ハードウェアへのアクセス

T. Obina (KEK, Accelerator Division 7)

2018/11/01(木) 15:30 - 17:00

概要

EPICS 入門セミナー第 1 日目; 午後の最後。Raspberry Pi を使って外部のハードウェアを制御する。最も簡単な例として、GPIO（General Purpose Input/Output）を使って LED を点灯させる。

目次

1　はじめに

Raspberry Pi を使っている人は多く、Web に多くの記事がある。もちろん Google で検索するも良いけど、迷ったときには公式ドキュメントを読むことを勧めたい。

https://www.raspberrypi.org/documentation/

例えば、ピン配置に関しては公式ドキュメントの

```
Document -> Usage -> GPIO
```

からドキュメントが参照できる。配布している pinout を印刷した紙もここにあるリンクからダウンロードして印刷している。

https://github.com/splitbrain/rpibplusleaf

https://pinout.xyz/

https://elinux.org/RPi_Low-level_peripherals

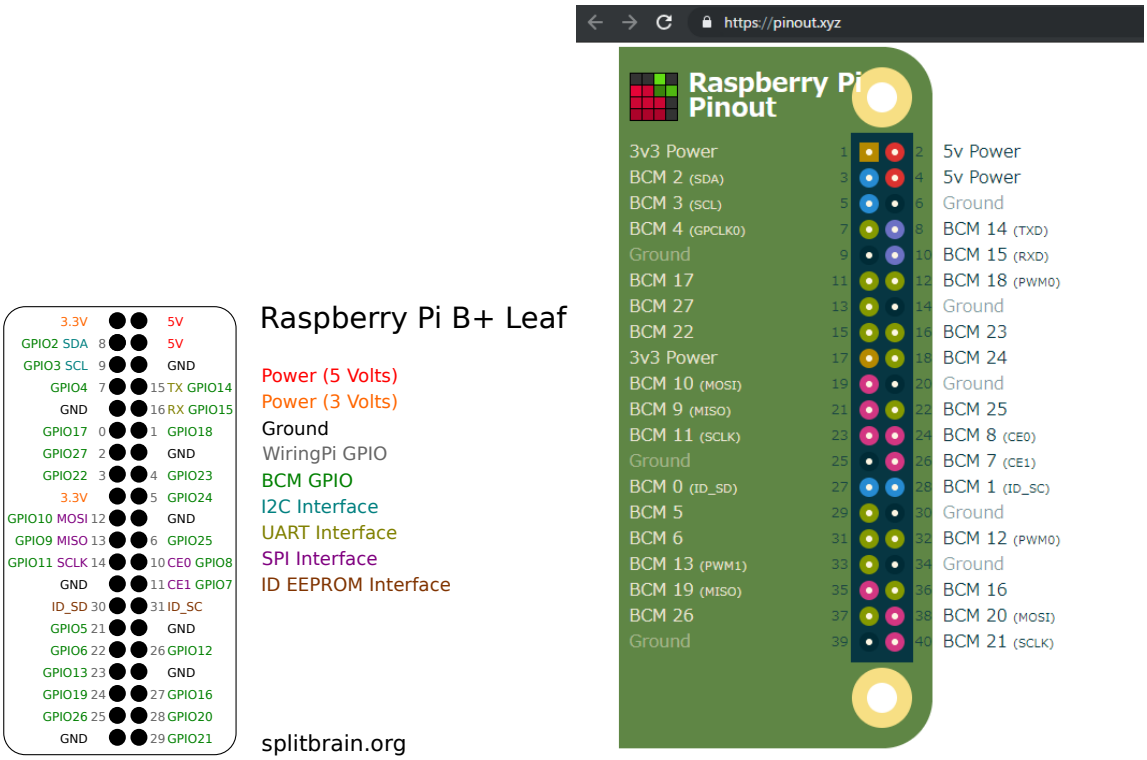

図 1　Raspberry Pi の GPIO ピン。元ファイルはリンク先からダウンロード可能。BCM は Broadcom のピン番号

これらの Web にアクセスできない場合、ログインした状態で「pinout」コマンドによってターミナルで確認できる。これは便利なので覚えておくと良い。

詳細なハードウェア情報に関しては Broadcom 社の BCM2835 のマニュアルを読むしかない https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf RasPi3 B+ は BCM2837 を使っているので、そちらを参照したいが適切なものが無い。(ほぼ 2835 と同じと思われるが)

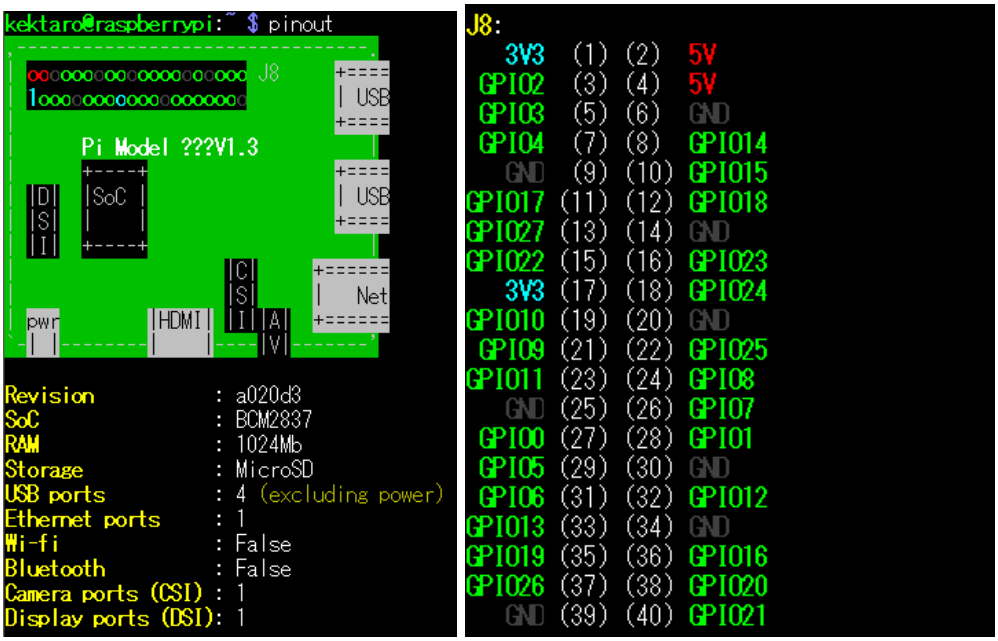

図 2　pinout コマンドの出力

2　回路

2.1　最大定格について

Raspberry Pi の GPIO は 3.3 V 出力。取り出せる電流はモデルによって異なるようだが、典型的には各ピンごとに 16 mA までで、全てのピン合計が 50 mA 以内 [1]。今回使用する LED は Vf = 2.3 V, 最大電流 20 mA. 実際に適当な明るさで点灯するには 2〜3mA 流せば十分のようだ。まずは単純に GPIO ピンに電流制限抵抗をつけて LED をつなぐ。(抵抗が無いと、最悪の場合回路や RasPi を壊してしまう可能性がありますので注意してください）[Q] 抵

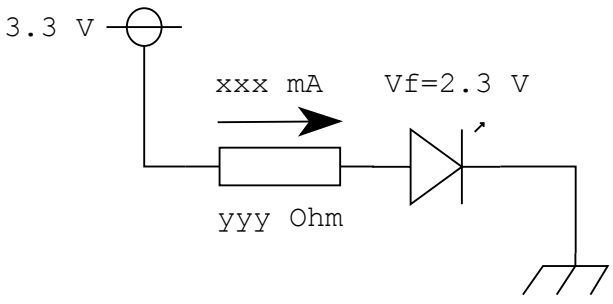

図 3　LED 点灯のための回路図

抗は何Ωにすれば良い？

【参考】5V 系の電源は USB で供給できる最大電流まで流すことが出来る（とはいえ 300 mA くらいまでが現実的)。(明日の実習では 5V 系の電源とトランジスタ (あるいは FET) を on/off スイッチとして使うやり方についても紹介する）

2.2　ブレッドボードの使い方

solderless breadboard（ソルダーレス・ブレッドボード）とは、回路部品を差し込むだけで、はんだ付け不要で電子回路を組むことが出来る基板のこと。実習で使うタイプでは両脇に縦方向につながったライン（+, - 表記）があり、これは主に電源線として使うことが多い。5 個ずつ穴が開いている部分は横方向につながっている。(写真でいうと abcde の 5 個と、fghij の 5 個はそれぞれつながっている) 真ん中に溝がある部分で分かれている。これは IC を個

の部分に設置するのに便利なことが分かるだろう。今回の実習で使用している各種のパーツ類やブレッドボードなどは秋月電子 [2] から購入しているものが多くあります。「ソルダレス・ブレッドボードの達人」なる文書もダウンロード可能です [3]。その他、多くの文書が web にあります [4] ので興味のある人は検索してください。

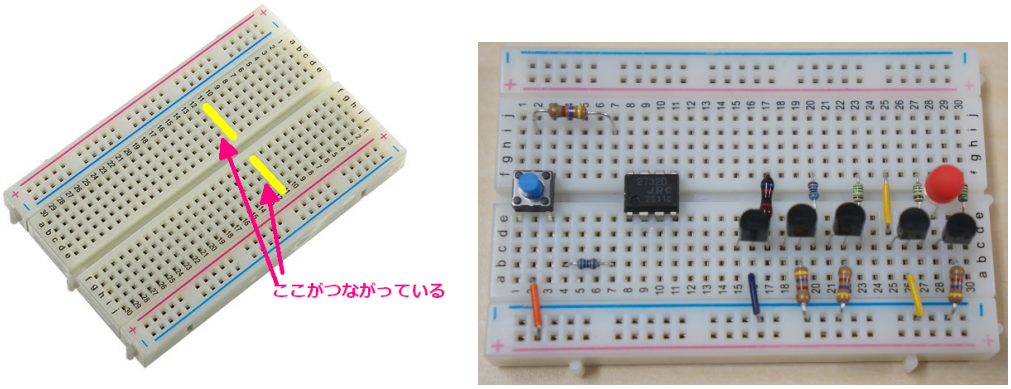

図 4　ブレッドボードの例。 黄色で示す横方向がつながっている。右図は部品を乗せた例。

【実習】以下の例では GPIO17 のピンを使って ON/OFF 制御します。この回路図を実現するように配線してください。

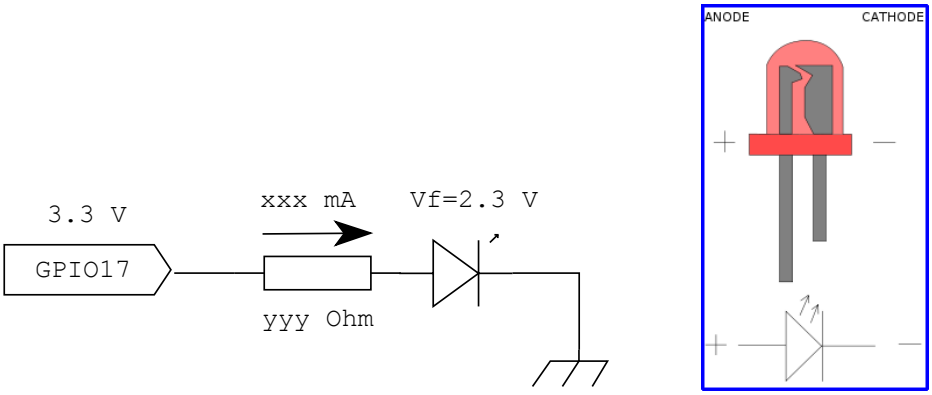

図 5　GPIO17 を LED に接続

3　EPICS を使わずに、LED 点灯試験

3.1　gpio コマンドを使った制御

動作確認のため、EPICS を使わずに簡単に検証する例を示します。最初は WiringPi と呼ばれる C 言語ライブラリをつかって作成されたプログラムを使って制御する例です。(明日午後のセミナーで gpio コマンドについて少し説明します) 今日はあくまでも動作確認が目的ですので、ここでは呪文と思って使ってください。GPIO ピン番号を指定するため -g オプションをつけています。

```
$ gpio -g mode  17 out    <-- GPIO #17 を出力モードに設定
$ gpio -g write 17 1      <-- 電圧 HIGH 出力
$ gpio -g write 17 0      <-- 電圧 LOW 出力
```

HIGH/LOW に応じて LED が点灯/消灯するでしょうか？もしこの段階で点灯しないならば、回路が間違っていますので再確認を。

3.2 sysfs をつかった制御 (紹介のみ)

次に sysfs を使った例です。sysfs は Linux カーネル 2.6 で導入された仮想ファイルシステムです。(興味があれば Google 検索を)

0．作業前の状態を確認

```
$ ls /sys/class/gpio/
export  gpiochip0  gpiochip128  unexport
```

1．gpio ピンを使う宣言をおこなう

```
$ echo 17  > /sys/class/gpio/export
$ ls /sys/class/gpio/
export  gpio17  gpiochip0  gpiochip128  unexport
$ ls -l /sys/class/gpio/
gpio17 -> ../../devices/platform/soc/3f200000.gpio/gpiochip0/gpio/gpio17
$ ls /sys/class/gpio/gpio17/
active_low  device  direction  edge  power  subsystem  uevent  value
```

2．方向の確認

```
$ cat /sys/class/gpio/gpio17/direction
in
```

となっているので、out に書き直す

```
$ echo out > /sys/class/gpio/gpio17/direction
$ cat /sys/class/gpio/gpio17/direction
out
```

3．1 or 0 を書き込んで ON/OFF する

```
$ cat /sys/class/gpio/gpio17/value
0
$ echo 1 > /sys/class/gpio/gpio17/value
$ cat /sys/class/gpio/gpio17/value
1
```

4．最後に、消灯してから

```
$ echo 0 > /sys/class/gpio/gpio17/value
```

5．使用終了宣言をおこなう

```
$ echo 17 > /sys/class/gpio/unexport

$ ls /sys/class/gpio/
export  gpiochip0  gpiochip128  unexport
```

3.3 python をつかった制御 (紹介のみ)

Python から使う例は公式ドキュメントを参照する。

https://www.raspberrypi.org/documentation/usage/gpio/python/README.md

```
$ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from gpiozero import LED
>>> led = LED(17)
>>> led.on()
>>> led.off()
```

4 EPICS による GPIO 制御

4.1 IOC 作成

テンプレートとして”ioc”を使用してひな形をつくる。ここではホームディレクトリの下に epics/app というディレクトリを作り、その下に各 application のディレクトリを作る方針とする。例として、名前を gpio1 とする。最終的なディレクトリ構造は以下のようになる

```
.
|--epics
|   |-- app
|   |   |
    |   |-- gpio1
        |   |-- configure
        |   |   |-- Makefile
        |   |   |-- RELEASE
        |   |
        |   |-- gpio1App
        |   |   |-- Db
        |   |   |   |-- Makefile
        |   |   |   `-- test.db
        |   |   |-- Makefile
        |   |   `-- src
        |   |       |-- gpio1Main.cpp
        |   |       `-- Makefile
        |   |-- iocBoot
        |   |   |-- iocgpio1
        |       |   |-- Makefile
        |           `-- st.cmd
        |
        |-- example
        |-- gpio2
        |-- i2c
        |-- stream
        |
```

4.1.1 ioc ひな形作成

コマンドラインでは以下のように入力していく（一部省略）。テンプレートとして ioc を指定する。

```
$ mkdir -p epics/app/gpio1
$ cd epics/app/gpio1/
$ makeBaseApp.pl -t ioc gpio1
$ makeBaseApp.pl -i -t ioc gpio1
Using target architecture linux-arm (only one available)
The following applications are available:
    gpio1
What application should the IOC(s) boot?
The default uses the IOC's name, even if not listed above.
Application name?   <---- Enter を入れるのみ
$
```

4.1.2 configure/RELEASE 編集

次に <TOP>/configure/RELEASE ファイルを編集して、GPIO ライブラリを指定する。先人の努力により、Raspberry Pi の GPIO を使うデバイスサポートを作成してくれている [5]。ソースは https://github.com/ffeldbauer/epics-devgpio からダウンロード可能。

```
#SNCSEQ=/opt/epics/R315.6/modules/soft/seq/2.2.4
#ASYN=/opt/epics/R315.6/modules/soft/asyn/4-31
#STREAM=/opt/epics/R315.6/modules/soft/stream/2-7-7
#STREAM=/opt/epics/R315.6/modules/soft/stream/2-7-7_I2C
RPIGPIO=/opt/epics/R315.6/modules/soft/gpio/20160308   <--- コメントを外す
#RPII2C=/opt/epics/R315.6/modules/soft/i2c/20170603
```

4.1.3 Db 編集

<TOP>/gpio1App/Db ディレクトリに移動し、データベースを作成する。DTYP は "devgpio" とする。以下のようなファイルを作成して保存する。GPIO17 番を使う場合は OUT フィールドに @17 と記述する。H/L は電圧がかかっている状態とゼロの状態で、どちらが HIGH レベルかを指定している。(Makefile に追加することを忘れないように。例えばデータベースファイル名を test.db としたならば Makefile の中に DB += test.db と記述。)

```
record(bo, "$(head):GPIO17:OUT") {
  field(DTYP, "devgpio")
  field(OUT,  "@17 H")    # or GPIO17, Active High
  field(ZNAM, "OFF")
  field(ONAM, "ON")
}
```

4.1.4 src 編集

<TOP>/gpio1App/src ディレクトリに移動し、Makefile を編集する。通常ではここに各種のソースコードを追加するのだが、今回は既存のコンパイル済みデバイスサポートを使用する (configure/RELEASE で指定している）ので、ここではライブラリやデータベース定義ファイル (.dbd ファイル）を追加するのみで良い。

```
...
gpio1_DBD += devgpio.dbd
gpio1_LIBS += devgpio
...
```

4.1.5 make

<TOP> ディレクトリに移動し、make 実行. エラーが出たら修正すること。

```
$ cd ~/epics/app/gpio1
$ make
make -C ./configure install
make[1]: ディレクトリ '/home/kektaro/epics/app/gpio1/configure' に 入ります
perl -CSD /opt/epics/R315.6/base/bin/linux-arm/makeMakefile.pl O.linux-arm ../..
.....
make[1]: ディレクトリ '/home/kektaro/epics/app/gpio1/iocBoot' から 出ます
```

4.1.6 スタートアップスクリプト編集

<TOP>/iocBoot/iocgpio1 ディレクトリに移動し、st.cmd を編集する。

```
#!../../bin/linux-arm/gpio1
....

dbLoadDatabase "dbd/gpio1.dbd"
dbLoadRecords("db/test.db", "head="ET_kektaro")    <--- 追加
GpioConstConfigure("RASPI B+")                     <--- 追加

cd "${TOP}/iocBoot/${IOC}"
iocInit
```

4.1.7 ioc 実行

最後に、./st.cmd を実行する。ファイルに実行権限を付けてから

```
$ chmod +x ./st.cmd
```

ioc を起動する。

```
$ ./st.cmd
#!../../bin/linux-arm/gpio1
< envPaths

dbLoadDatabase "dbd/gpio1.dbd"
gpio1_registerRecordDeviceDriver pdbbase
## Load record instances
dbLoadRecords("db/test.db", "head=ET_kektaro")

GpioConstConfigure("RASPI B+")
GpioConst: Loading Look-Up table for Raspberry Pi B+

iocInit
Starting iocInit
############################################################################
## EPICS R3.15.6
## EPICS Base built Oct 18 2018
############################################################################
iocRun: All initialization complete

epics>
```

4.2 コマンドラインから制御

まず dbl コマンドでデータベースリスト確認

```
epics> dbl
ET_kektaro:GPIO17:OUT
epics>
```

dbpf コマンドでデータベースリスト確認

```
epics> dbpf ET_kektaro:GPIO17:OUT 1
DBR_STRING:          "ON"
epics> dbpf ET_kektaro:GPIO17:OUT 0
DBR_STRING:          "OFF"
```

別の端末を開き、caget/caput で値を確認/設定

```
$ caget ET_kektaro:GPIO17:OUT
ET_kektaro:GPIO17:OUT          OFF

$ caput ET_kektaro:GPIO17:OUT 1
Old : ET_kektaro:GPIO17:OUT          OFF
New : ET_kektaro:GPIO17:OUT          ON

$ caget ET_kektaro:GPIO17:OUT
ET_kektaro:GPIO17:OUT          ON

$ caput ET_kektaro:GPIO17:OUT OFF  <--- 0/1 or "ON"/"OFF"で設定可能
Old : ET_kektaro:GPIO17:OUT          ON
New : ET_kektaro:GPIO17:OUT          OFF
```

4.3 GUI 作成

CSS を使って GUI 作成。ON/OFF するとともに、ステータス表示もおこなう。

5 時間がある方へ

- 他の人の LED をモニター (or 制御!)
- caput で bo レコードに 0/1 以外の数値を入れてみる（ex. 0.5, 9 など)
- bo レコードの HIGH フィールドを設定してパルス出力（モーメンタリー出力）に変更してみる
- ioc を起動する際、２回目以降は警告表示が出る (動作に問題はない)。これの原因は?
- devgpio のソースコード `/opt/epics/R315.6/modules/soft/gpio/20160308` を眺めてみる
- 複数の LED を同時に設定したい：ビット列のデータを同時に設定する (mbbo または longout)。 デバイスサポートは bi/bo のみなので、DB Link で実現するのがとりあえずは簡単。

参考文献

[1] elinux.org Low-evel Peripherals `https://elinux.org/RPi_Low-level_peripherals`

[2] 秋月電子 http://akizukidenshi.com

[3] ブレッドボードの達人 `http://akizukidenshi.com/download/akibread.pdf`

[4]「武蔵野電波のプロトタイパーズ」`https://pc.watch.impress.co.jp/docs/column/musashino_proto/517061.html`

[5] RasPi EPICS devGpio : Florian Feldbauer, `https://github.com/ffeldbauer/epics-devgpio`