組込システム　I
–7–

説明・演習
1. イベント検出
2. 電圧比較回路

入力の検出方法

- ポーリング
  - ➡ 入力装置の状態変化をソフトウェアに通知する場所を読み、状態変化をソフトウェア的に判断すること。
- 割り込み
  - ➡ 状態の変化をハードウェア的に検知し、プロセッサに割り込みを発生して、登録した処理を実行する。

ポーリング

- ポーリング
  - ➡ 一定時間間隔で入力状態を読み込む。

ポーリング例

```
（一部省略）
GPIO.setmode ( GPIO.BCM )
SW = 21                                  # 例えば、GPIO21にスイッチ
GPIO.setup ( SW, GPIO.IN, pull_up_down=GPIO.PUD_UP )
try:
    while True:
        if GPIO.input ( SW ) == 0:
            print ( "SW on" )
        else:
            print ( "SW off" )
        time.sleep(1)
except KeyboardInterrupt:
    pass
（一部省略）
```


イベント検出

- イベント（RPi.GPIO）
  - ➡ 特定のGPIO端子への入力信号が変化（入力エッジ）
  - ➡ イベント発生を検知して、登録した処理を実行

- イベント登録
  - ➡ どの端子のイベントか
  - ➡ どんなイベントか
    - ・ GPIO端子の入力エッジ（立ち上がり／立ち下がり）
  - ➡ どんな処理か
    - ・ def 文による関数定義

イベント検出

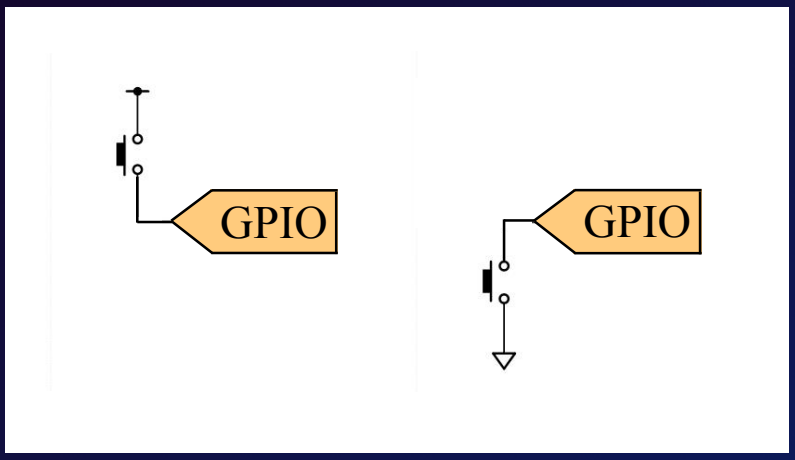
- イベント登録
  - ➡ GPIO.add_event_detect（ch, edge, callback, bouncetime）
    - ch：イベント登録するGPIO端子番号
    - edge：イベント発生の信号のエッジ
      - ➡GPIO.RISING（立ち上がり）
      - ➡GPIO.FALLING（立ち下がり）
      - ➡GPIO.BOTH（両方）
    - callback：呼び出す関数
    - bouncetime：次のイベント検出までの時間（ms）
- イベント解除 GPIO.remove_event_detect(ch)

イベント検出

- イベント検出
  - ➡ GPIO端子の状態変化でcallback関数実行

イベント例(1/2)

```
    （一部省略）
GPIO.setmode ( GPIO.BCM )
SW = 21                                  # 例えば、GPIO21にスイッチ
GPIO.setup ( SW, GPIO.IN, pull_up_down=GPIO.PUD_UP )
s = 0
def checkSW ( pin ):                     # pin：イベント発生ピン（同じ関数を複数のイベントで呼出）
    global s                             # 関数外の変数（グローバル）を使用する
    s = 1
GPIO.add_event_detect ( SW, GPIO.FALLING, callback=checkSW, bouncetime=200 )
try:
    while True:
        if s==1:
            print('SW on')
            s = 0
        else:
            print('SW off')
        time.sleep(1)
    （一部省略）
```



イベント検出

- イベント検出
  - ➡ GPIO端子の状態変化でcallback関数実行

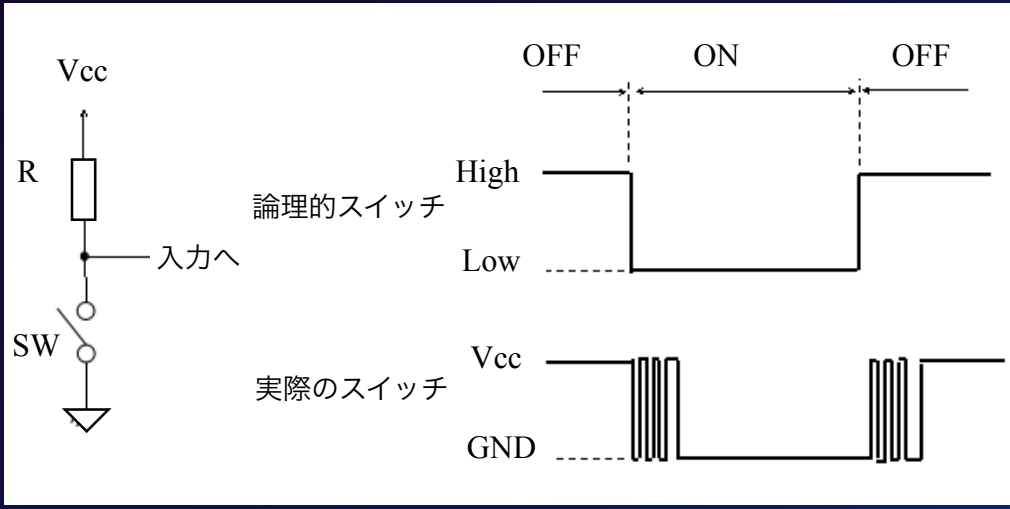
イベント例(2/2)

```
except KeyboardInterrupt:
    pass

GPIO.remove_event_detect ( SW )
GPIO.cleanup( )
```

チャタリング

- スイッチ変化時の連続的な ON / OFF
  - ➡ チャタリングによる繰り返しのイベント発生を回避する.（bouncetime使用例）



演習（1）

- SW入力をイベントを利用して検出する.

```
（一部省略）

GPIO.setmode ( GPIO.BCM )
SW = 21
GPIO.setup ( SW, GPIO.IN, pull_up_down=GPIO.PUD_UP )
s = 0

def checkSW ( pin ):
    global s
    s = 1

GPIO.add_event_detect ( SW, GPIO.FALLING, callback=checkSW, bouncetime=200 )
try:
    while True:
        if s==1:
            print('SW on')
            s = 0
        else:
            print('SW off')
        time.sleep(1)

（一部省略）
```

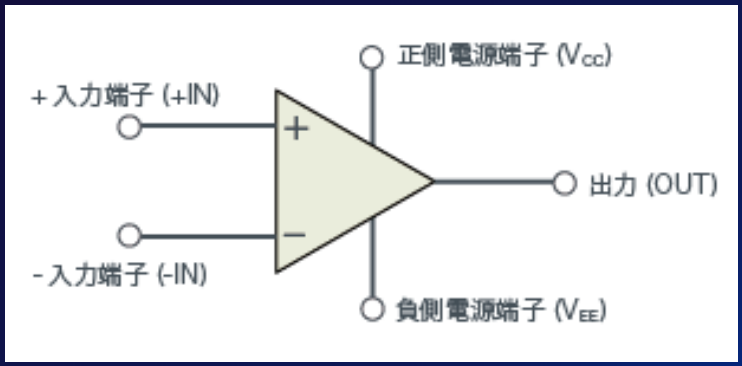
ループ内では入力を読み込まない

回路はプログラムから各自考える
- **21番端子を使うらしい**
- **プルアップ設定を使うらしい**
- **信号の立ち下がりを使うらしい**

注意：大文字 I（アイ）と小文字 l（エル）

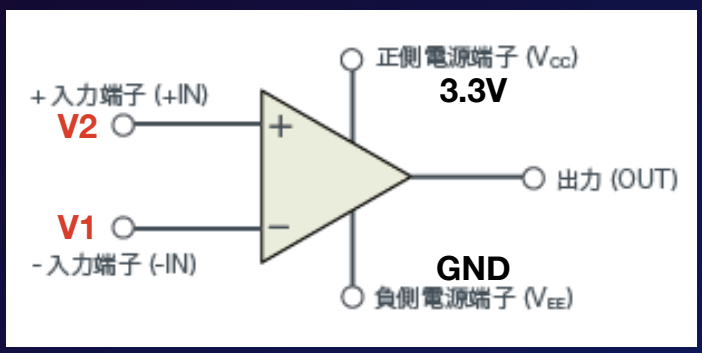
電圧比較回路

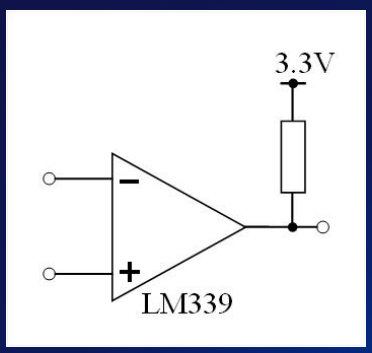
- オペアンプ（OperationalAmplifier/演算増幅器）
  - 2入力差動増幅器（2つの信号の差を増幅）
  - 応用例
    - ➡増幅回路
    - ➡フィルタ
    - ➡比較回路



電圧比較回路

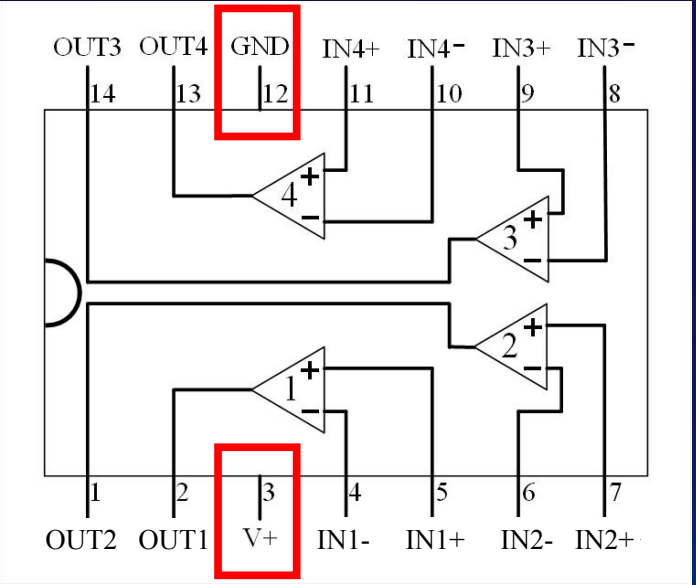
- 比較回路（LM339は電圧比較用IC）
  - V2 － V1 ＞ 0　出力 V_{CC} （High）
  - V2 － V1 ＜ 0　出力 V_{EE} （Low）
- 注意
  - ➡ LM339はオープンコレクタタイプ／プルアップ抵抗**必須**





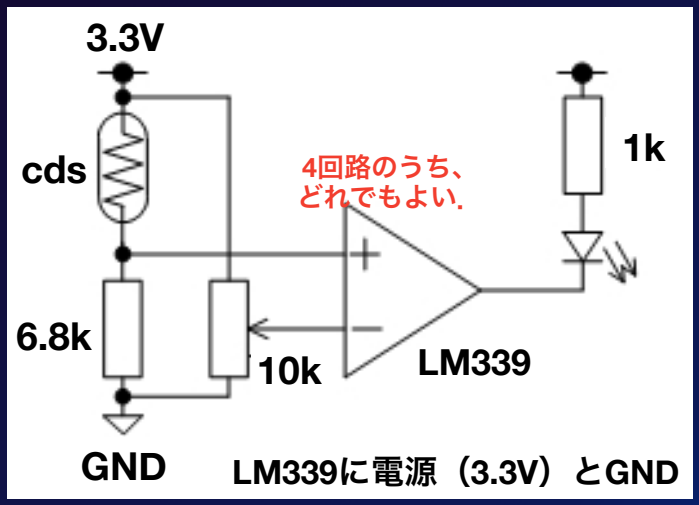
電圧比較回路（LM339）

- 4回路電圧比較器



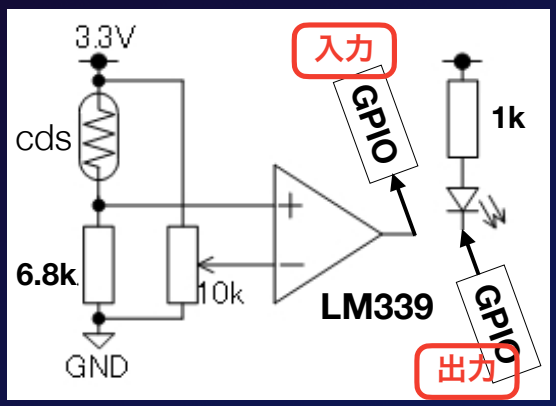
演習（2）

- 下図の回路を作製しなさい.
  - 可変抵抗を調整して、cds素子への光を遮った時と遮らない時とでLEDの点滅状態が変わるようにしなさい.　（可変抵抗を調整する）



演習（3）

- LEDを比較回路から切り離し、GPIOの出力に繋ぎ、比較回路の出力をGPIOの入力に繋ぎなさい.



演習（3）

- LEDを比較回路から切り離し、GPIOの出力に繋ぎ、比較回路の出力をGPIOの入力に繋ぎなさい.
- cds素子により、暗くなったことを検知した時にLEDを点灯し、SWを押すことでLEDを消灯するシステム（プログラム）を設計、製作しなさい. cds素子（比較回路出力）、SWの変化の検知にはイベント検出を用いること.

➡ ヒント
- 回路は演習（1）（2）を参考に.
- 比較器出力はオープンコレクタである.
- 比較器出力はエッジに気をつけてSWと同様に扱う.

問い

- オープンコレクタとはどの様なものか.

課題

- 演習(1)（(3)について全体が把握できる様に詳細にまとめること．最低限、以下の項目は含むこと．（課題～考察で1200字以上）
  - ・表紙（授業名、学籍番号・氏名、提出日）
  - ‒ ‒ ‒ ‒ ‒ ‒ ‒ ‒ ‒ ‒
  - ・本文
    - ➡課題
    - ➡使用部品
    - ➡回路の説明（RaspberryPiの端子を明記）
    - ➡アルゴリズムの説明
    - ➡結果（説明を工夫すること）
    - ➡考察
    - ➡問いの解答
    - ➡参考文献（任意）
  - ‒ ‒ ‒ ‒ ‒ ‒ ‒ ‒ ‒
  - ・図表
    - ➡回路図と回路の写真は必須

レポー作成上の注意

- 締め切り 6月5日（厳守）
- 提出物
  - レポート(学籍番号.pdf)
  - 演習（1）（3）ソースファイル（＊＊＊.py）

- 注意
  - 実体配線図は回路図として認めない.
  - 授業資料のコピペ、流用は認めない.
  - レポートはA4版としてPDFファイルで提出する.
  - 印刷して適切なレポートであること.
  - キャプチャ画像等の文字は読めるものであること.
  - PDFファイルに変換後、必ず、確認すること.

次回

- LED ダイナミック点灯