ハードウェア I/O 2
day2 13:00 ～

KEK, High Energy Accelerator Research Organization
帯名　崇
(takashi.obina@kek.jp)

2018/11/01 ～ 02 EPICS入門セミナー@KEK つくばキャンパス（３号館１階会議室）

このセッションでのおはなし

最初に全員でできる実習(GPIO の割り込みでEPICS レコードを動かす) を行う。その後、各自でトピックスを選んで実習を行う。人数が限定されるものもあるので、その場合は適当に交代しながら実施する。

お手元の手引書に実習の詳細は記載してあります

1） 共通：GPIOの割り込みでレコードプロセス

2） Arduinoでアナログ入力（10名交代で）

3） 時間があれば（SNL/Sequencer でLED 点滅）

4） 昨日の補足講義：

- CSS DataBrowser
- CSS OPI Sample

将来のトピックス

- I2C 経由でデータ取得
- LEDのON/OFF : DIO直接ではなくトランジスタ経由で

実習１）GPIOの割り込みでレコードプロセス

これまでのサンプルで　SCAN フィールドは何があったでしょうか？

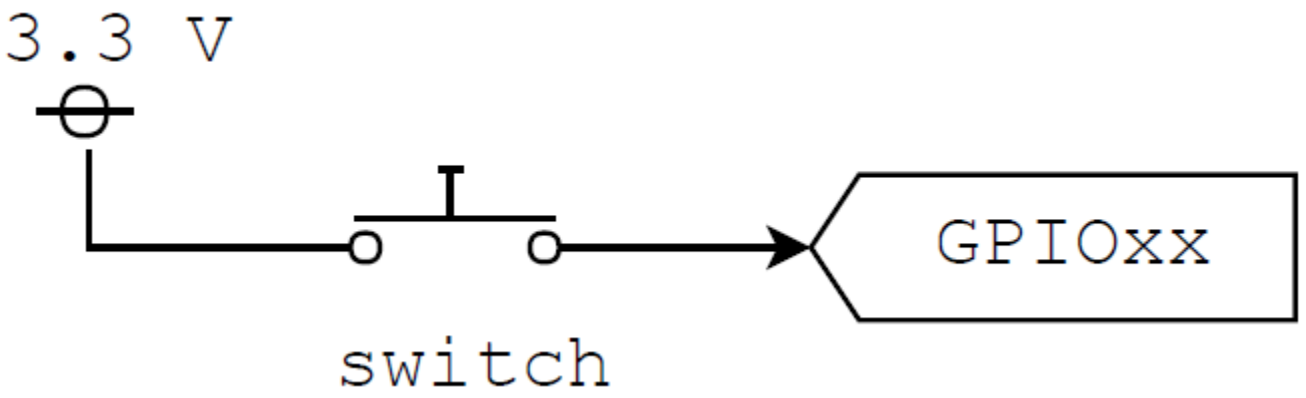
何が使えるかは<appTop>/dbd/<appName>.dbd ファイルにかいてあります。

```
menu(menuScan) {
        choice(menuScanPassive, "Passive")
        choice(menuScanEvent, "Event")
        choice(menuScanI_O_Intr, "I/O Intr")
        choice(menuScan10_second, "10 second")
        choice(menuScan5_second, "5 second")
        choice(menuScan2_second, "2 second")
        choice(menuScan1_second, "1 second")
        choice(menuScan_5_second, ".5 second")
        choice(menuScan_2_second, ".2 second")
        choice(menuScan_1_second, ".1 second")
}
```

実習１）GPIOの割り込みでレコードプロセス

目的：外部からの電圧入力を検出してEPICS レコードを動かしたい

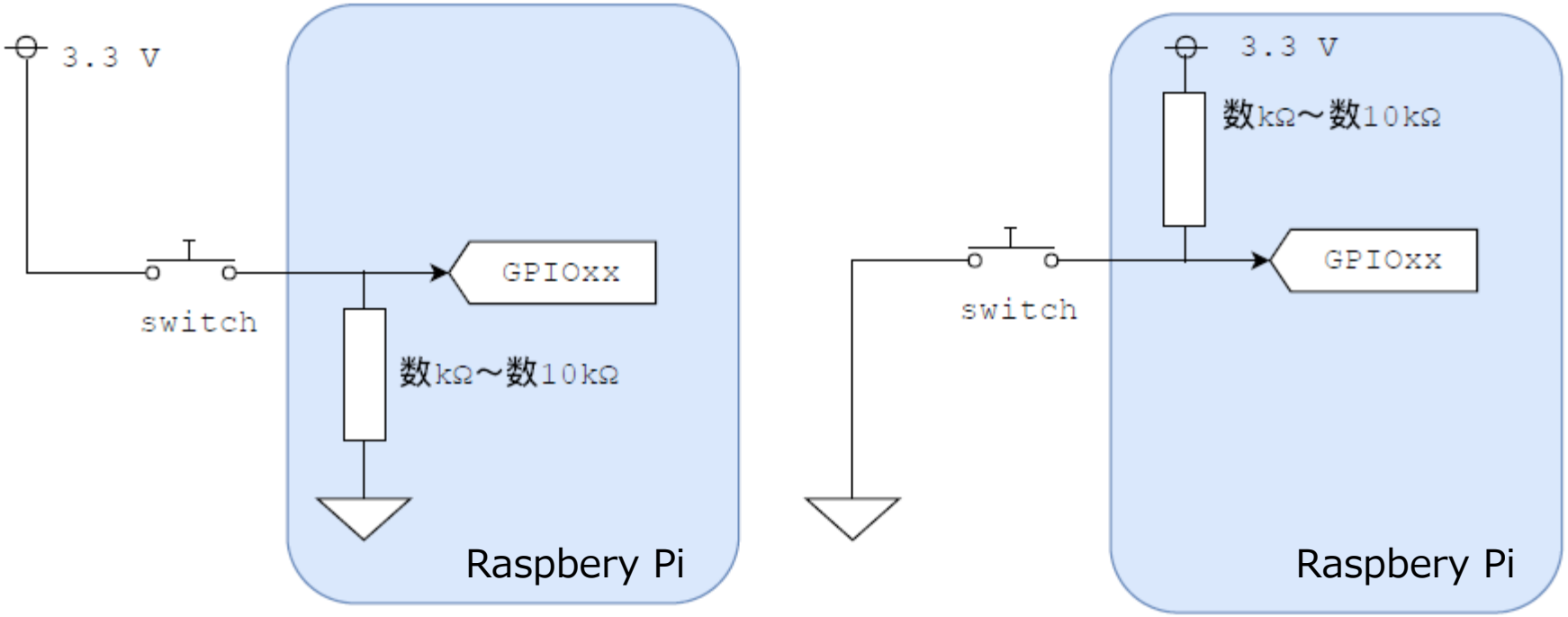
～ソフトで定義した 1 second とかではなく～



例えば、こんな回路を考える

入力がOpenのときは High or Low?

Openだと、入力はどちらになるのかわかりません。
Raspberry Pi の内部ではプルアップまたはプルダウンされています
（これはピンによって異なる）

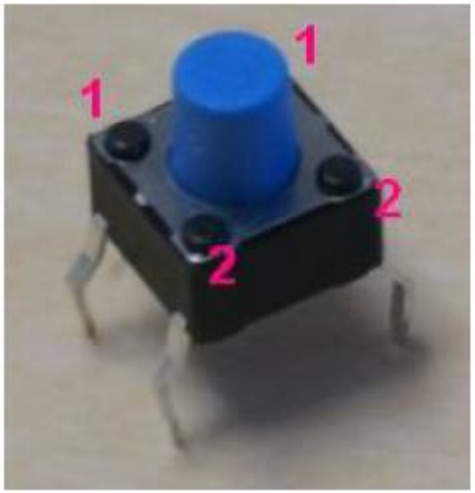


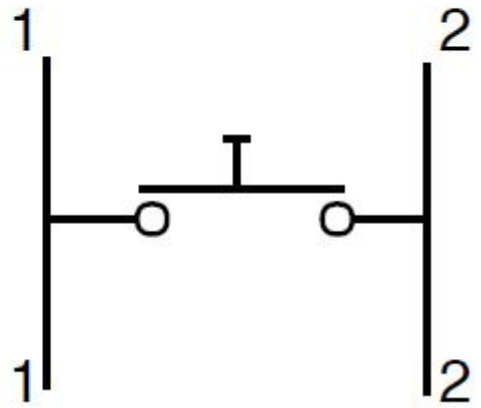
gpio readall

```
+-----+-----+---------+------+---+---Pi 3+---+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
|     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
|   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5v      |     |     |
|   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
|   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  |
|     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
|  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
|  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
|  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
|     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
|  10 |  12 |    MOSI | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
|   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
|  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
|     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
|   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
|   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
|   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
|  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
|  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
|  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
|     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+---Pi 3+---+---+------+---------+-----+-----+
```

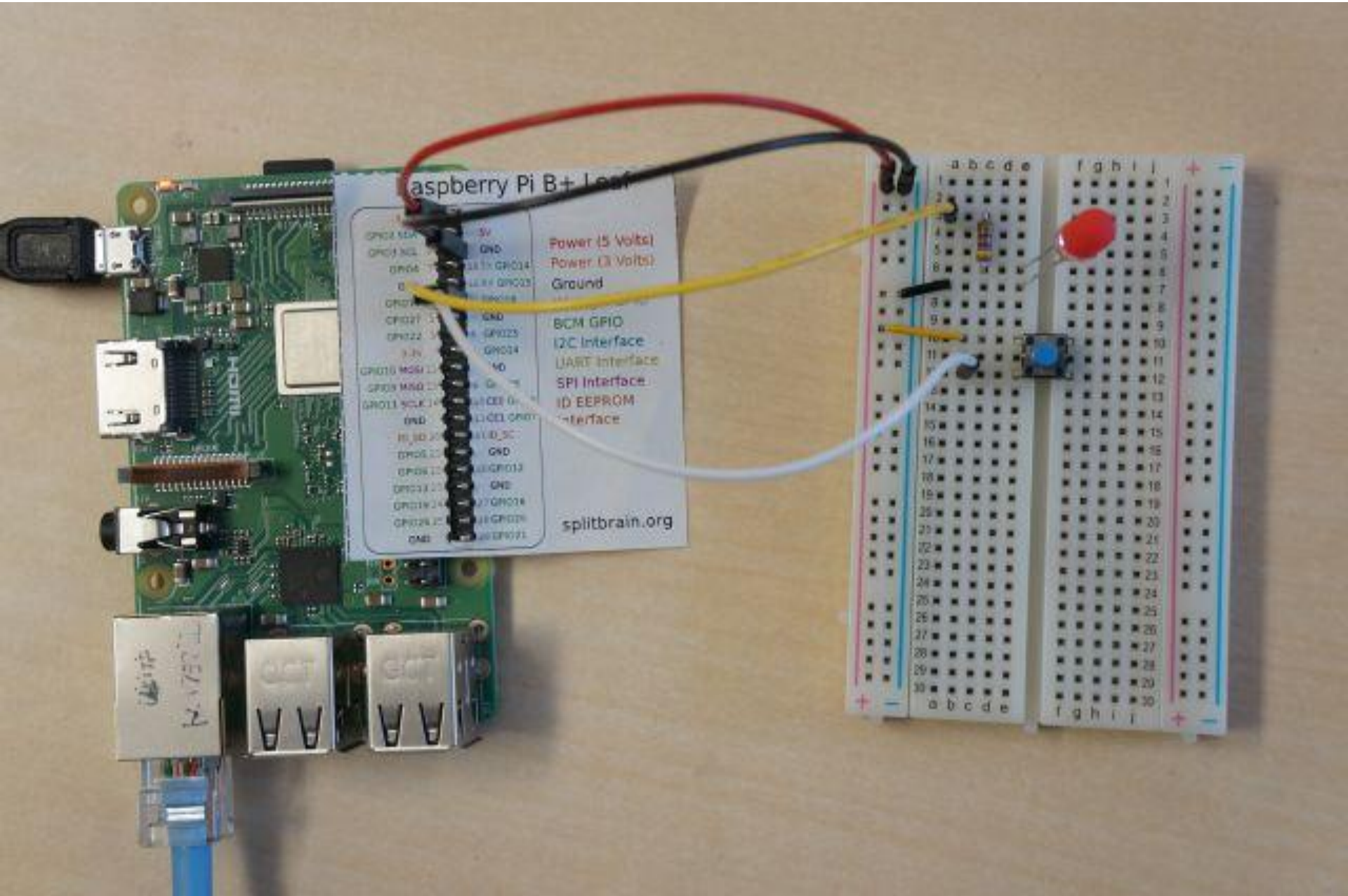
プッシュスイッチ

押している間だけ、１と２の間が導通します





配線例


Power (5 Volts)
Power (3 Volts)
Ground
BCM GPIO
I2C Interface
UART Interface
SPI Interface
ID EEPROM
splitbrain.org

dbファイルに追記

昨日LEDを点灯したファイルをそのまま使います。
dbファイルは残っていますか？18番ピンを追記すればOK。

```
record(bo, "$(head):GPIO17:OUT") {
        field(DTYP, "devgpio")
        field(OUT,  "@17 H")
        field(SCAN, "Passive") #Default
        field(ZNAM, "OFF")
        field(ONAM, "ON")
}
record(bi, "$(head):GPIO18:IN") {
        field(DTYP, "devgpio")
        field(INP,  "@18 H")
        field(SCAN, "I/O Intr")
        field(ZNAM, "OFF")
        field(ONAM, "ON")
}
```

その後

いつものように、makeして st.cmd 実行

ボタンを押したらレコードが動くことをcamonitorで確認

その後の実習項目

- 今回追加したレコードをGUI で表示し、ボタンを押すのに同期して表示が変わることを確認する。
- 自分のRaspberry Pi だけでなく、ほかの人のレコードも表示してみる

時間の余った人へ

追加実習項目

- 負論理のスイッチを作成する。プルアップしているピンを使うこと。
- 単にロジックを反転したいだけならばPU/PDどちらでも構わず、ソフトで反転するので十分。実際には(どちらかといえば)プルアップ入力の方が使われることが多い。理由を考える。
- devGpio デバイスサポートのソースコードを読んでみる
- 現在は信号の立ち上がり/立下りエッジの両方を検出してレコードが動いています。場合によっては立ち上がりエッジだけを検出したいこともある（例えばリモコンのボタンとか）ので、ソフトウェア（レコードのリンク)で対処する方法を考えてみる。
- デバイスサポートに手を入れてどちらかのエッジだけで割り込みをかける方法を考えてみる（可能 or 無理まで含め）
- gpio コマンド(wiring pi) を使ってGPIOピンを(EPICSを使わずに)制御してみる。
- コマンドラインからLEDを制御する"だけ"ならば、gpioコマンドで十分。わざわざ EPICS IOC にする意義（御利益）は何なのかを考える。

実習２：アナログ入力信号を測定したい

ADCの入力をどうやって RPi まで運んでいくか？

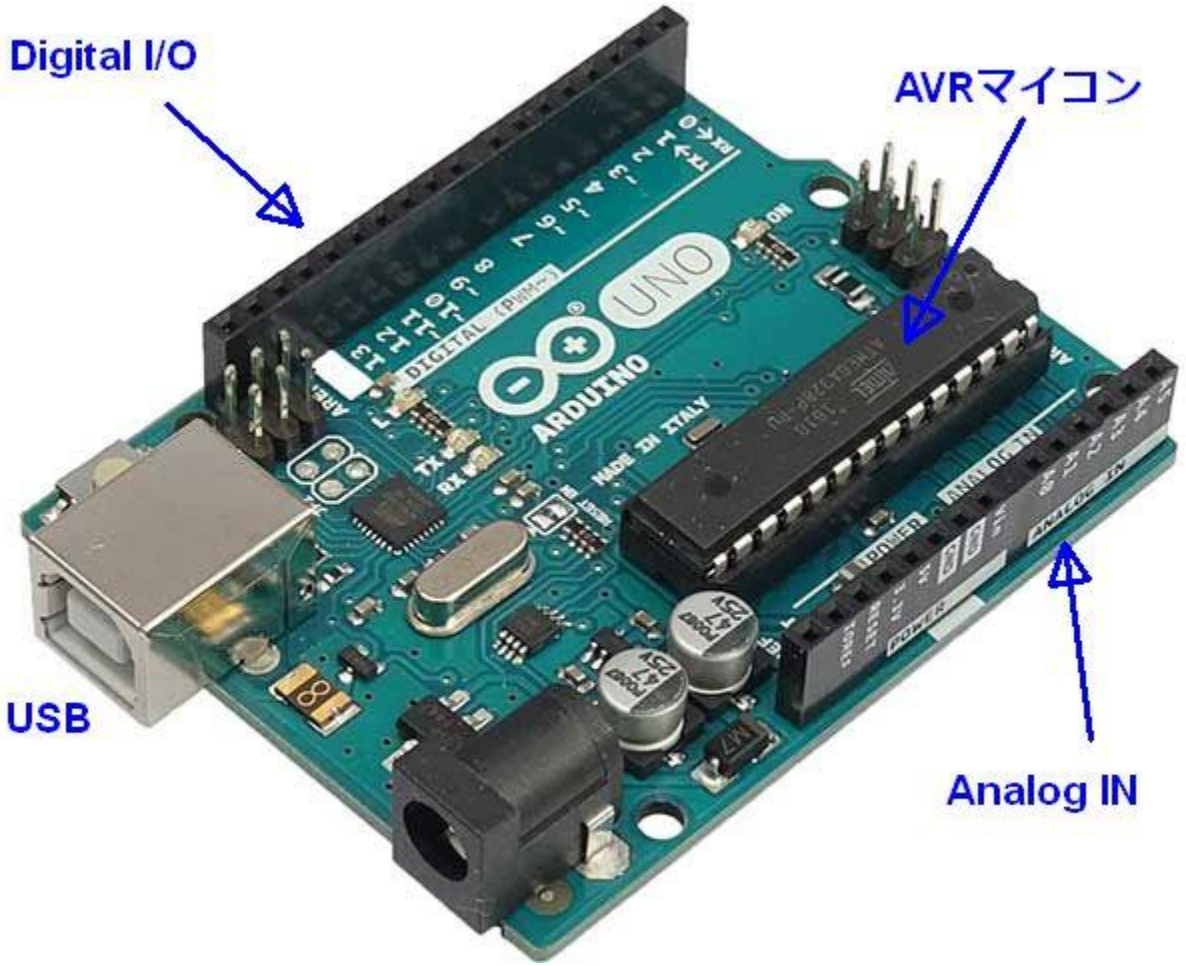
- ADC on Arduino <==[UART(RS232/USB)]==> RPi
- ADC + I2C ==[I2C]==> RPi
- ADC + SPI ==[SPI]==> RPi
- ADC + I2C ==[I2C]== [マイコン] UART(RS232/USB)]==> RPi

など、色々な構成が考えられる

まずはArduinoを使ってアナログデータを取る方法を説明

Arduino

Arduinoを使ってアナログ入力



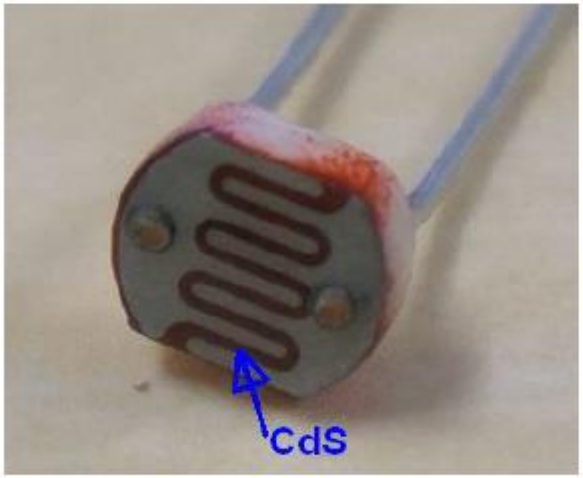
通信仕様、コマンド

item	spec
通信速度	115200 bps
parity	none
デリミタ	*

func	command
version	?
analog input	S
analog output	s
digital input	R

例：analog 1 ch を読み込むには “S1“を送る

アナログ照度センサー



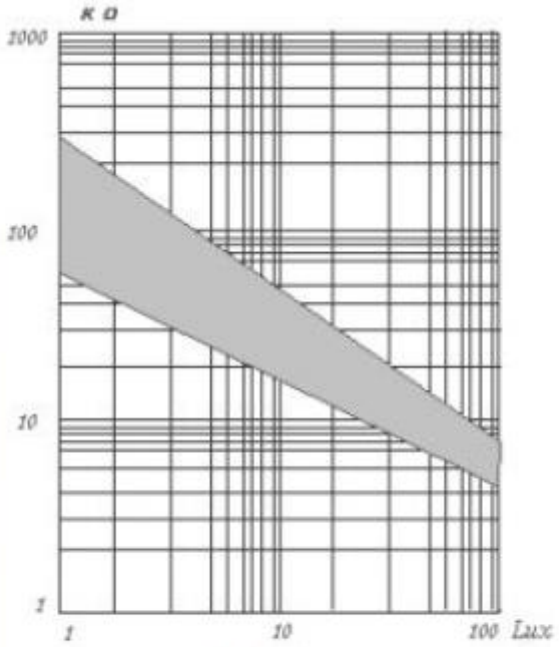


Fig. 3

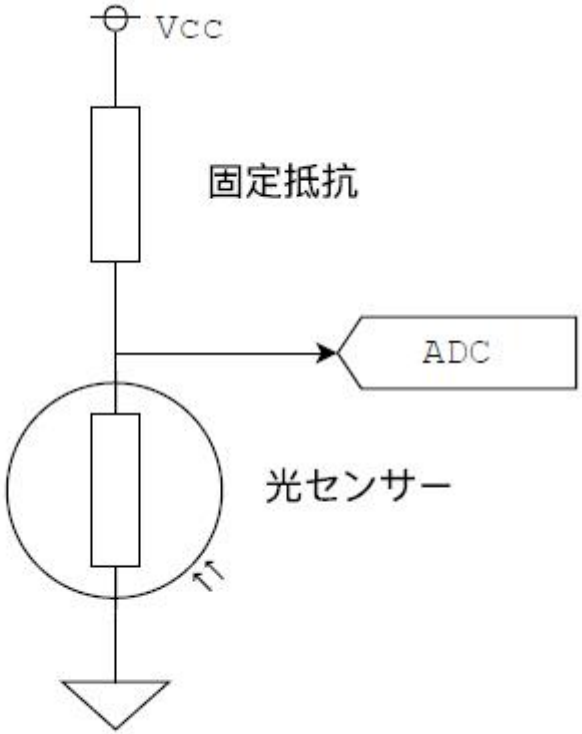


図 7　CdS センサーの写真と特性, 回路例。

Terminalでデバッグ

- GNU Screen 使用
  - ? コマンドを送り、バージョン番号とデリミタが返ってくることを確認

- 最後に “CTRL-a k” （killコマンド）を送るか、USB を引き抜くかしてターミナルを切断する

EPICS Application 作成

主な手順は以下の通り：

- テンプレートとして “nykit” を使う
- configure/RELEASE 編集
- st.cmd に /dev/ttyACM0 あるいは /dev/serial/by-id/usb-….. を記述
- st.cmd に実行フラグを付ける

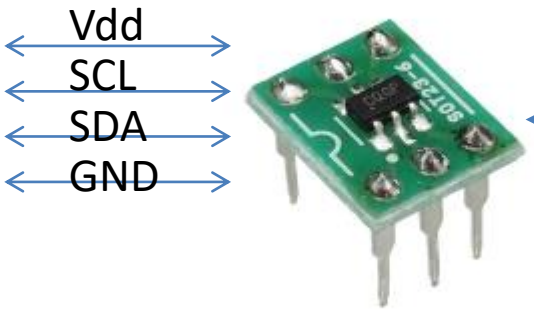
エラー無く起動したら

- dbl　でリスト確認
- $(head):Arduino:LUM を camonitor , 手をかざす
- 該当するdbファイルと protocol ファイルを見る
- 他のコマンド（温度センサーなど）をみる

I2CでRPiに取り込む

いままで説明してきたのは Arduino でアナログ信号取り込み

- ADCが10bit（0 - 1024）しかないのはさみしい
- 例えば、16bit ADC を I2C バス経由で取り込むことが出来る



I2Cインターフェース付き
16bit ADC

I2Cインターフェース付きセンサー

Vdd
SCL
SDA
GND

I2Cインターフェース付き
温度センサー

他にもいろいろなセンサーがある

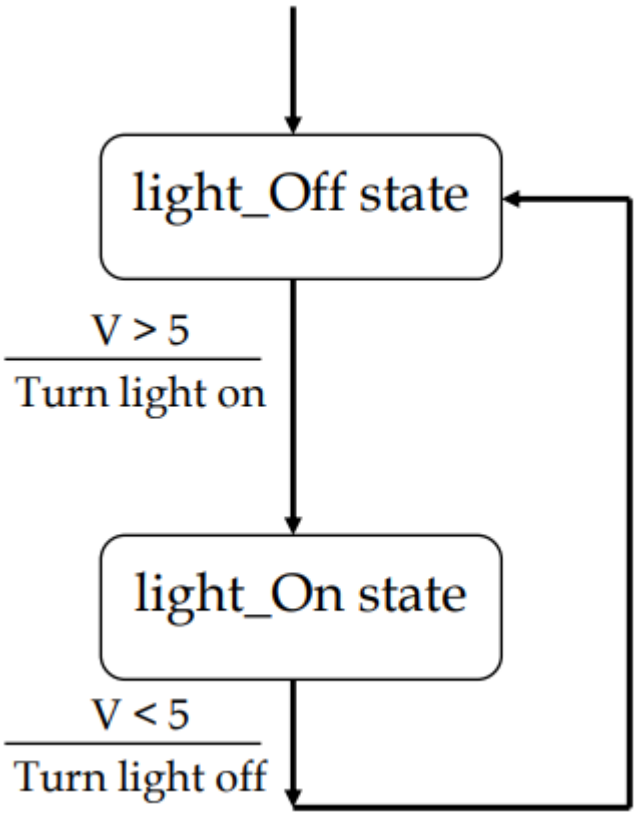
実習３：SNL/Sequencerを使ってLED制御

ここでは EPICS 標準配布物に含まれる State Notation Language (SNL) および Sequencer について簡単に説明し、一定周期でLEDを点滅させるために使用する。

SNLは状態記述言語と呼ばれる言語の１つで、プログラマは「状態(state)」および「状態間の遷移条件」記述することで制御プログラムを見通し良く、また信頼性・保守性の高いプログラムを書くことができる。

SNLで書きやすいプログラム

２つの “State” がある：Light_ON と Light_OFF
レコードの値によって２つのステートを行き来する




```
program sncExample
double v;
assign v to "{user}:aiExample";
monitor v;

ss ss1 {
  state low {
    when (v > 5.0) {
      printf(“Changing to high¥n");
    } state high
  }
  state high {
    when (v <= 5.0) {
      printf("Changing to low¥n");
    } state low
  }
}
```

１秒間隔でON/OFFする

delay関数を使う: 引数の時間[sec]が経過するとTrueを返す関数。

```
assign out to "{head}:GPIO17:OUT";

ss ss1 {
    state LEDoff {
        when (delay(0.5)) {
            out = 1;
            pvPut(out);
        } state LEDon
    }
    state LEDon {
        when (delay(0.5)) {
            out = 0;
            pvPut(out);
        } state LEDoff
    }
}
```

ちょっと変更して、周期と幅を変更できるように

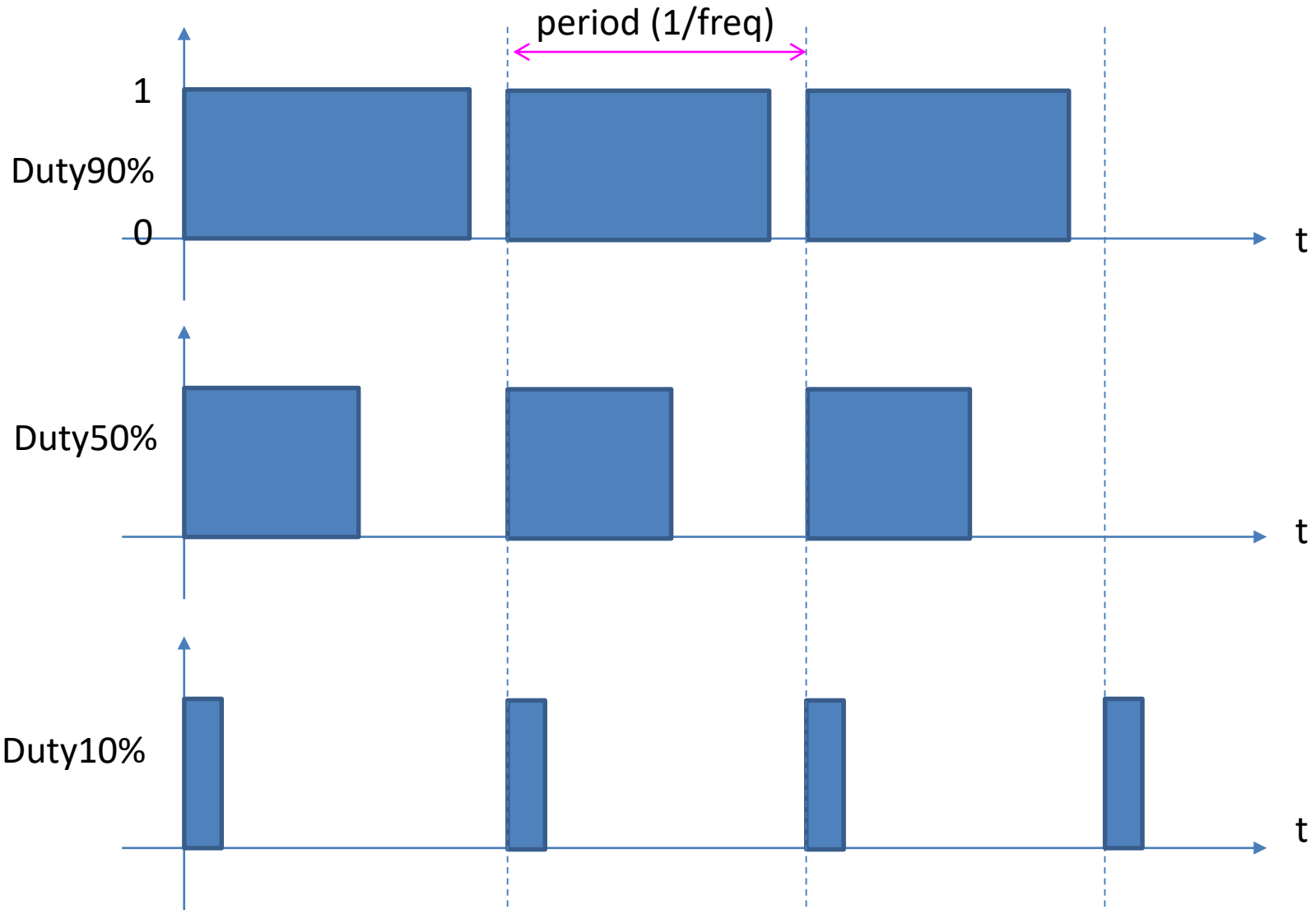
これがテンプレート（pwm）に入っています

```
program sncPwm

assign out to "{head}:GPIO17:OUT";
assign  freq to "{head}:GPIO17:PWM_FREQ";
assign  duty to "{head}:GPIO17:PWM_DUTY";

ss ss1 {
    state LEDoff {
        when (delay(1/freq*(100-duty)/100.0)) {
            out = 1;
            pvPut(out);
        } state LEDon
    }
    state LEDon {
        when (delay(1/freq*(duty)/100.0)) {
            out = 0;
            pvPut(out);
        } state LEDoff
    }
}
```

PWM



共通：CSSに関しての追加演習→山田さんを参照

昨日触っていますが、（おそらく）役立つ Tips をいくつか紹介

Data Browser について

- waveform データをアーカイブからとってきて(alan)表示するには？
- カーソルの表示
- グラフに注釈(annotation)を付けたい (例えば室温のデータ２日分取ってきて、どの時刻に何が起きたか記述）。
- 水平・垂直軸の拡大/縮小
- 水平軸を動かす：マウスで軸をつかんで左右に動かす
- 垂直軸を動かす：複数のデータをプロットしておき、そのうち１つのデータの縦方向位置を動かしてみる。
- プロットしたデータを保存する。保存形式やオプションを変えてみる。

OPI について

- Sampleをいくつか触ってみる
- OPIにDataBrowserを貼り付ける

注：講師陣も触ったことのないモノが多くあります。
実は動かないものもある？