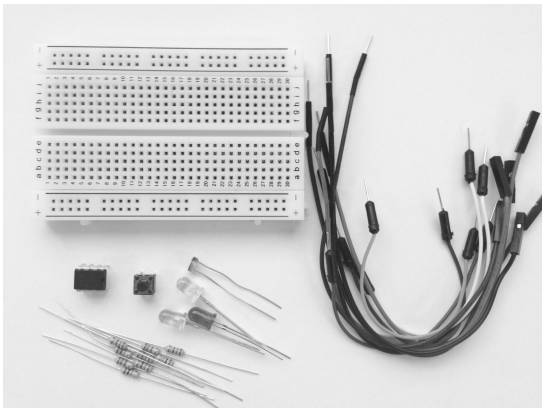


2 GPIO を使う準備

Raspberry Pi に接続する部品は、電子工作でよく使われるものばかりだが、両端がオスとメスになっているジャンパワイヤだけは少し特殊かもしれない。

- ブレッドボード
- LED を何個か
- オス - オスのジャンパワイヤ数本
- オス - メスのジャンパワイヤ数本
- タクトスイッチ
- 抵抗器

オス - メスのジャンパワイヤは Pi のヘッダとブレッドボードをつなぐときに必要。Amazon.co.jp では下記のようなブレッドボードセット（ラズベリーパイ電子工作入門キット）が販売されている。

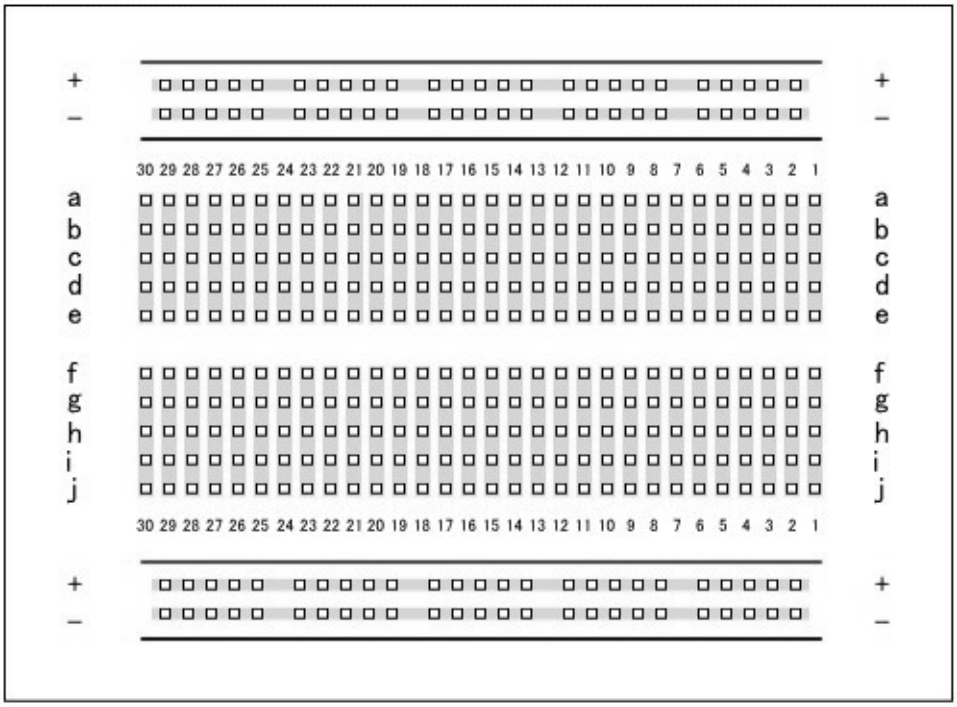


3　LED の点灯

GPIO を試す一番手軽な方法は LED （発光ダイオード）をつないでみることである。その LED をコマンドラインからオンオフしてみる。

3.1　ブレッドボード

ブレッドボードが初めてという人は、まず下図を見て、内部の端子の状態を理解すること。部品同士を電気的に接続するため、金属の端子が埋め込まれている。どこからどこまでがどの向きにつながっているかを、グレーのラインで示した。ブレッドボードによっては１列ずつだったり、まったくなかったりする。この図では、＋と－の記号がついた列が２本ずつある。これを電源ラインと呼ぶ。電源ラインは組み立てやすさのために用意されているもので、電気的には他の端子と変わらない。電源を電源ライン以外の列につなぐことも可能である。



それでは、回路の組み立て型をステップバイステップで説明する。用意した部品をひとつひとつつなぎ、それからコマンドラインで GPIO を操作する。

1. オス - メスのジャンパワイヤを使って、Raspberry Pi の GPIO 25 ピンとブレッドボードの任意の点をつなぐ。
2. もう１本同じワイヤーを用意し、Raspberry Pi のグランド（ground）端子と、ブレッドボードの－（マイナス）記号がついている電源ラインを接続する。なお、この回路のグランドとマイナスは同じ意味。

3. LED を挿す準備ができた。ただ、挿す前に 2 本の足（正しくはリードという）を確認すること。一方が少し長くなっているはず。長い方をアノードといい、この回路で GPIO 25 端子につながるのはアノードである。短い方の足はカソードといい、Pi のグランドにつなぐ。ブレッドボードに挿してしまうと、アノードとカソードの区別がつかなくなってしまいそうだが、LED を帽子に見立てたときのツバの部分を見ると判別可能である。欠けて平らになっている方がカソード。ただし、ツバのない LED もあるので、その場合はリードの長短を覚えておくこと。LED の挿す位置は GPIO 25 からの線をどこに挿したかで決まります。アノードと GPIO 25 がつながるよう、同じ列に挿すこと。カソードを指す先はグランドにつながっている電源ラインである。
4. コマンドラインから GPIO にアクセスするためにはスーパーユーザー権限、root のアカウントが必要である。 su コマンドを実行して、root になっておくこと。

```
pi@raspberrypi ~ $ sudo su
root@raspberrypi:/home/pi#
```

プロンプトが変化したら、root としてコマンドラインにいることが確認できる。
5. LED をオンオフする前に、GPIO 25 をユーザー空間にエクスポートする作業が必要。これをしないと、GPIO にアクセスすることができない。

```
root@raspberrypi:/home/pi# echo 25 > /sys/class/gpio/export
```

使いたいピンの番号（25）を、`echo` コマンドで、`/sys/class/gpio` ディレクトリにある `export` ファイルへ書き込むと、`/sys/class/gpio/gpio25` という新たなディレクトリが自動的に作成され、そのなかに GPIO をコントロールするのに必要なファイルが用意されます。
6. `cd` コマンドでそのディレクトリへ移動し、 `ls` コマンドで中身を確認する。

```
root@raspberrypi:/home/pi# cd /sys/class/gpio/gpio25
root@raspberrypi:/sys/class/gpio/gpio25# ls
active_low direction edge power subsystem uevent value
```

`cd` は “change directory” の略で、操作対象のディレクトリを変更するコマンドである。上記のようにしておくと、毎回 `/sys...` というフルパスでファイルを指定する手間が省ける。`ls` はファイルのリストを表示するコマンドで、実行すると 7 つのファイルが確認できるが、今回必要なのはこのうちの 2 つ、`direction` と `value` だけである。
7. `direction` ファイルはそのピンを入力と出力のどちらに使うのかを指定するためのもの。LED をつなぐので、次のようにして出力であることを伝える。

```
root@raspberrypi:/sys/class/gpio/gpio25# echo out > direction
```

8. いよいよライトオン。`value` ファイルに対して `echo` コマンドを使い、数字の 1 を書き込む。

```
root@raspberrypi:/sys/class/gpio/gpio25# echo 1 > value
```

9. エンターキーを押すと同時に LED が光るはずである。オフにするときは、 0 を書き込む。

```
root@raspberrypi:/sys/class/gpio/gpio25# echo 0 > value
```

ファイルに対する書き込みがピンに対する出力となるならば、逆にピンからの入力はどのように処理すればよいか？そう、正解は「ファイルを読む」。次はそれを試す。

4 スイッチを「読む」

Raspberry Pi のデジタル入力ピンは、3.3V ピンかグランド（0V）ピンのどちらかにつながっていないといけない。それ以外のピンに接続したり、どこにもつながっていなかったりすると動作が予測できなくなる。これから作る回路ではタクトスイッチを使って 3.3V とグランドのどちらかにつながるようにする。一旦仕組みが分かってしまえば、磁石式の防犯スイッチ、ジョイスティック、傾きセンサーなども扱えるようになる。

1. ブレッドボードの中央の溝をまたぐようにタクトスイッチを挿入する。LED のときに使ったブレッドボードの空いているエリアを利用するとグランドの配線を共通化できる。
2. Pi の GPIO 24 ピンとタクトスイッチの上側の端子をオス - メスのジャンパワイヤでつなぐ。
3. Pi の 3.3V ピンとブレッドボードの＋側の電源ラインをつなぐ。
   ※間違って、5V ピンとつながないよう注意。5V をデジタル入力ピンに与えると、電圧が高すぎて Pi が壊れてしまう。
4. タクトスイッチの下側の端子と＋電源ラインをオス-オスのジャンパワイヤで接続する。これで、スイッチを押すと GPIO 24 と 3.3V がつながるようになる。
5. 先ほど、デジタル入力ピンは 3.3V かグランドのどちらかにつながっていないといけない、と述べたが、ボタンが押されていない状態では GPIO 24 ピンの先は浮いた状態になる。浮いたデジタル入力には予測できない値が現れるので、10K Ωの抵抗器を使って GPIO 24 ピンとグランドを接続する。つまり、GPIO 24 ピンには、タクトスイッチと 10k Ω抵抗の 2 つがつながることになる。こうすることで Pi は、ボタンが押されていないときはグランドに、ボタンが押されたときは 3.3V につながっていると判断するようになる。
6. コマンドラインから操作する。root になっておくこと。
7. LED のときと同様にまずはエクスポート。プロンプトは `gpio25` のままになっているので、今回は `24` を `echo` する。

   ```
   root@raspberrypi:/sys/class/gpio/gpio25# echo 24 > /sys/class/gpio/export
   ```

8. 新しいディレクトリへ移動

   ```
   root@raspberrypi:/sys/class/gpio/gpio25# cd /sys/class/gpio/gpio24
   ```

9. 入出力の向きを入力に設定。

   ```
   root@raspberrypi:/sys/class/gpio/gpio24# echo in > direction
   ```

10. ファイルの中身を出力する `cat` コマンドを使って、ピンの状態を読み取る。`cat` は “concatenate”（連結する）の略で、ファイルをつなぐ働きをするのだが、単純に内容を表示する目的でもよく使われる。

    ```
    root@raspberrypi:/sys/class/gpio/gpio24# cat value
    0
    ```

11. ゼロとい表示されるはず。タクトスイッチに触れていない状態では、抵抗器を経由してグランドとつながっているので、0 が返ってくる。それでは、ボタンを押した状態でもう一度実行する。

```
root@raspberrypi:/sys/class/gpio/gpio24# cat value
1
```

12. 1 が表示されれば、すべて正常に動作している。

5 Python を使って GPIO プログラミング

Python を使って GPIO を自由に操作することができる。今回は、Python 用のプログラミングファイルを作成する。

5.1 gpio_led_blink.py

```
#!/usr/bin/python

import RPi.GPIO as GPIO
import time

# GPIO Pin Number
IO_NO = 25

print("press ^C to exit program ...\n")
# Use GPIO Pin Numer
GPIO.setmode(GPIO.BCM)
# Use Number on Board
#GPIO.setmode(GPIO.BOARD)

GPIO.setup(IO_NO, GPIO.OUT)

try:
 while True:
  GPIO.output(IO_NO, True)
  time.sleep(0.5)
  GPIO.output(IO_NO, False)
  time.sleep(0.5)
except KeyboardInterrupt:
 print("\ndetect key interrupt\n")
```

```
GPIO.cleanup()
print("Program exit\n")
```

5.2 gpio_input.py

```
#!/usr/bin/python

import RPi.GPIO as GPIO
import time

IO_NO = 24

print("press ^C to exit program ...\n")
GPIO.setmode(GPIO.BCM)
GPIO.setup(IO_NO, GPIO.IN)

try:
 while True:
  print(GPIO.input(IO_NO))
  time.sleep(1)
except KeyboardInterrupt:
 print("\ndetect key interrupt\n")

GPIO.cleanup()
print("Program exit\n")
```

5.3 gpio_led_switch.py

```
#!/usr/bin/python

import RPi.GPIO as GPIO
import time

INPUT = 24
OUTPUT = 25
message = 0

print("press ^C to exit program ...\n")
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(INPUT, GPIO.IN)
GPIO.setup(OUTPUT, GPIO.OUT)
GPIO.output(OUTPUT, False)

try:
 while True:
  input_value = GPIO.input(INPUT)
  if input_value == True:
      print "The button has been pressd. Lighting LED."
      GPIO.output(OUTPUT, True)
      while input_value == True:
        input_value = GPIO.input(INPUT)
      print "The button has been released. Extinguishing LED."
  if input_value == False:
      GPIO.output(OUTPUT, False)
except KeyboardInterrupt:
 print("\ndetect key interrupt\n")

GPIO.cleanup()
print("Program exit\n")
```

5.4 gpio_shutdown.py

```
#!/usr/bin/python

import RPi.GPIO as GPIO
import os

INPUT = 24
GPIO.setmode(GPIO.BCM)
GPIO.setup(INPUT, GPIO.IN, pull_up_down=GPIO.PUD_UP)

try:
        GPIO.wait_for_edge(INPUT, GPIO.FALLING)
except KeyboardInterrupt:
        GPIO.cleanup()  # clean up GPIO on CTRL+C exit
GPIO.cleanup()          # clean up GPIO on normal exit
os.system("/sbin/shutdown -h now")
```