Raspberry Piにおける I/O操作メモ

山本健一

2020 年 9 月 14 日

1　シェルを用いたI/O動作

1.1　Lチカ

まず、図 1(a) のように接続します.　ポート 5 の電位を high(3.5V) にすれば LED は点灯し, low(0V,GND) にすれば LED は消えます. これを Raspberry Pi の LXTerminal にて (sh を用いて) 実施するには, 次の手順で入力します [1].

1. ポート 5 の使用開始

   ```
   echo "5" > /sys/class/gpio/export
   ```

2. ポート 5 を出力用にする

   ```
   echo "out" > /sys/class/gpio/gpio5/direction
   ```

3. ポート 5 の電位を”high” にする

   ```
   echo "1" > /sys/class/gpio/gpio5/value
   ```

4. ポート 5 の電位を”low” にする

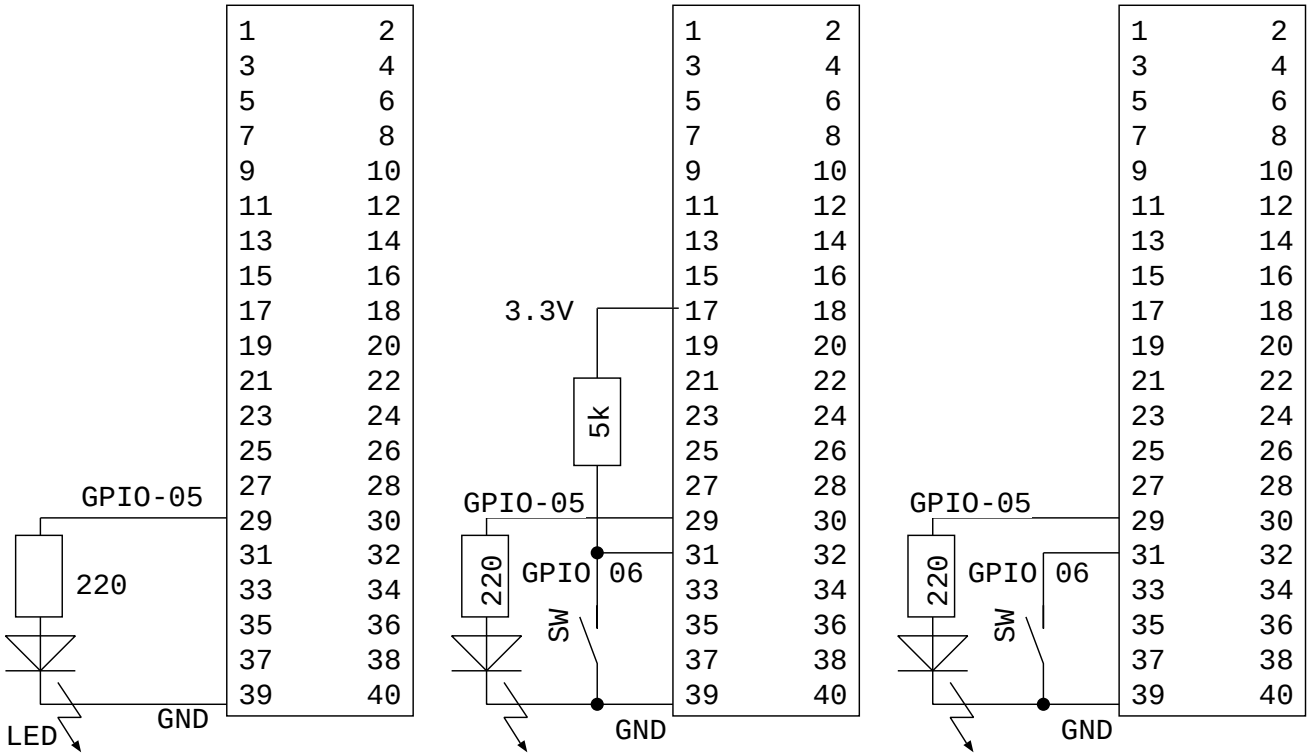
   ```
   echo "0" > /sys/class/gpio/gpio5/value
   ```

5. ポート 5 の使用終了

   ```
   echo "5" > /sys/class/gpio/unexport
   ```

これを sh プログラミングを用いて自動的に (あるいは対話的に) 動作させることも可能ですで, 手軽にプログラム (シェルプログラミングについて勉強する必要はありますが...) できて便利な一方で, 実行速度は遅いです. ちなみに, ここで用いたコマンド “echo” は引数の部分を表示する命令で, “>” はリダイレクトといって標準出力に表示されるべき内容をファイル等に書き込む動作をします [2]. ここではファイルの代わりに当該 I/O ポートの状態に対応するフラグに書き込み, 変更操作をこなっています.

実行速度を改善するには C プログラミングで上記と同じことを実施しますが, これについては後に書きます.



(a) (b) (c)

図 1: Raspberry Pi I/O 端子 と LED, スイッチの接続

1.2 スイッチ

スイッチの ON/OFF を判定します. 図 1(b) のように接続されていれば, スイッチが ON のときにはポート 6(の電位) は low(0V), スイッチが OFF で high(3.5V) となります. それを確認する手順は次のようになります.

1. ポート 6 の使用開始

   ```
   echo "6" > /sys/class/gpio/export
   ```

2. ポート 6 を入力用にする

   ```
   echo "in" > /sys/class/gpio/gpio5/direction
   ```

3. ポート 5 の電位を確認する

   ```
   cat /sys/class/gpio/gpio5/value
   ```

   スイッチが ON なら 0, OFF なら 1 が表示されるはずです.

4. ポート 5 の使用終了

   ```
   echo "5" > /sys/class/gpio/unexport
   ```

前と似た作業手順となっていることを確認してください. ちなみに “cat” は引数 (ファイルなど) の内容を出力する命令です.

次のシェルプログラム (プログラム左の行番号は入力する必要はありません.) はスイッチ ON で LED が点灯, スイッチ OFF で LED 消灯します. 実行にあたっては, 「sh ファイル名」とキー入力しリターンキーを押せば実行されます. このプログラムは無限ループになっていますので, 何もしなければずーっと実行され続けます. 終了したければ, 「Ctrl-C」(コントロールキーを押しながら C キーを押す) を入力すればプログラムは中止されます.

```
1  #!/bin/sh
2
3  echo "5" > /sys/class/gpio/export
4  echo "out" > /sys/class/gpio/gpio5/direction
5
6  echo "6" > /sys/class/gpio/export
7  echo "in" > /sys/class/gpio/gpio6/direction
8
9  while :
10 do
11     if test `cat /sys/class/gpio/gpio6/value` -eq 1
12     then
13         echo "0" > /sys/class/gpio/gpio5/value
14     else
15         echo "1" > /sys/class/gpio/gpio5/value
16     fi
17 done
18
19 echo "5" > /sys/class/gpio/unexport
20 echo "6" > /sys/class/gpio/unexport
```

このプログラムでは実行後に”unexport” をしていないので, 複数回実行すると冒頭でエラーが出るなど, 改善の余地はありますが, 簡単な動作確認が可能ですのでお試しください.

(追記) スイッチおよびポート 6 が 5.1kΩ 抵抗でプルアップされている回路を先に紹介しましたが, Raspberry-Pi 内部にもこのプルアップ抵抗が用意されています. これを利用する場合には, 図 1(c) のように接続し, 次のように入力すると, 内部プルアップが有効になります.

```
echo "high" > /sys/class/gpio/gpio6/direction
```

2 C 言語

2.1 GPIO デバイスドライバ

C 言語を用いたサンプルを作成してみました. 動作はシェルプログラムとほぼ同じですが, ここではエラー処理が入っています.

```
1  /*************************************************
2  Raspberry Pi I/O sample program
```

```
3  coded by K.Yamamoto, 2020/09/11
4
5  ------------------------------------------------
6  pin
7  (29)#5 --> LED -- R -- GND
8  (31)#6 --> SW -- GND
9          |--> R -- 3.3V(17)
10 ************************************************/
11
12 #include <stdio.h>
13 #include <stdlib.h>
14 #include <string.h>
15 #include <fcntl.h> // for open
16 #include <unistd.h> //for close
17
18 /************************************************/
19 void gpio_export(int *fd);
20 void gpio_direction(int *fd, int i);
21 void gpio_value(int *fd, int i);
22 void init_gpio();
23 void gpio_close();
24
25 /************************************************/
26 int main(){
27   int fd_led; // port# for LED
28   int fd_sw;  // port# for SW
29   char sw[2];
30
31   init_gpio();
32   gpio_value(&fd_led, 5);
33   while(1){
34         gpio_value(&fd_sw, 6);
35         read(fd_sw, sw, 2);
36         close(fd_sw);
37         switch(sw[0]){
38         case '0':
39           write(fd_led, "0", 2); // LED OFF
40           break;
41         default:
42           write(fd_led, "1", 2); // LED ON
43         }
44   }
45   gpio_close();
46   return(0);
```

```
47 }
48
49 /*********************************************/
50 void gpio_export(int *fd){
51   *fd = open ("/sys/class/gpio/export", O_WRONLY);
52   if(*fd < 0){
53         printf("GPIO export open error\n");
54         exit(1);
55   }
56 }
57
58 /*********************************************/
59 void gpio_direction(int *fd, int i){
60   char s[64];
61   sprintf(s, "/sys/class/gpio/gpio%d/direction",i);
62   *fd = open (s, O_RDWR);
63   if(*fd < 0){
64         printf("GPIO %d direction open error\n", i);
65         exit(1);
66   }
67 }
68
69
70 /*********************************************/
71 void gpio_value(int *fd, int i){
72   char s[64];
73   sprintf(s, "/sys/class/gpio/gpio%d/value",i);
74   *fd = open (s, O_RDWR);
75   if(*fd < 0){
76         printf("GPIO %d value open error\n", i);
77         exit(1);
78   }
79 }
80
81 /*********************************************/
82 void init_gpio(){
83   int fd;
84   int i;
85   gpio_export(&fd);
86   write(fd, "5", 2); // LED
87   write(fd, "6", 2); // SW
88   close(fd);
89
90   i=5;//LED
```

```
91    gpio_direction(&fd, i);
92    write(fd, "out", 4);
93    close(fd);
94
95    i=6;//SW
96    gpio_direction(&fd, i);
97    write(fd, "in", 3);
98    //  write(fd, "high",5); // pull-up
99    close(fd);
100 }
101
102 /*************************************************/
103 void gpio_close(){
104   int fd;
105   fd = open("/sys/class/gpio/unexport", O_WRONLY);
106   if(fd < 0){
107         printf("GPIO unexport open error\n");
108         exit(1);
109   }
110   write(fd, "5", 2);
111   write(fd, "6", 2);
112   close(fd);
113 }
```

なお, このプログラムにはスイッチのチャタリング対策などが施されていませんので, 実用的にはもう一工夫が必要となります.

2.2 Wiring-Pi

I/O 制御用のライブラリ Wiring-Pi を利用してみます.

```
1  /*************************************************
2  WiringPi IO control sample program
3  coded by K.Yamamamoto, 2020/09/13
4
5  compile: cc  gpio_wp.c -lwiringPi
6  exec: sudo ./a.out
7  ------------------------------------------------
8  pin
9  (29)#5 --> LED -- R -- GND
10 (31)#6 --> SW -- GND
11        |--> R -- 3.3V(17)
12 *************************************************/
13 #include <wiringPi.h>
14
```

```
15 int main(void){
16   wiringPiSetupGpio();
17   pinMode(5, OUTPUT);
18   pinMode(6, INPUT);
19   //  pullUpDnControl(6, PUD_UP);
20
21   while(1){
22         if(digitalRead(6) == 0){
23           digitalWrite(5, LOW);
24         }else{
25           digitalWrite(5, HIGH);
26         }
27   return 0;
28   }
29 }
```

上記のようにプログラム (gpio_wp.c) を簡潔に書くことができます. このコンパイルは次のようにします.

```
cc gpio_wp.c -lwiringPi
```

ここでプログラム中の「#include <wiringPi>」およびコンパイル時のオプションスイッチ「-l」(L の小文字) によって wiringPi ライブラリを用いることをコンパイラーに教える必要があります.

なお, このプログラムにはスイッチのチャタリング対策が含まれていないことは前節と同様です. また, このプログラムは I/O を直接操作するために, 一般ユーザ権限 (pi) ではなく root 権限を必要とする場合があります. そのときは「sudo ./a.out」などによって実行してください.

WiringPi は, このような I/O の個別操作に加えて, I/O ポートを用いるデータ通信方式であります I^2C や SPI, 割り込み処理などのコマンドが用意されていて近年よく用いられています. WiringPi に関する文法など詳細は文献 [3] の付録部にまとめてあるので, こちらを参照してください.

参考文献

[1] 「お手軽 ARM コンピュータラズベリー・パイで I/O」 インターフェース SPECIAL, CQ 出版 (2013)

[2] ブルース・ブリン「入門 UNIX シェルプログラミング」改訂第 2 版, 山下訳, ソフトバンク (2003)

[3] 「ラズパイで入門! Linux I/O プログラミング教科書」第 2 版, CQ 出版社 (2019)