実習： GPIO割り込み（+おまけ）

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

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) を行います。
もし時間に余裕があれば2～4 を行います

1）　共通：GPIOの割り込みでレコードプロセス

2）　pythonでepics制御
3）　SNL/Sequencer でLED 点滅
4）　Arduinoでアナログ入力

実習１）GPIOの割り込みでレコードプロセス

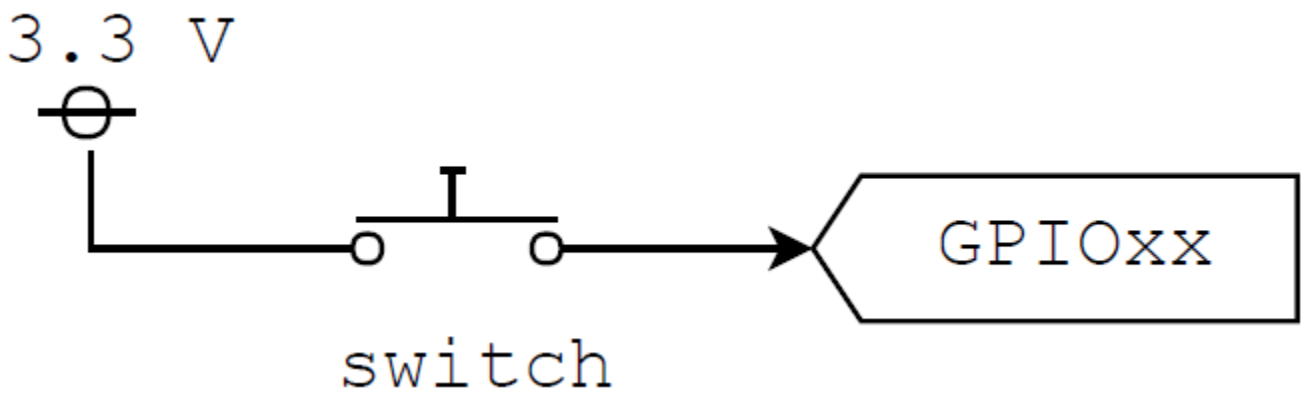
これまでのサンプルで SCAN フィールドは何があったでしょうか？
何が使えるかは<appTop>/dbd/<appName>.dbd ファイルにかいてあります。

```
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

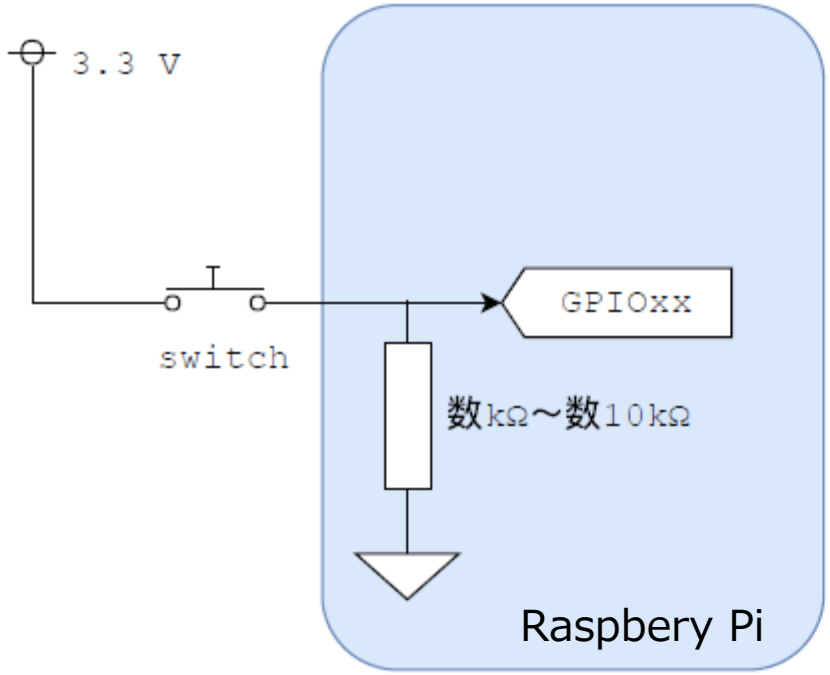
LEDではGPIO ピンを「出力」として使いましたが、今度は「入力」として使います。

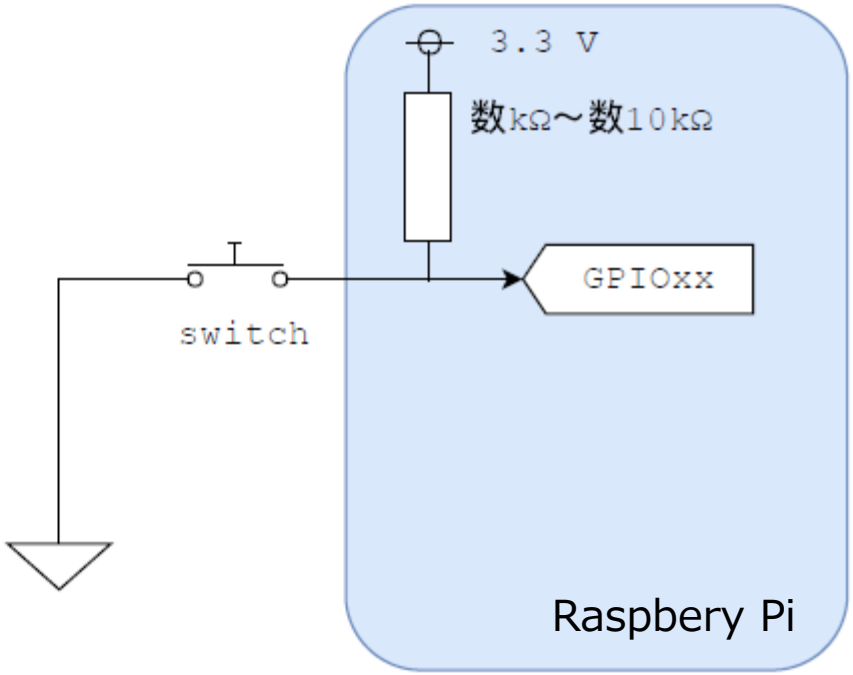


例えば、こんな回路を考えます

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
（どちらになっているかはピンによって異なる）





raspi-gpioコマンド

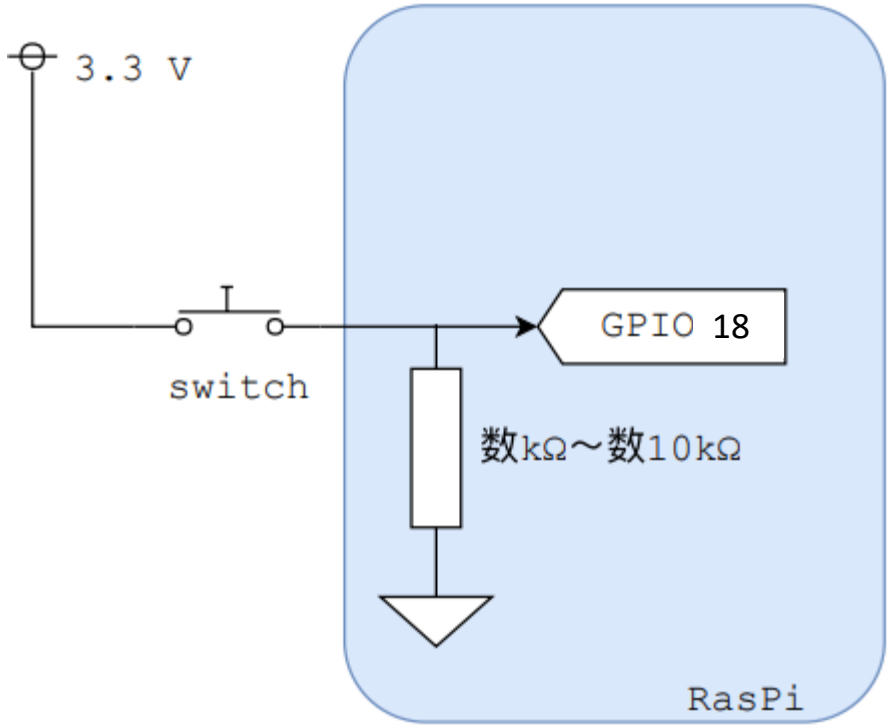
- 以前はWiringPiというパッケージにgpioコマンドがありましたが、今は非推奨となっています
- 替わりに raspi-gpio コマンドを使います。ここで level=1となっているところがpull-up。

```
epics@raspberrypi:~ $ raspi-gpio get
BANK0 (GPIO 0 to 27):
GPIO 0: level=1 fsel=0 func=INPUT
GPIO 1: level=1 fsel=0 func=INPUT
GPIO 2: level=1 fsel=0 func=INPUT
GPIO 3: level=1 fsel=0 func=INPUT
GPIO 4: level=1 fsel=0 func=INPUT
GPIO 5: level=1 fsel=0 func=INPUT
GPIO 6: level=1 fsel=0 func=INPUT
GPIO 7: level=1 fsel=0 func=INPUT
GPIO 8: level=1 fsel=0 func=INPUT
GPIO 9: level=0 fsel=0 func=INPUT
GPIO 10: level=0 fsel=0 func=INPUT
GPIO 11: level=0 fsel=0 func=INPUT
GPIO 12: level=0 fsel=0 func=INPUT
GPIO 13: level=0 fsel=0 func=INPUT
```

```
GPIO 14: level=0 fsel=0 func=INPUT
GPIO 15: level=1 fsel=0 func=INPUT
GPIO 16: level=0 fsel=0 func=INPUT
GPIO 17: level=0 fsel=0 func=INPUT
GPIO 18: level=0 fsel=0 func=INPUT
GPIO 19: level=0 fsel=0 func=INPUT
GPIO 20: level=0 fsel=0 func=INPUT
GPIO 21: level=0 fsel=0 func=INPUT
GPIO 22: level=0 fsel=0 func=INPUT
GPIO 23: level=0 fsel=0 func=INPUT
GPIO 24: level=0 fsel=0 func=INPUT
GPIO 25: level=0 fsel=0 func=INPUT
GPIO 26: level=0 fsel=0 func=INPUT
GPIO 27: level=0 fsel=0 func=INPUT
```

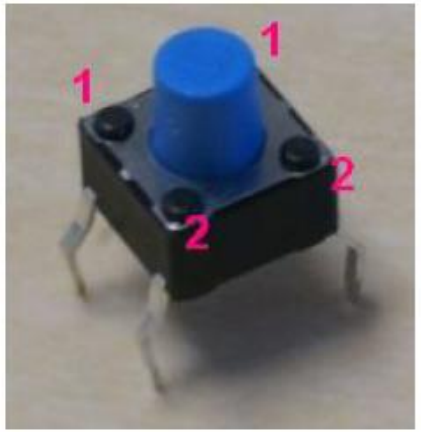
つくりたい回路

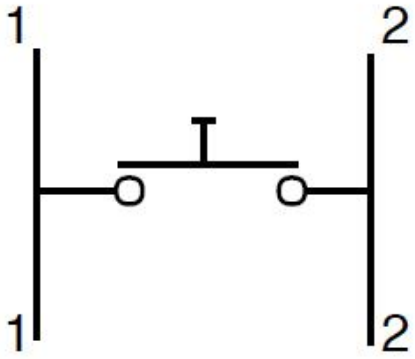
ここでは、プルダウンのピン（GPIO18を例にします）をつかって下の回路を構成しましょう



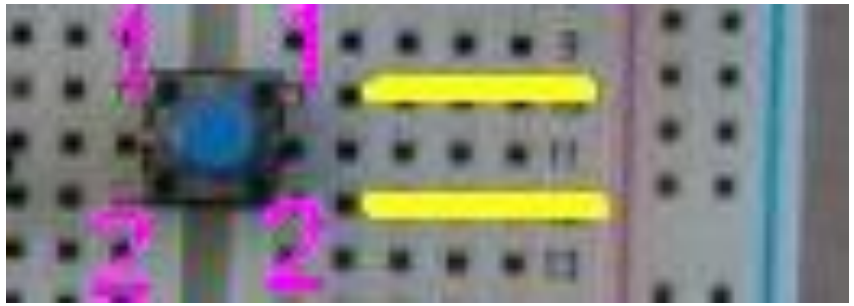
プッシュスイッチ

押している間だけ、1と2の間が導通します

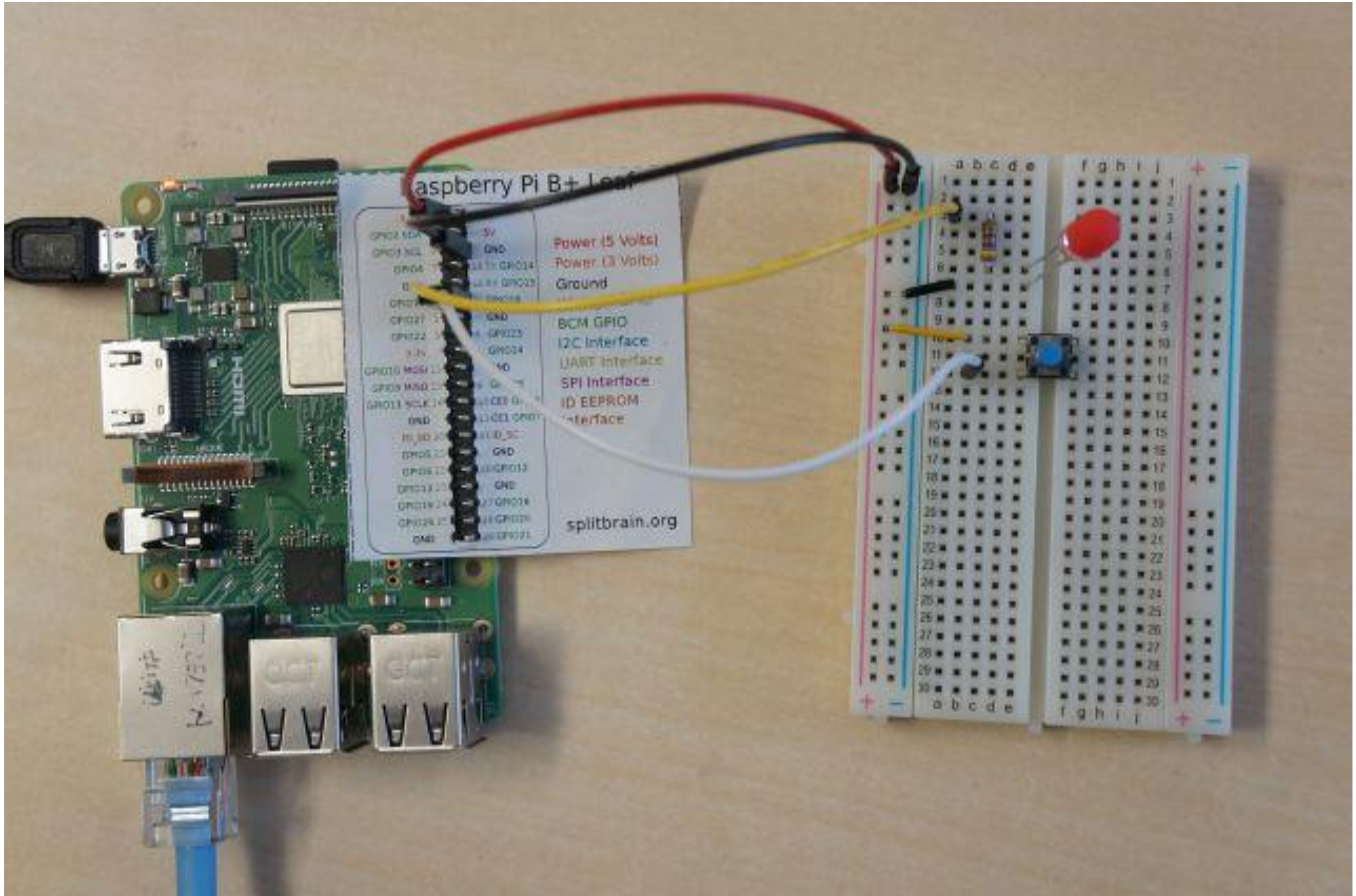




どちらが1なのか2なのか、分かりづらいかもしれません。お勧めはテスターでのチェックですが、
下の写真のように「中央の溝」をまたぐように置いてみると、簡単に挿せる方向があると思います
（90°回すと足のピンが届かない）。このパーツの場合はこれで向きを確認するのが簡単です。



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れば18番ピンを追記するだけ。

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

その後

いつものように、<appTop>ディレクトリで 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 や CS-Studio で確認してください

時間の余った人へ

追加実習項目

- トグルスイッチを押したら、LEDが点灯するようにしてみましょう
  - GPIO18:IN のレコードをプロセスしたら、 GPIO17:IN が連動して動くように FLNK でつなぐ
  - GPIO17:OUT の DOL (Desired Output Location) をGPIO18の値に設定する
  - GPIO17:OUT の OMSL (Output Mode Select) を "closed_loop" に設定する
  - これらの項目については EPICS Base: Record Reference Documentation を参照
  - (次ページに sample dbfile)
- 現在は押している間だけ点灯し、離すと消灯する動作になっているので、これを「押すたびにON/OFFが切り替わる動作」に変更してみる

- 負論理のスイッチを作成してみる: 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。
- RasPiのコマンドラインからLEDを制御する"だけ"ならば、raspi-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。

dbファイル例

2つのレコードを連動させる

```
record(bo, "$(user):GPIO17:OUT") {
    field(DTYP, "devgpio")
    field(SCAN, "Passive")
    field(OUT,  "@17")
    field(ZNAM, "OFF")
    field(ONAM, "ON")
    field(OMSL, "closed_loop")
    field(DOL,  "$(user):GPIO18:IN")
}
record(bi, "$(user):GPIO18:IN") {
    field(DTYP, "devgpio")
    field(SCAN, "I/O Intr")
    field(INP,  "@18 BOTH")
    field(ZNAM, "OFF")
    field(ONAM, "ON")
    field(FLNK, "$(user):GPIO17:OUT")
}
```

実習２）python + epics

エディタを開いて以下のコードを作成。実行してみる。（kektaroの部分は自分のユーザー名で!）

LEDの点灯・消灯を繰り返す

```
import epics
import time

pv = epics.PV("kektaro:GPIO17:OUT")

print(pv.info)

for i in range(3):
    pv.put(1)
    time.sleep(1)
    pv.put(0)
    time.sleep(1)
```

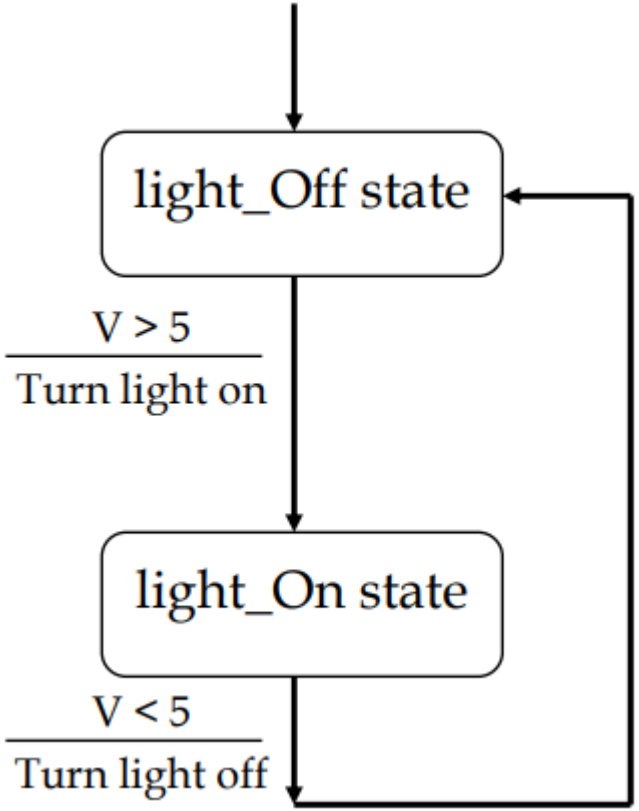
実習３：SNL/Sequencerを使ってLED制御

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
  }
}
```

example application で SNL を使う: 手順概略

- srcディレクトリのソース(sncExample.stt)を眺める
- configure/RELEASEファイルを編集

```
SNCSEQ = $(EPICS_BASE)/../modules/seq/2.2.9    <--- ここのコメントを外す
#ASYN = $(EPICS_BASE)/../modules/asyn/R4-43
#STREAM = $(EPICS_BASE)/../modules/stream/2.8.22
#AUTOSAVE = $(EPICS_BASE)/../modules/autosave/R5-10-2
#RPIGPIO = $(EPICS_BASE)/../modules/RPi/gpio/R2-0-1
```

- <appTop>でmake

- st.cmdの最後の行のコメントを外す

```
seq sncExample, "user=kektaro" <-- ここのコメント外す
```

実習４：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？

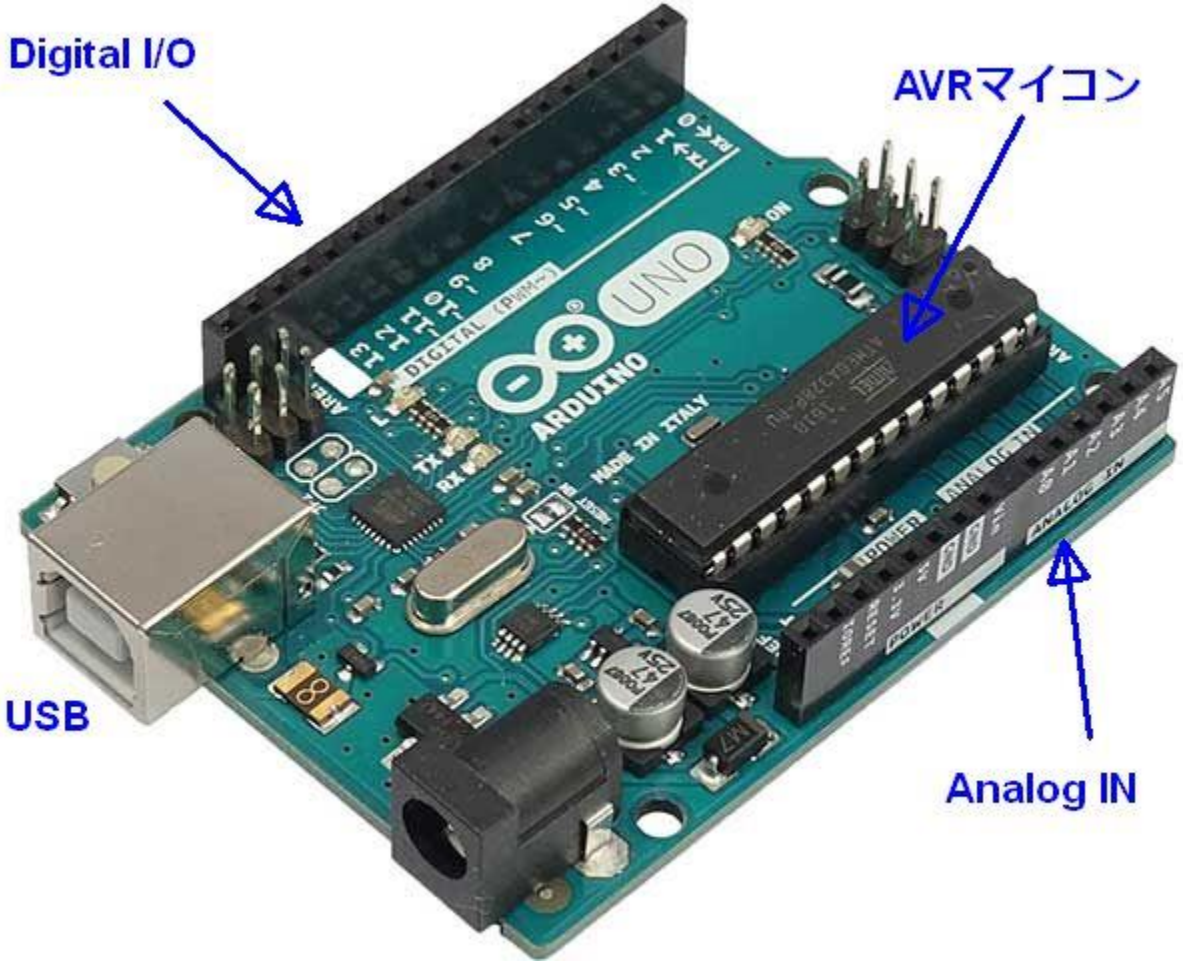
- ADC on Arduino <==[UART(RS232/USB)]==> RPi
- ADC + I2C ==[I2C]==> RPi
- ADC + SPI ==[SPI]==> RPi
- ADC + I2C ==[I2C]== [マイコン] UART(RS232/USB)]==> RPi

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

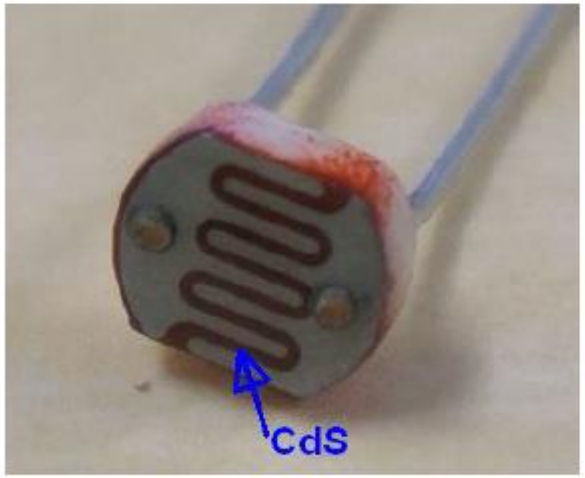
実習用の Arduino には、定期的にアナログ入力値やデジタル入出力を読んでシリアル経由で値を送るプログラムが既に書き込まれています。条件とコマンドは以下の通り。

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



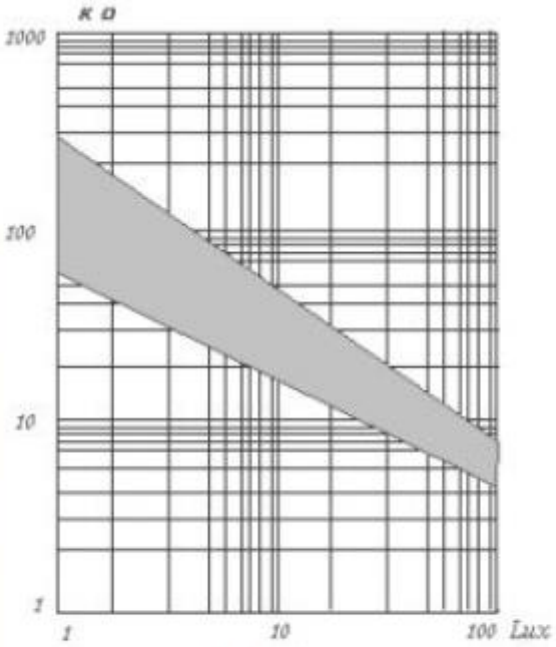


Fig. 3

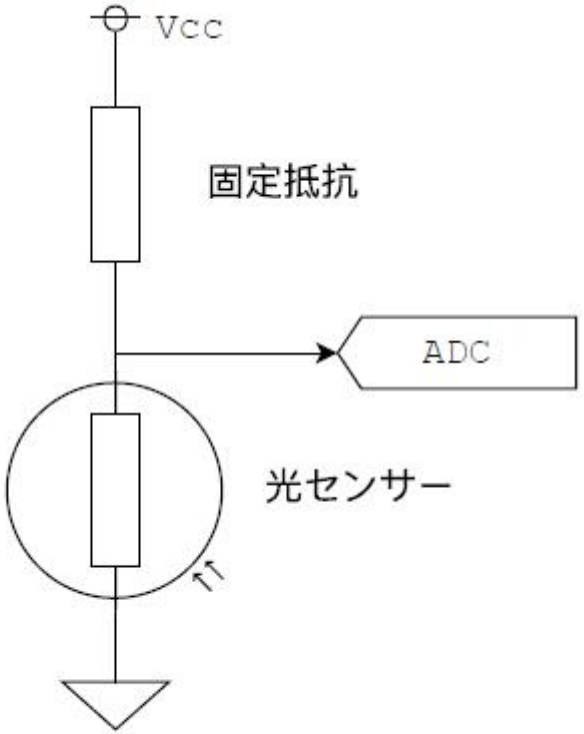


図 7　CdS センサーの写真と特性，回路例。

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:

- 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

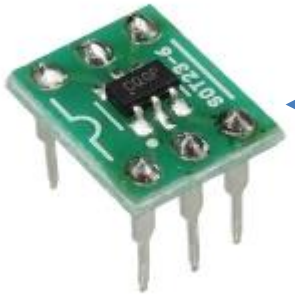
- dbl でリスト確認
- $(head):Arduino:LUM を camonitor , 手をかざす
- 該当するdbファイルと protocol ファイルを見る
- 他のコマンド(温度センサーなど)をみる

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

Vdd
SCL
SDA
GND



アナログ信号

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