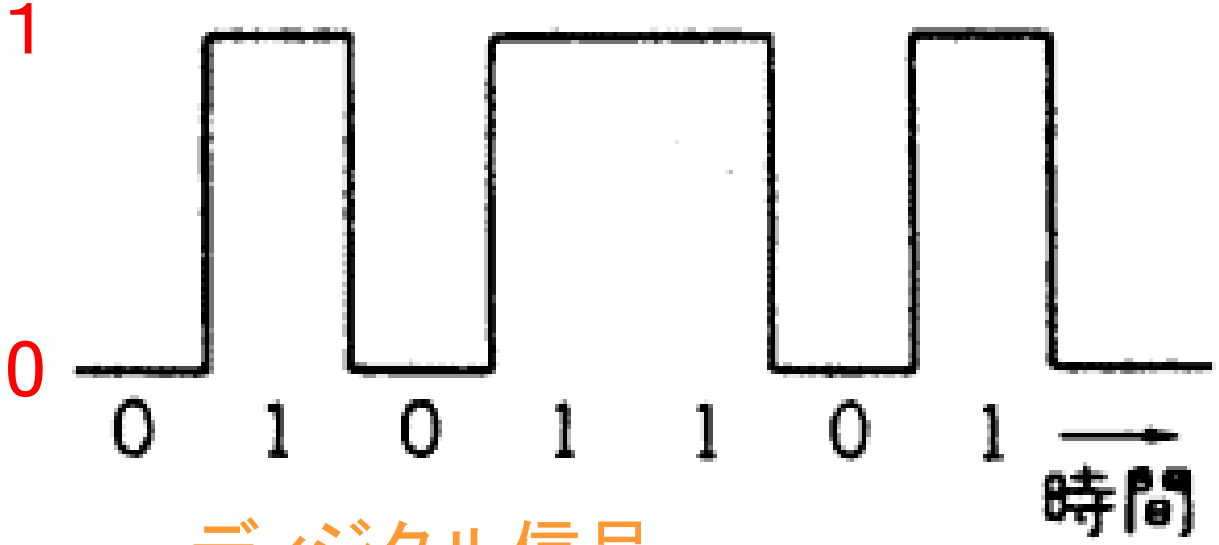
桐生高校SSH

プログラミング講座
（アナログ信号編）

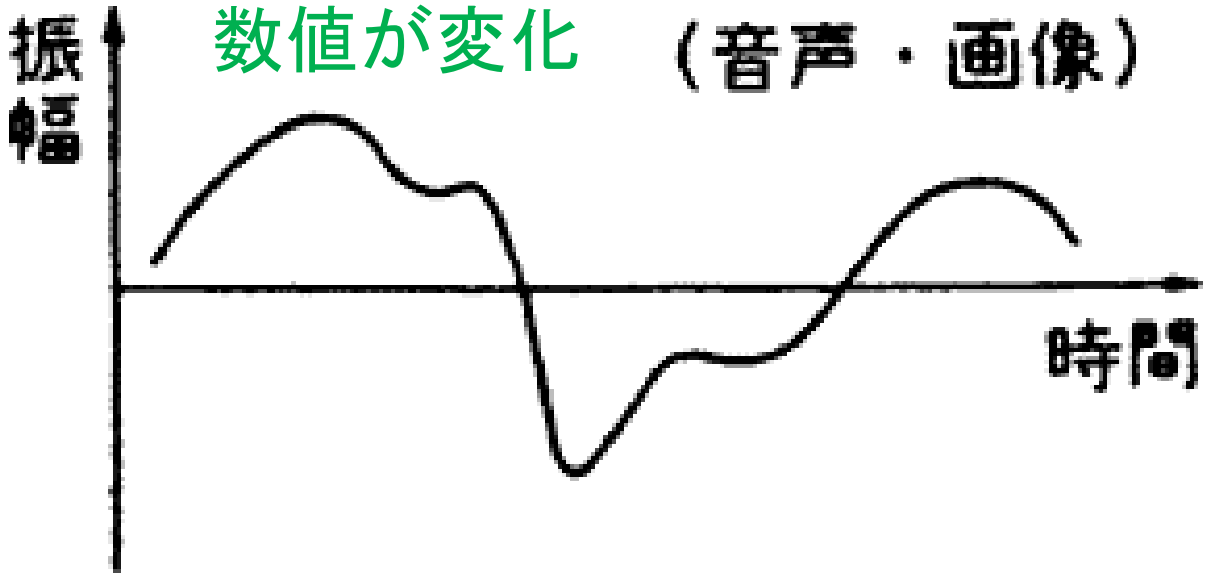
センサの出力信号

0か1のみ
（第1回、第2回の講座で行った）



ディジタル信号
（コンピュータが直接扱える）

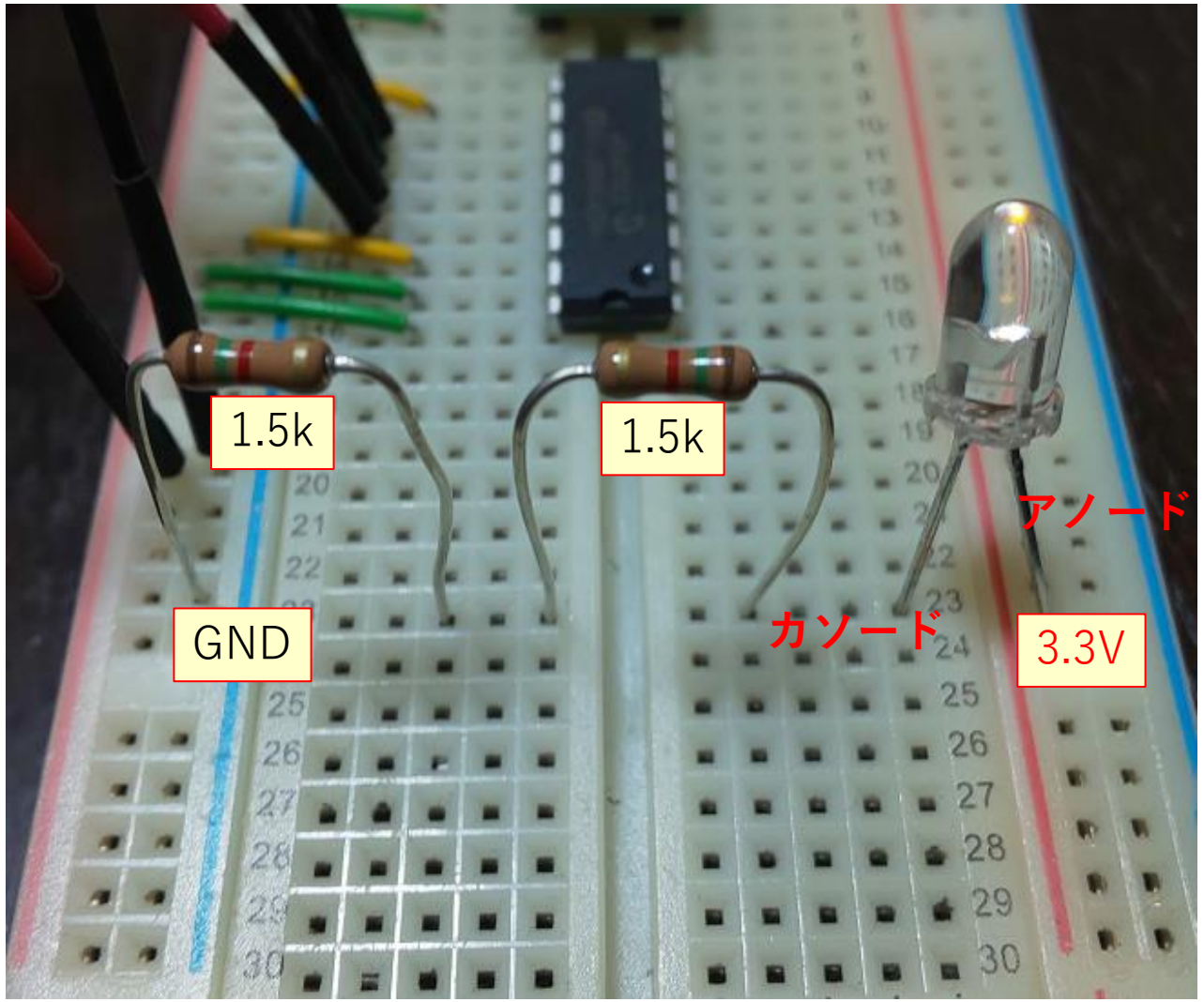
数値が変化

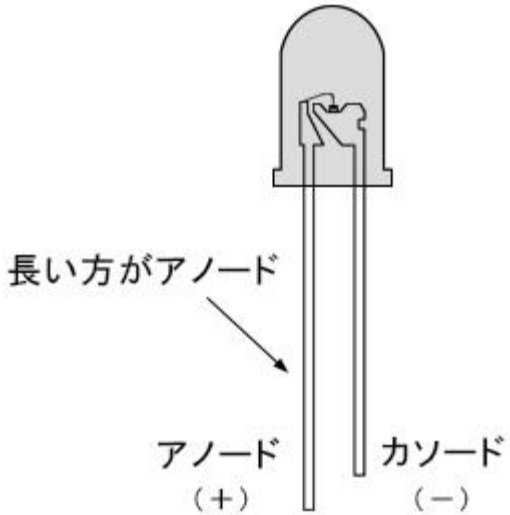


アナログ信号
（コンピュータが直接扱えない）

照度センサ（Cdsセル）の接続

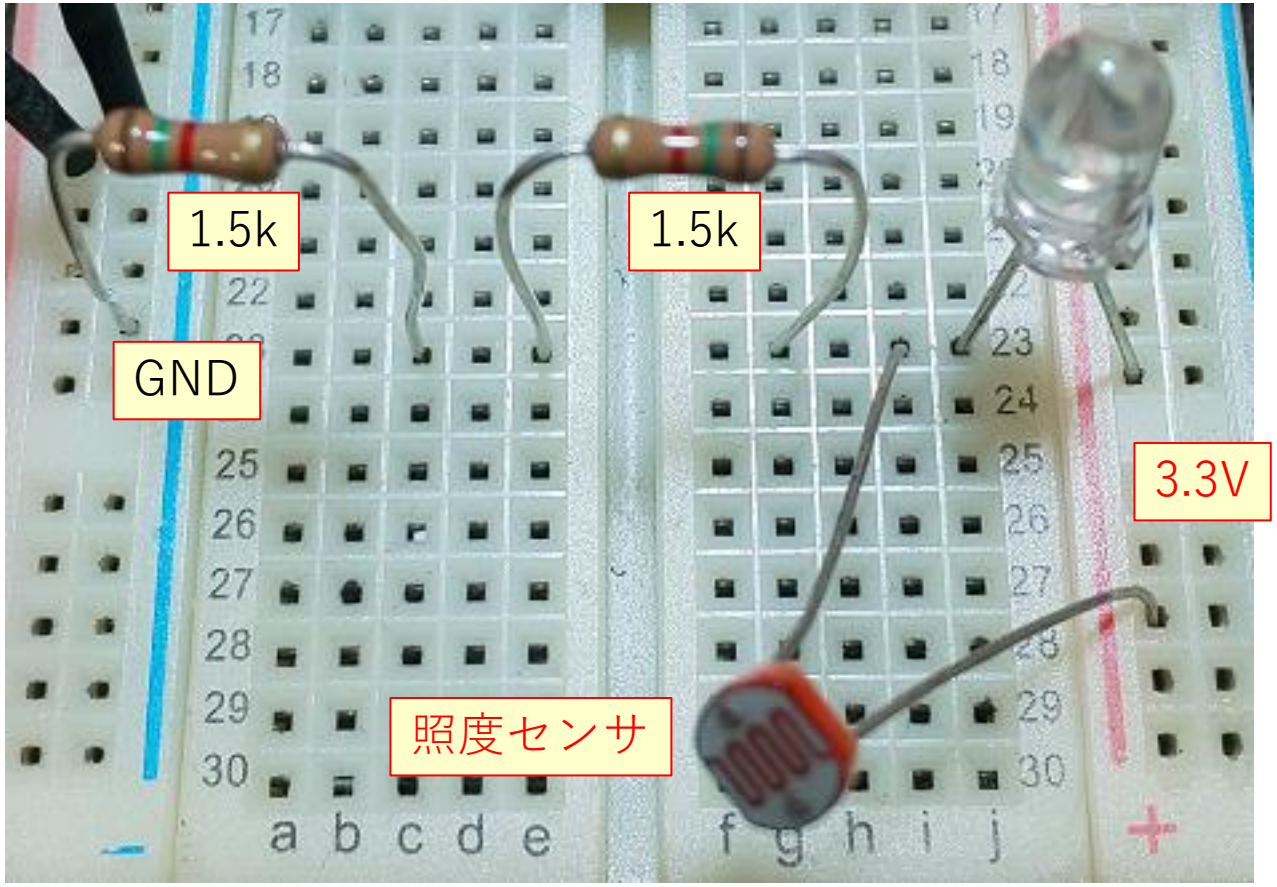
※センサが3.3V仕様のものがあるため、必ず3.3Vに接続





照度センサ（Cdsセル）の接続

※センサが3.3V仕様のものがあるため、必ず3.3Vに接続



[確認]
照度センサを手で覆い被せてみよう。

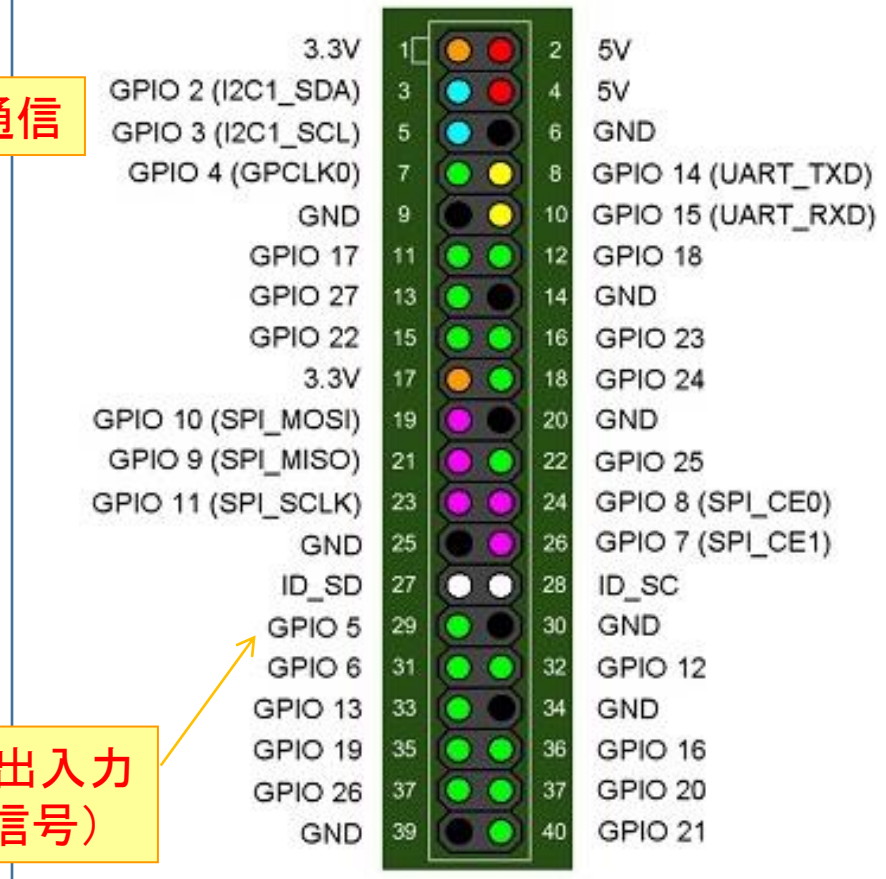
LEDに加わる電圧が変化

⇩

アナログ信号

Raspberry Pi仕様

GPIO

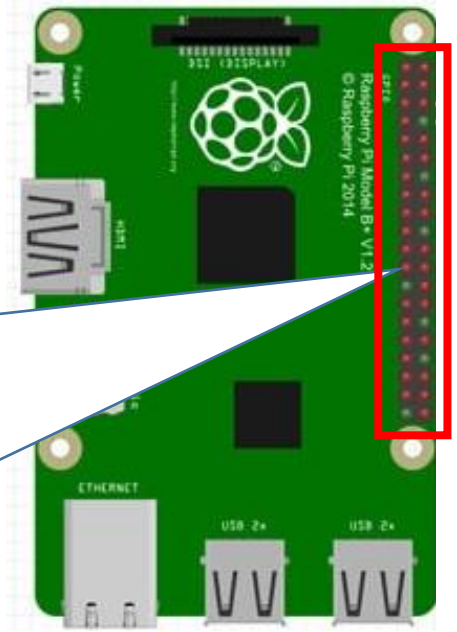


I2C通信

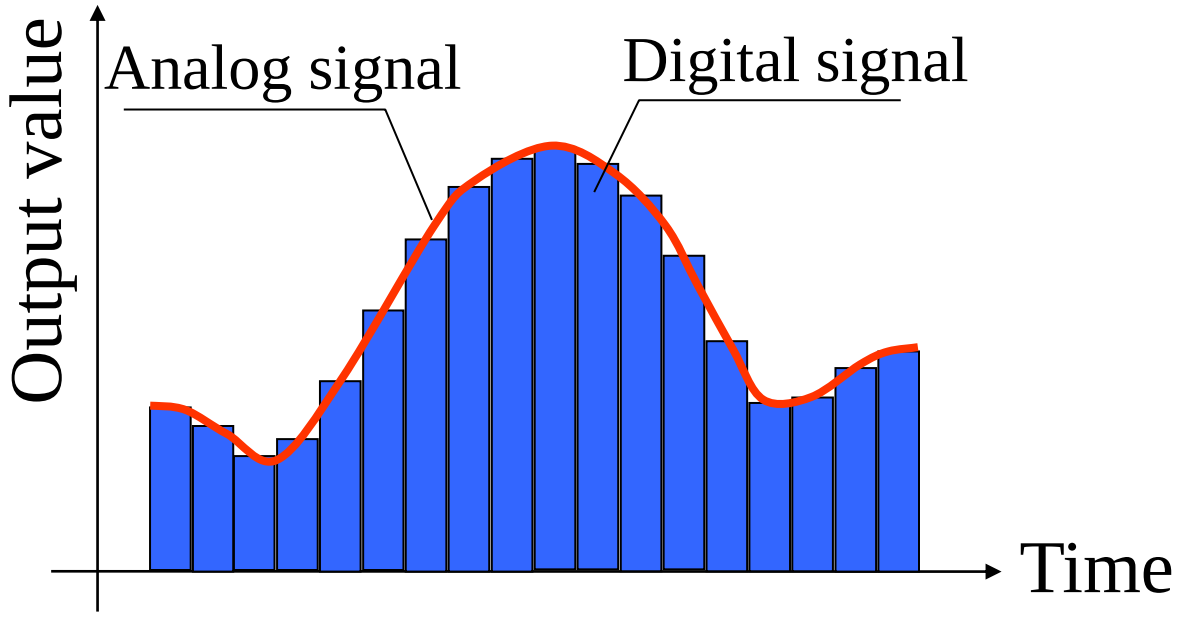
シリアル通信

ディジタル出入力
（0 or 1の信号）

アナログ信号を扱えない



アナログ信号からディジタル信号への変換



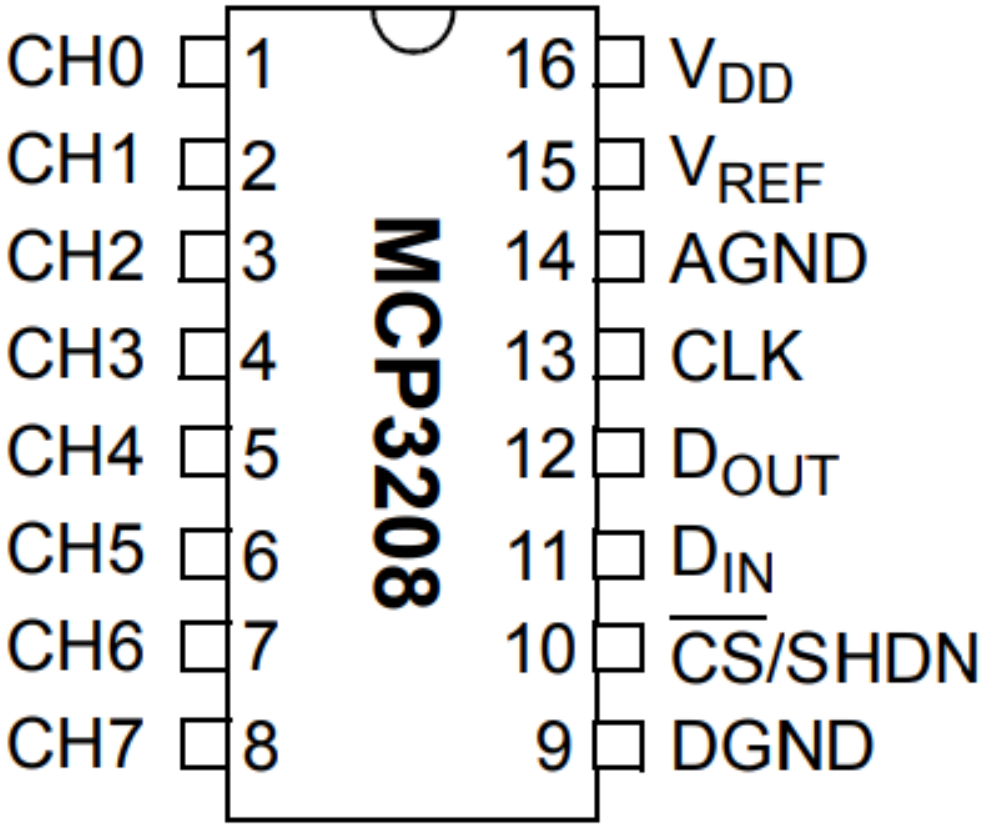
アナログ信号
（センサの出力電圧）
連続信号

変換が必要

AD変換器

ディジタル信号
（コンピュータが扱える）
離散信号

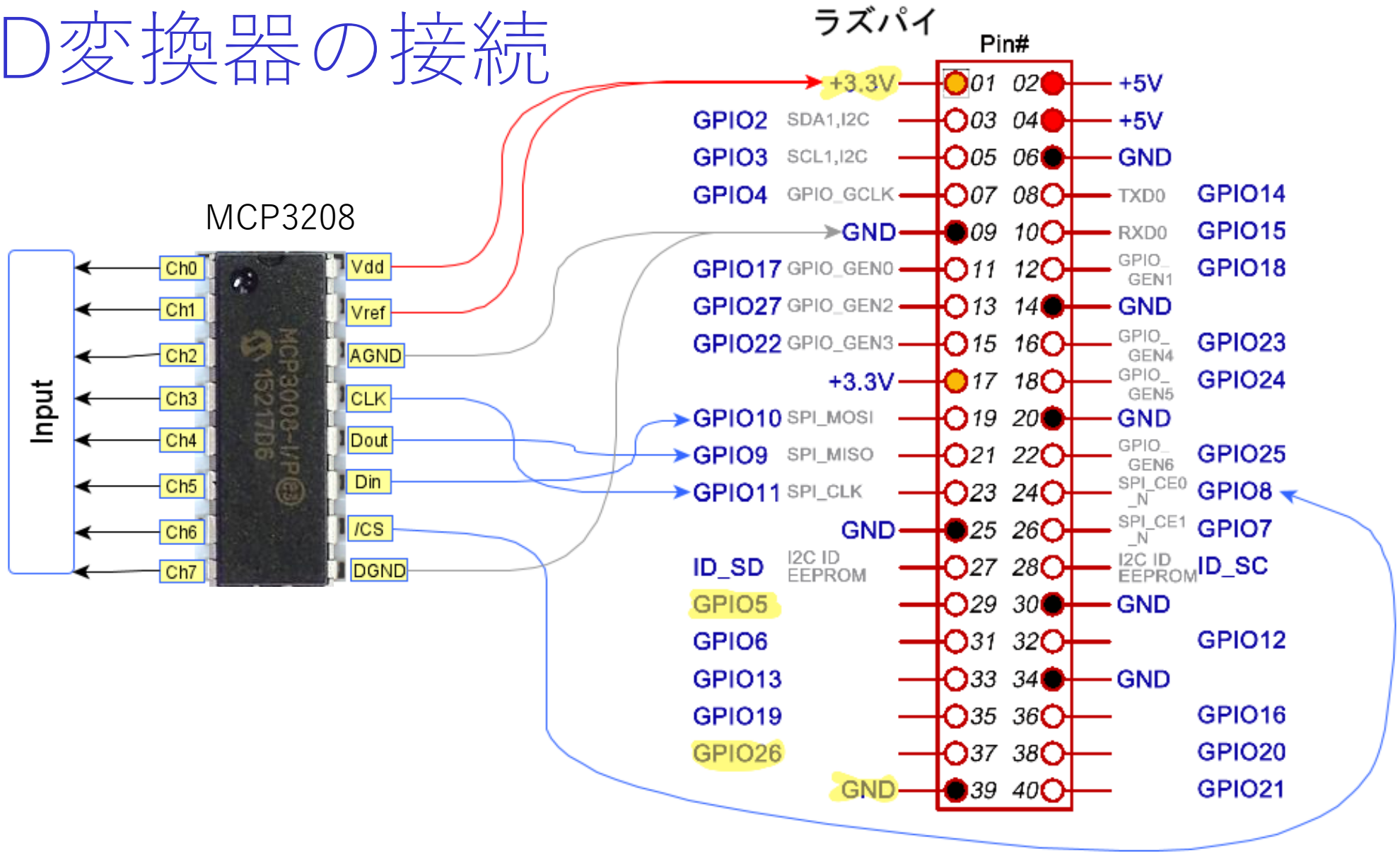
AD変換器(MCP3208)



12ビットAD変換器

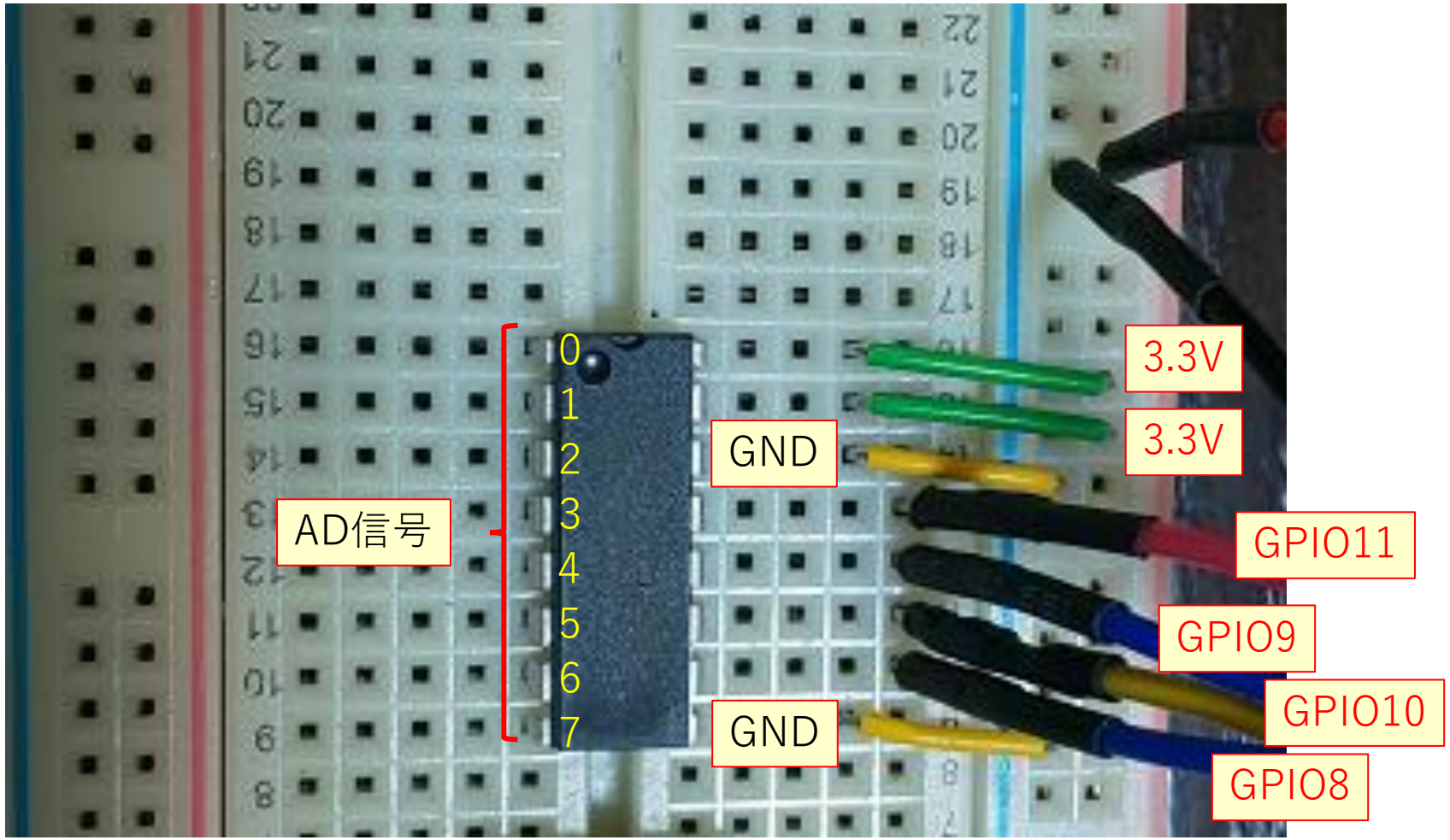
MCP3008	名称	役割
1	CH0	アナログ電圧入力
2	CH1	アナログ電圧入力
3	CH2	アナログ電圧入力
4	CH3	アナログ電圧入力
5	CH4	アナログ電圧入力
6	CH5	アナログ電圧入力
7	CH6	アナログ電圧入力
8	CH7	アナログ電圧入力
9	DGND	デジタルGND
10	CS/SHDN	チップセレクト/シャットダウン
11	Din	シリアルデータIN
12	Dout	シリアルデータOUT
13	CLK	シリアルクロック
14	AGND	アナログGND
15	Vref	リファレンス電圧
16	Vdd	電源電圧 (+2.7～5.5V)

AD変換器の接続

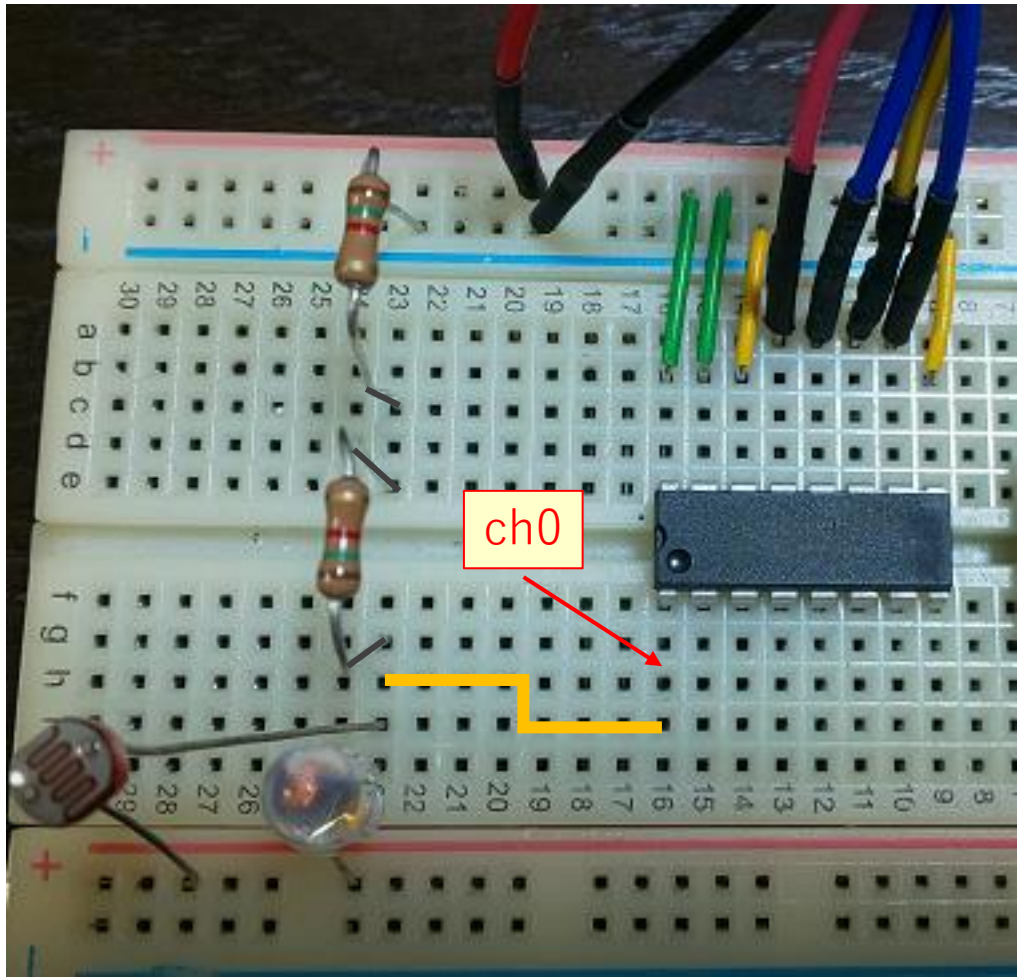

ラズパイ
Pin#
MCP3208
Input
Ch0
Ch1
Ch2
Ch3
Ch4
Ch5
Ch6
Ch7
Vdd
Vref
AGND
CLK
Dout
Din
/CS
DGND
+3.3V
GPIO2 SDA1,I2C
GPIO3 SCL1,I2C
GPIO4 GPIO_GCLK
GND
GPIO17 GPIO_GEN0
GPIO27 GPIO_GEN2
GPIO22 GPIO_GEN3
+3.3V
GPIO10 SPI_MOSI
GPIO9 SPI_MISO
GPIO11 SPI_CLK
GND
ID_SD I2C ID EEPROM
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
GND
01 02 +5V
03 04 +5V
05 06 GND
07 08 TXD0 GPIO14
09 10 RXD0 GPIO15
11 12 GPIO_GEN1 GPIO18
13 14 GND
15 16 GPIO_GEN4 GPIO23
17 18 GPIO_GEN5 GPIO24
19 20 GND
21 22 GPIO_GEN6 GPIO25
23 24 SPI_CE0_N GPIO8
25 26 SPI_CE1_N GPIO7
27 28 I2C ID EEPROM ID_SC
29 30 GND
31 32 GPIO12
33 34 GND
35 36 GPIO16
37 38 GPIO20
39 40 GPIO21

AD変換器の接続

※センサが3.3V仕様のものがあるため、必ず3.3Vに接続



AD変換器への接続とソフトウェア（1）



Analog_ADの中の
ad_3208_ch1.pyを
利用します。

[確認]
照度センサを手で覆い
被せたときの出力値の
変化を見てみよう。

ad_3208_ch1.py

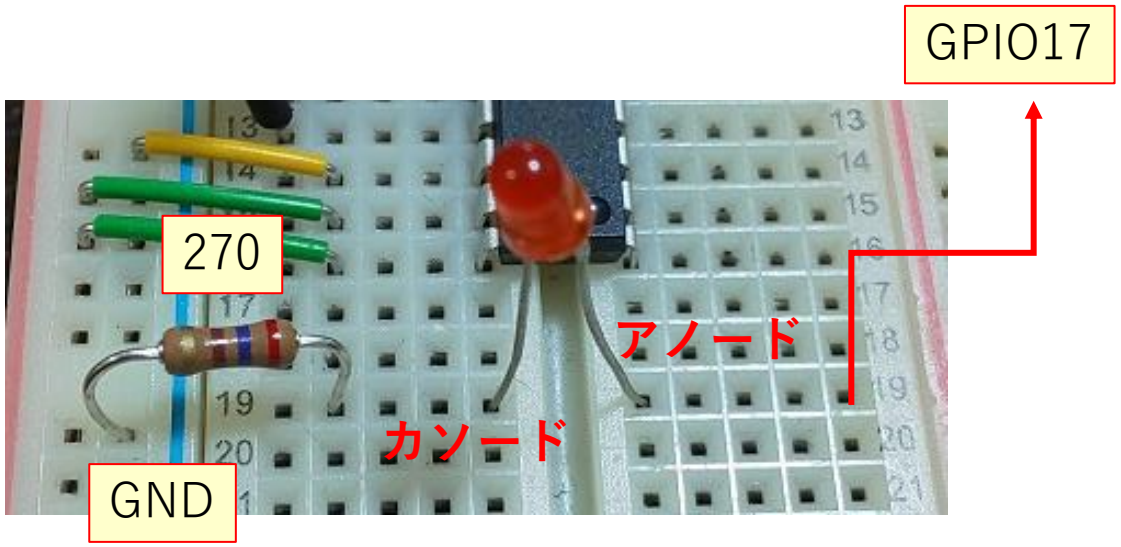
```
from gpiozero import MCP3208
from time import sleep

data = []
Max=4096 #12bit : 2^12=4096

count=0

for i in range(1): # Number of channel
    data.append(MCP3208(channel=i))
while True:
    ch_0=int(Max* data[0].value)
    print(count, ch_0)
    sleep(0.05) #Sampling speed
    count+=1
```

LEDの点灯への反映

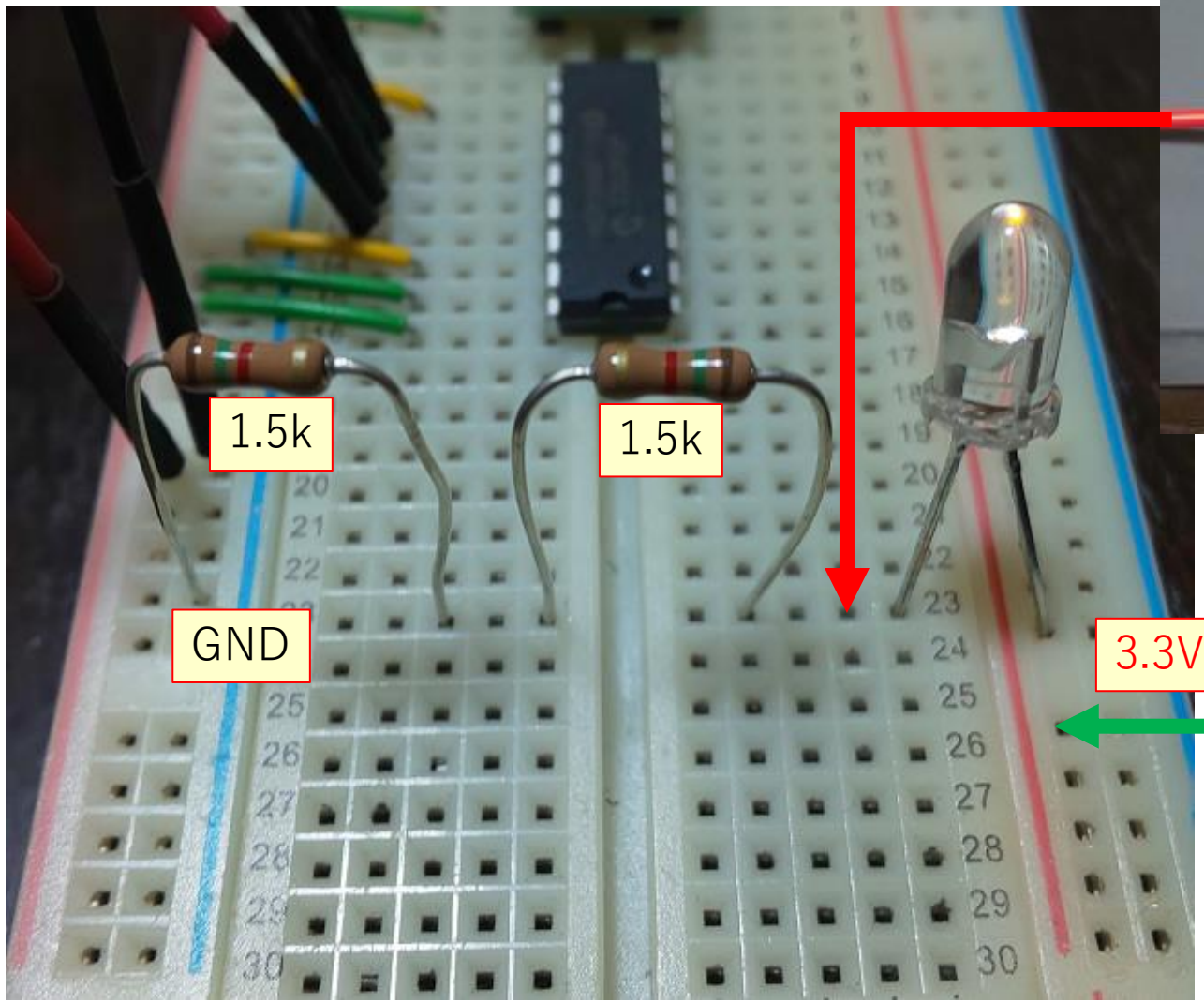


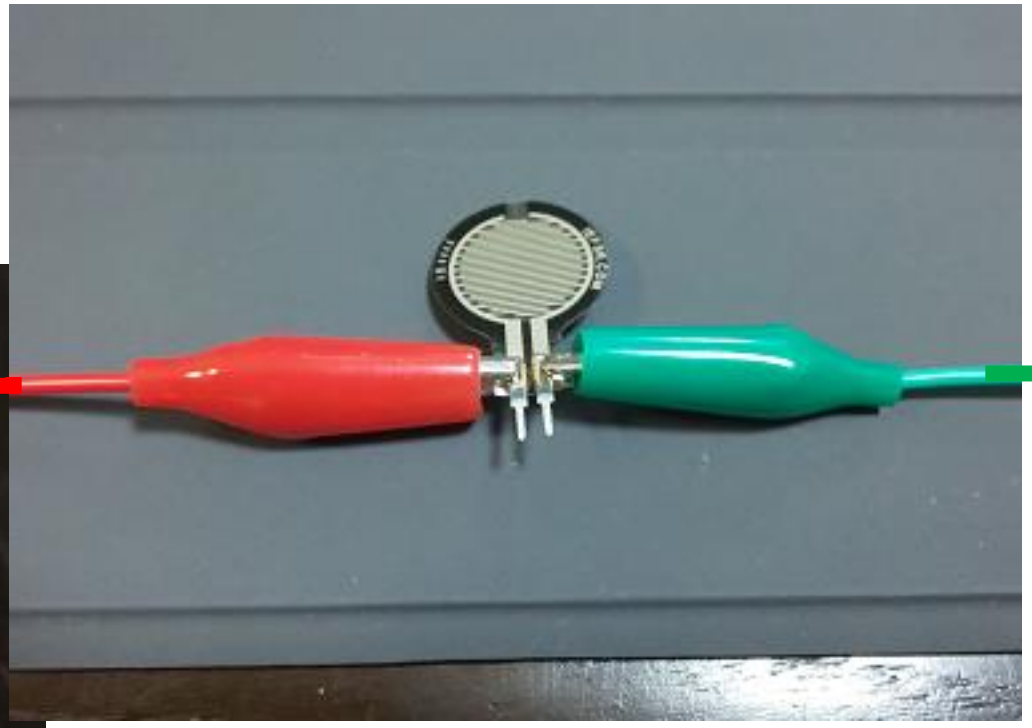
ON-OFFの境界（しきい値）を決定しましょう。

[課題] 照度センサを手で覆うとLEDが点灯するプログラムを作成しましょう。

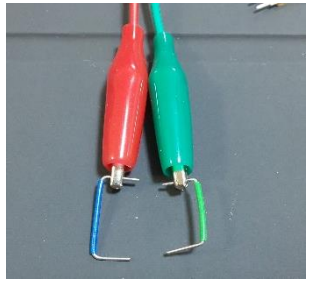
圧力センサ

ターミナルから
cd Desktop
cd Analog_AD



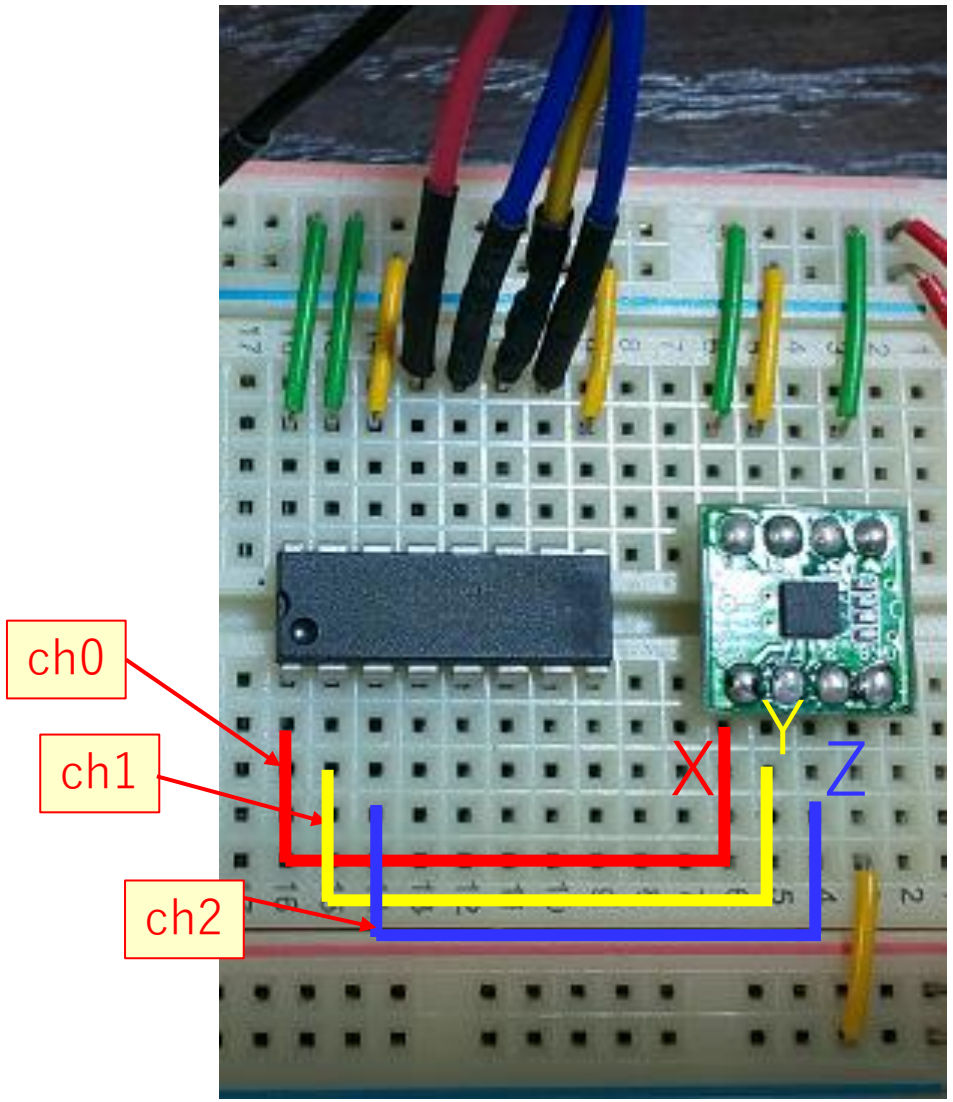


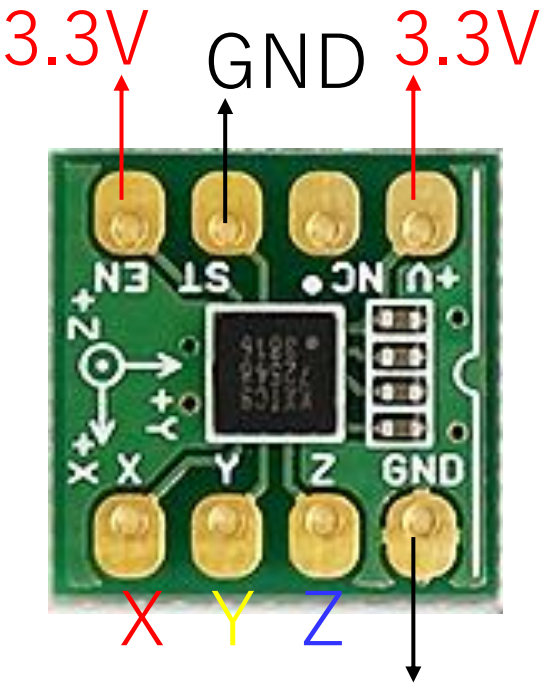
※AD変換器（MCP3208）の周辺回路はそのままにします。
※照度センサと同じ位置に圧力センサを接続してください。



※ワニ口クリップを活用しましょう。

加速度センサ

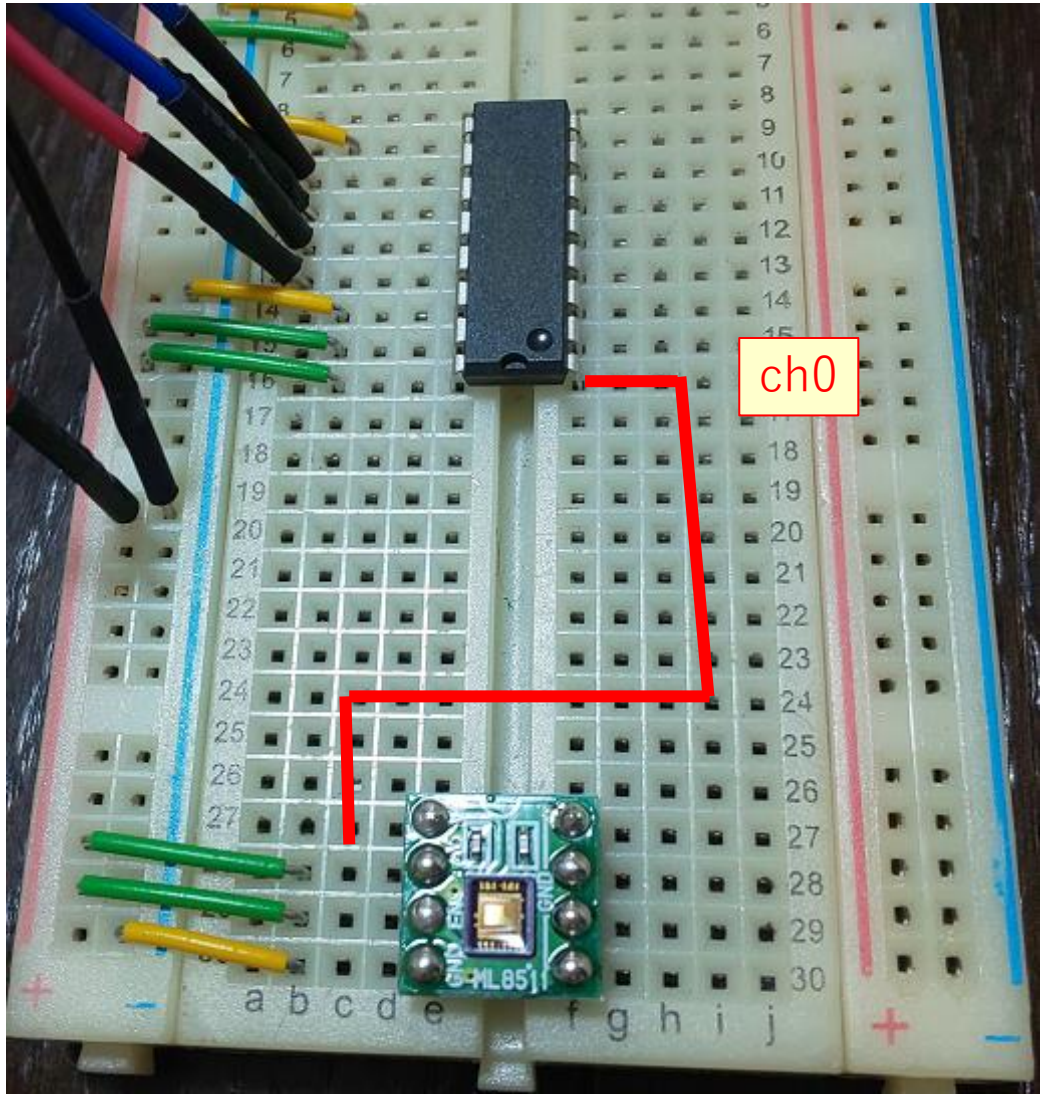


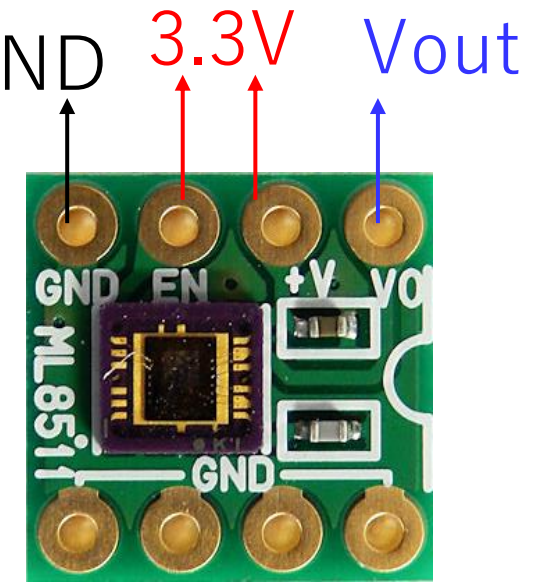


- 電源入力：3.3V（標準）
- 感度：660mV/g（標準）
- 0g 出力:1.65V（標準）

[確認] **ad_3208_ch1. py**を利用して、入力チャンネルを3つにしよう。

紫外線センサの接続





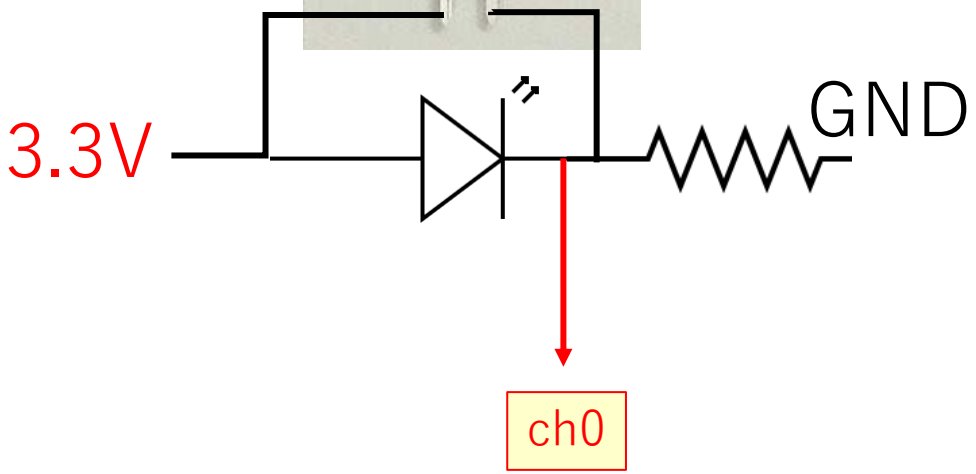
■ピン番号表■

番号	名称	動作
1	Ｖｏ	出力（負荷抵抗１００ＫΩ以上推奨）
2	ＶＤＤ	電源＋（標準３．３Ｖ）
3	ＥＮ	イネーブル（Ｈ＝動作モード、Ｌ＝待機モード）
4～8	ＧＮＤ	グランド

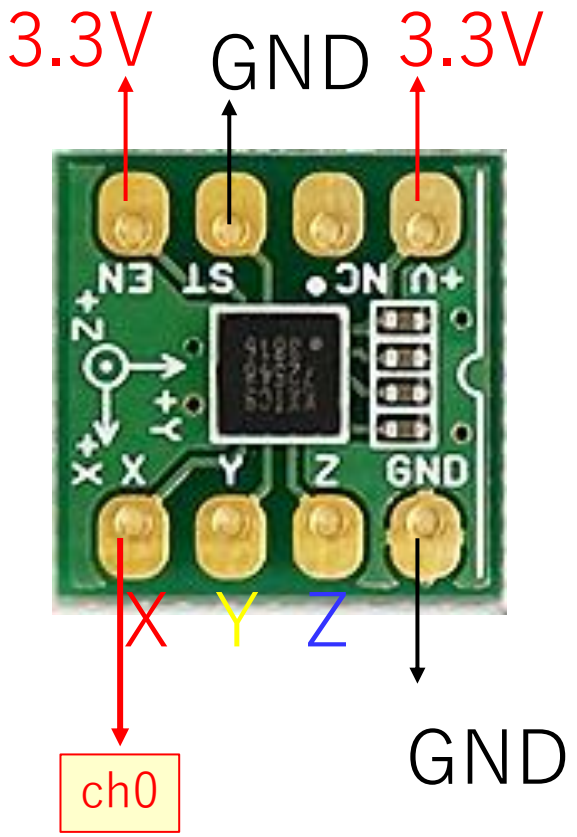
[確認]**ad_3208_ch1. py**を利用して、紫外線の有無による値の変化を見てみよう。

センサの出力信号

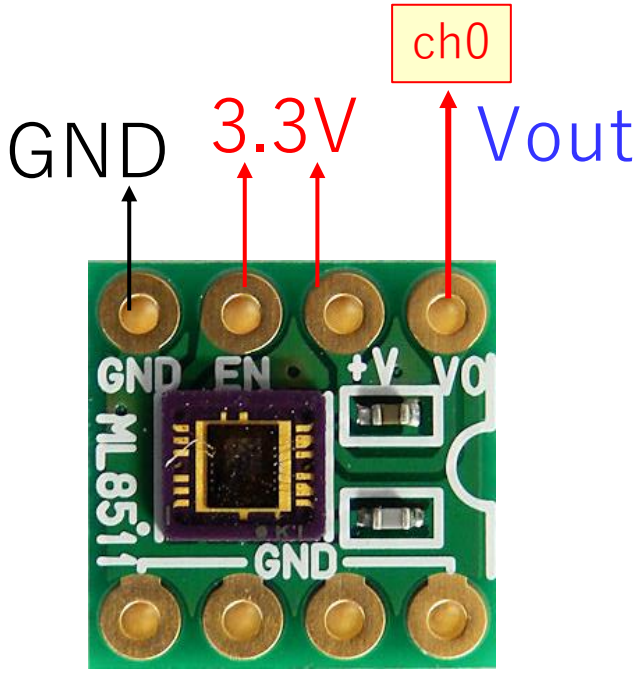
(1) 圧力センサ



(2) 加速度センサ



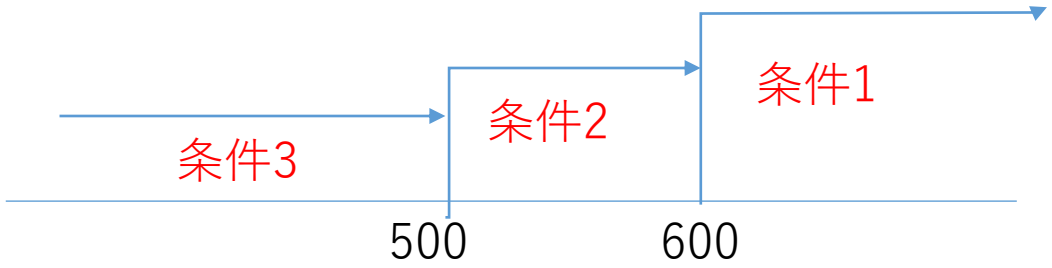
(3) 紫外線センサ



計測値の大きさでLEDの点灯を制御

（ヒント）例えば、消灯（力を加えていない）が600、
少し力を加えるときを500以上、さらに力を加えるときを
500未満とすれば、

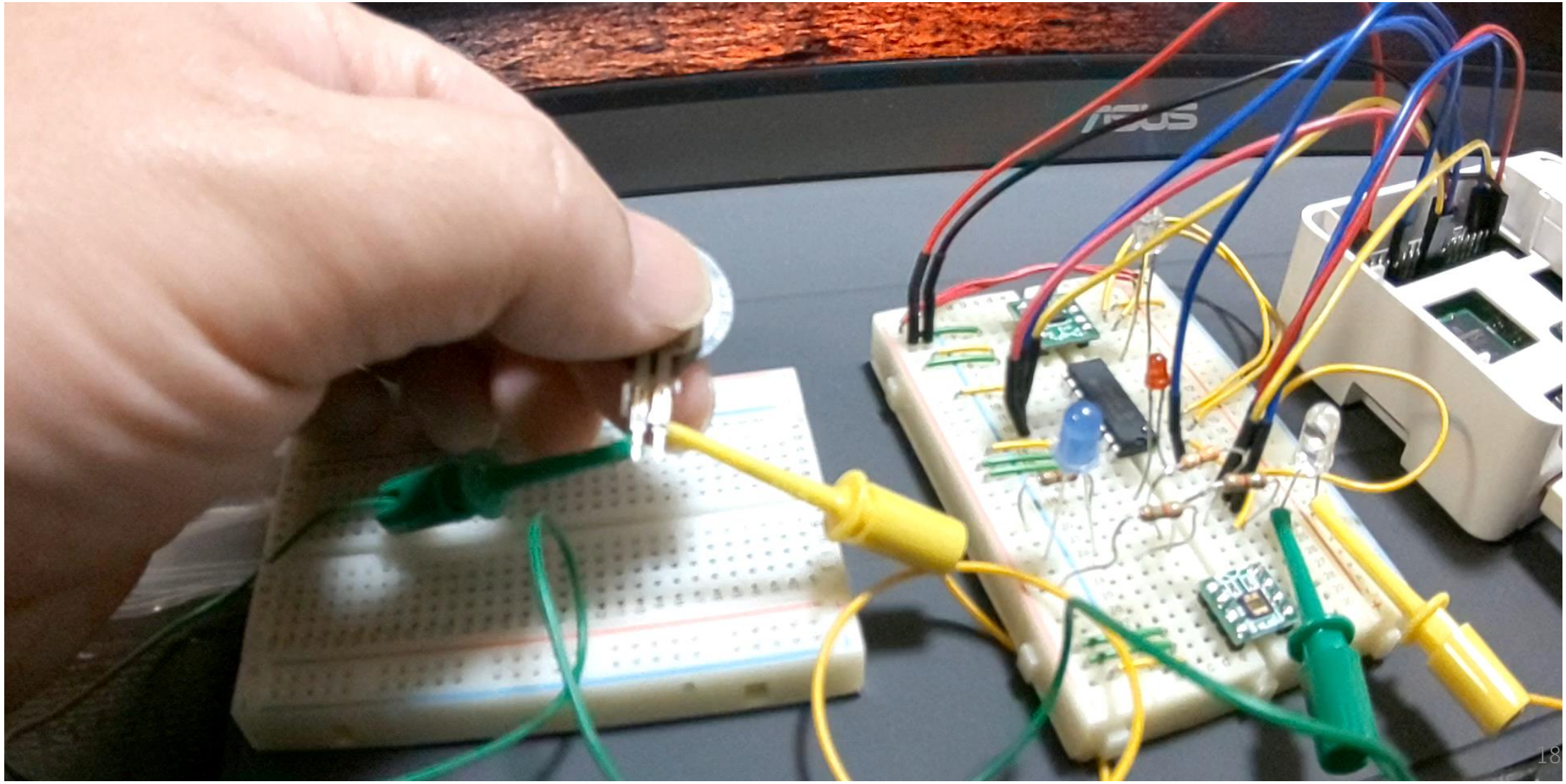
```
If value1 > =600:     ※条件1
        print(“力を加えていない”)
elif value1>=500:     ※条件2
        print(“弱い”)
else:                          ※条件3
        print(“強い")
```



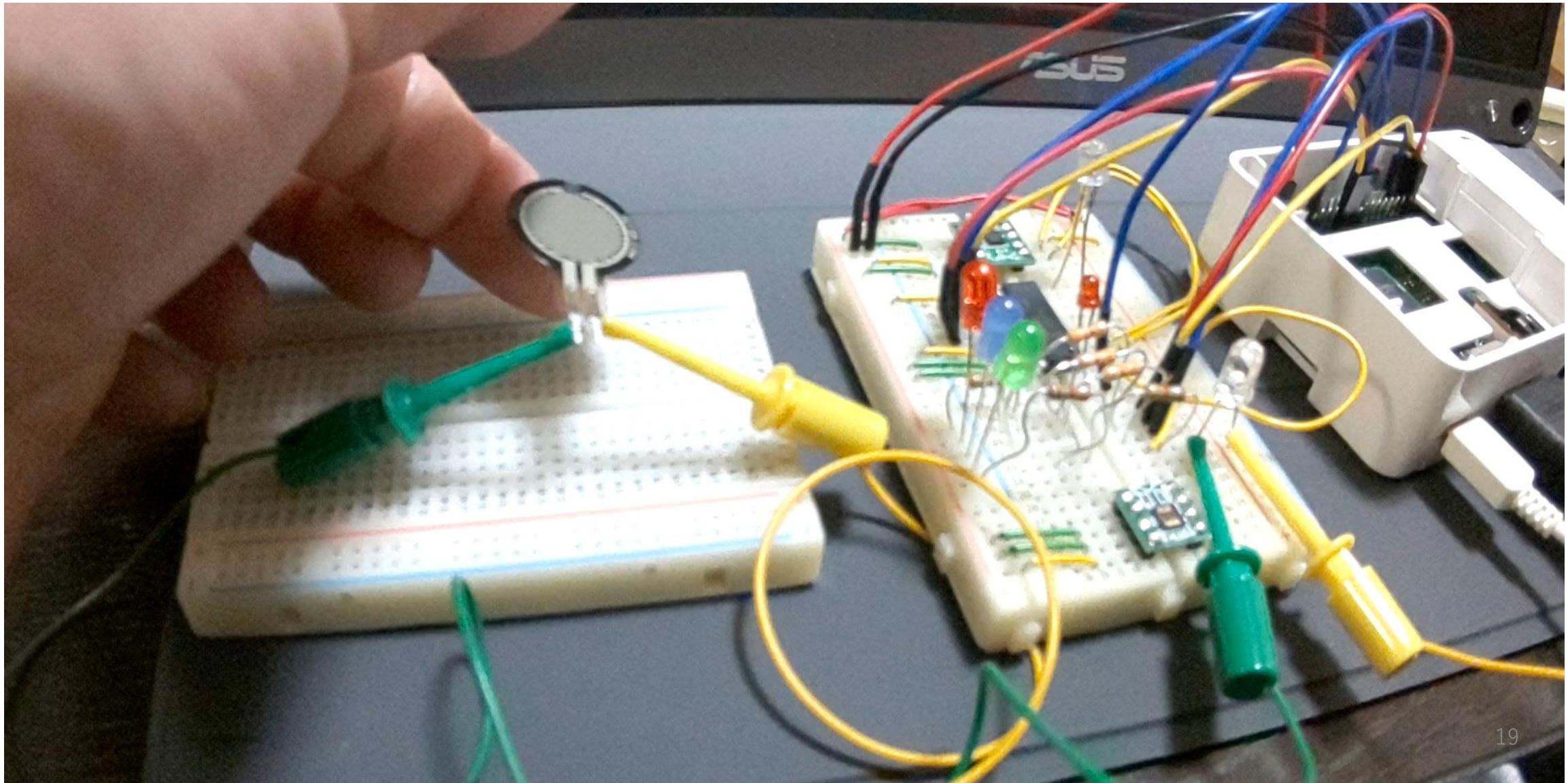
※<u>3段階の場合</u>

```
if value1 > =600:
        print(“力を加えていない")
elif value1>=550:
        print(“弱い")
elif value1>=500:
        print(“普通")
else:
        print(“強い")
```

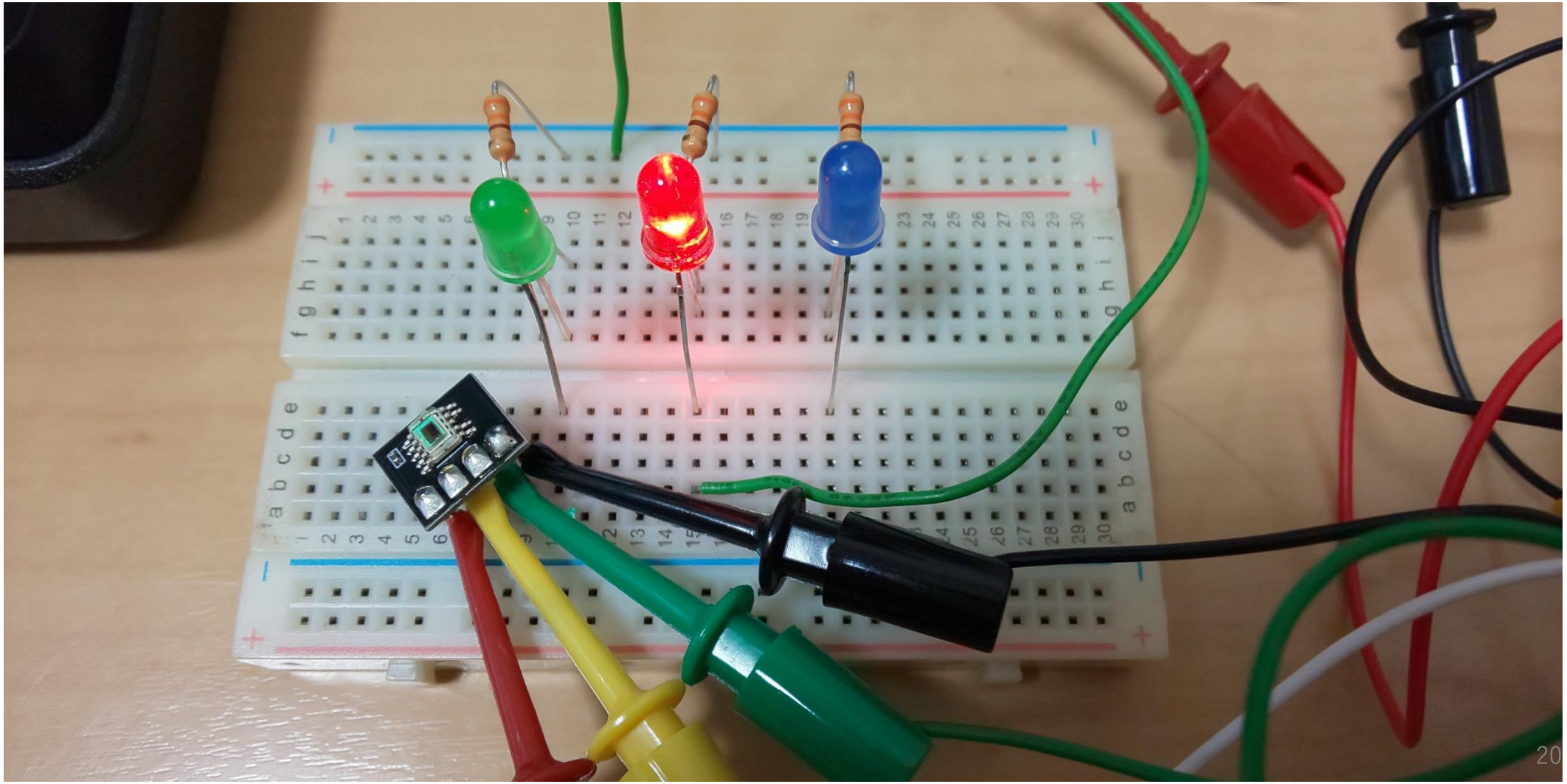
圧力センサ（1個）



圧力センサ（3個）



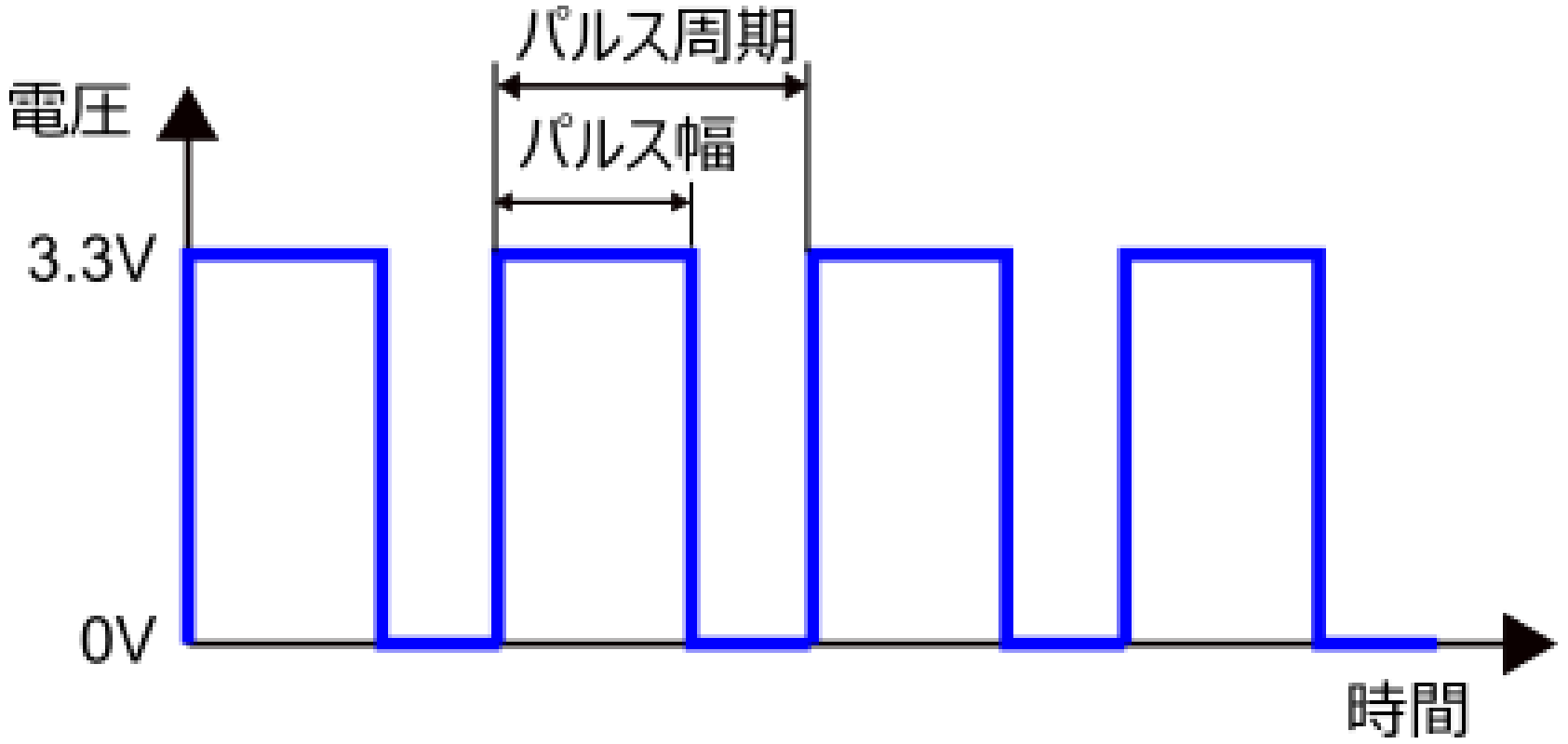
色センサ



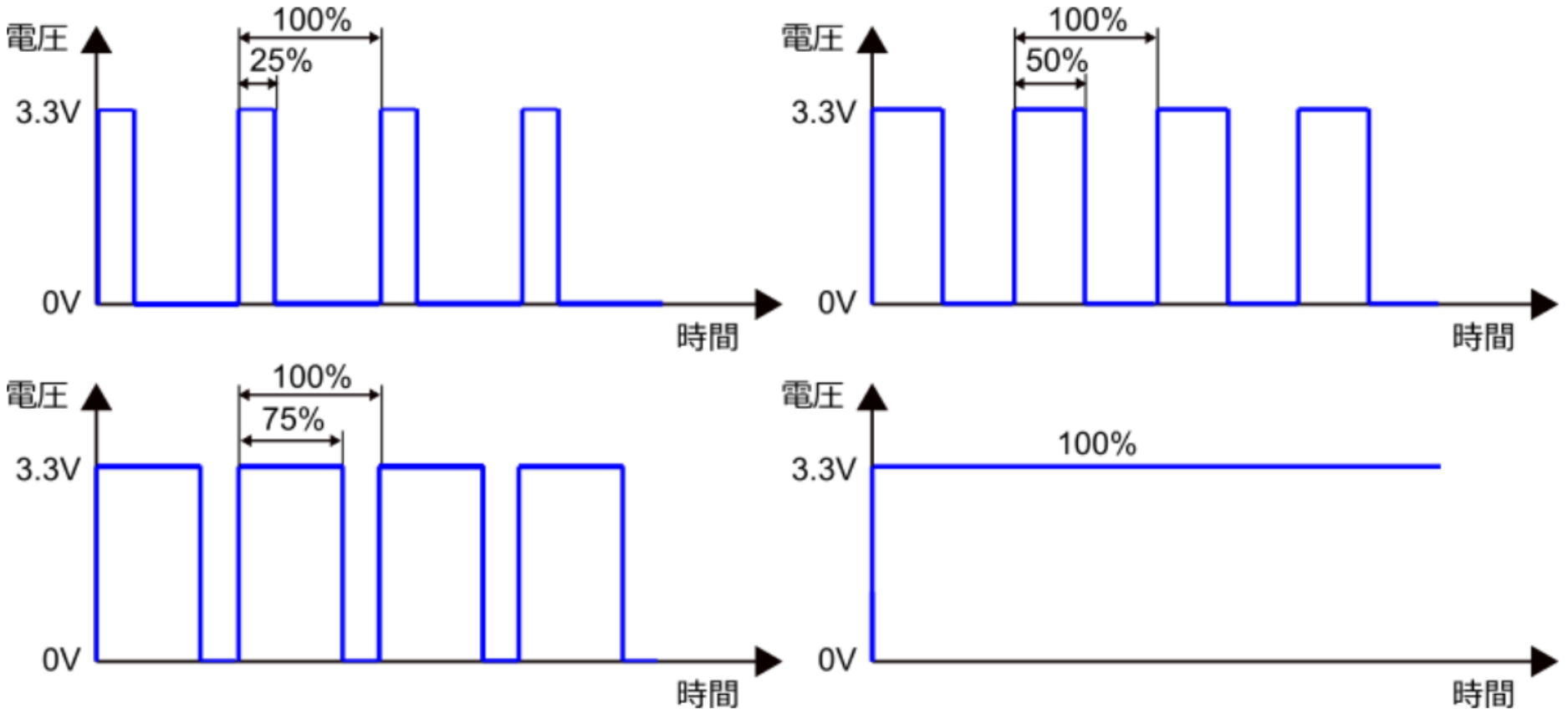
RCサーボモータ

RCサーボ
外部からの信号に応じて、一定の角度まで回転する小型モータ
用途
ロボットの関節、ラジコン飛行機の翼の角度制御など

PWM制御

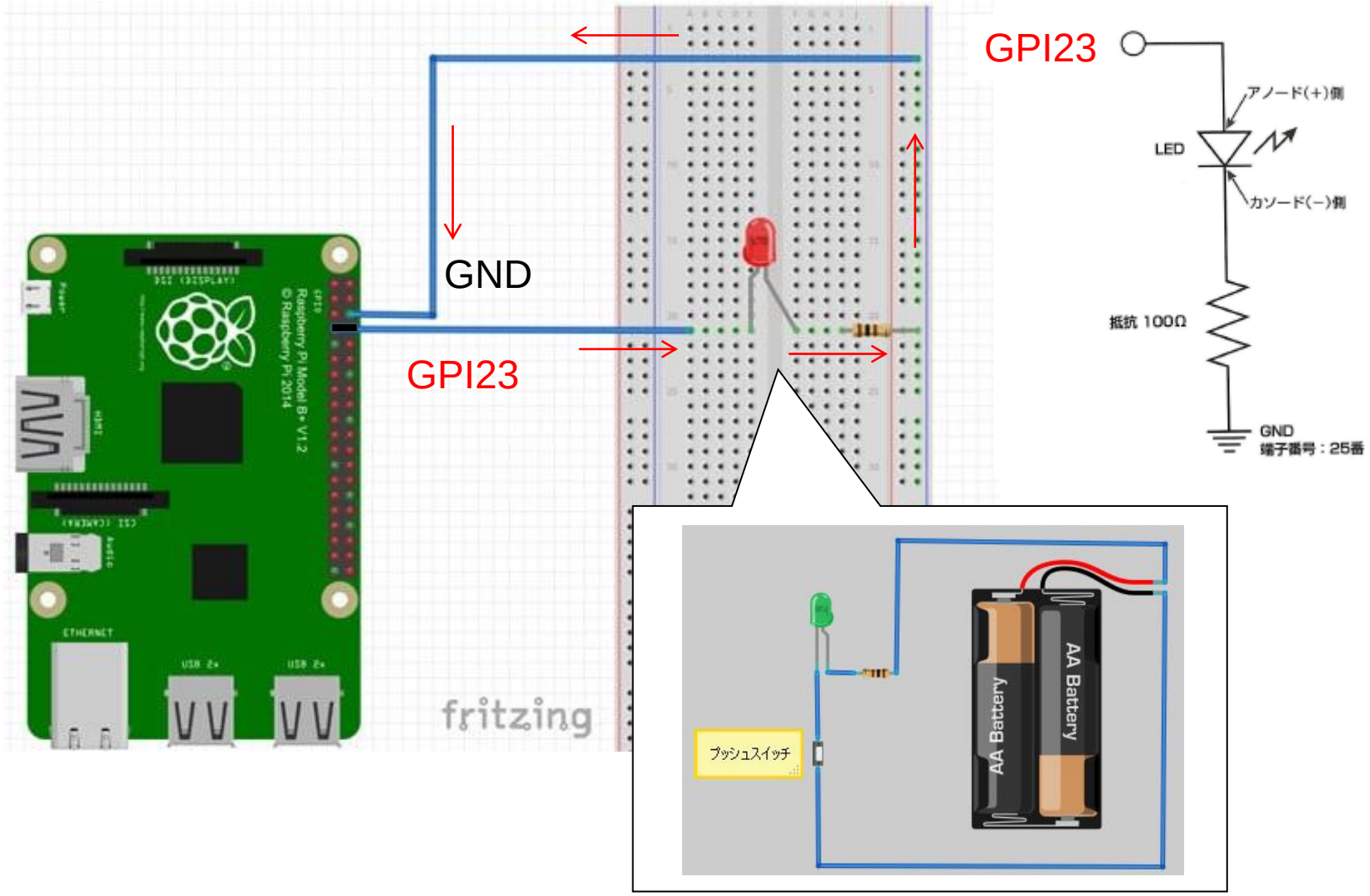


PWM制御


電圧
3.3V
0V
100%
25%
時間
電圧
3.3V
0V
100%
50%
時間
電圧
3.3V
0V
100%
75%
時間
電圧
3.3V
0V
100%
時間

PWM制御によるLED

pwm_led.py



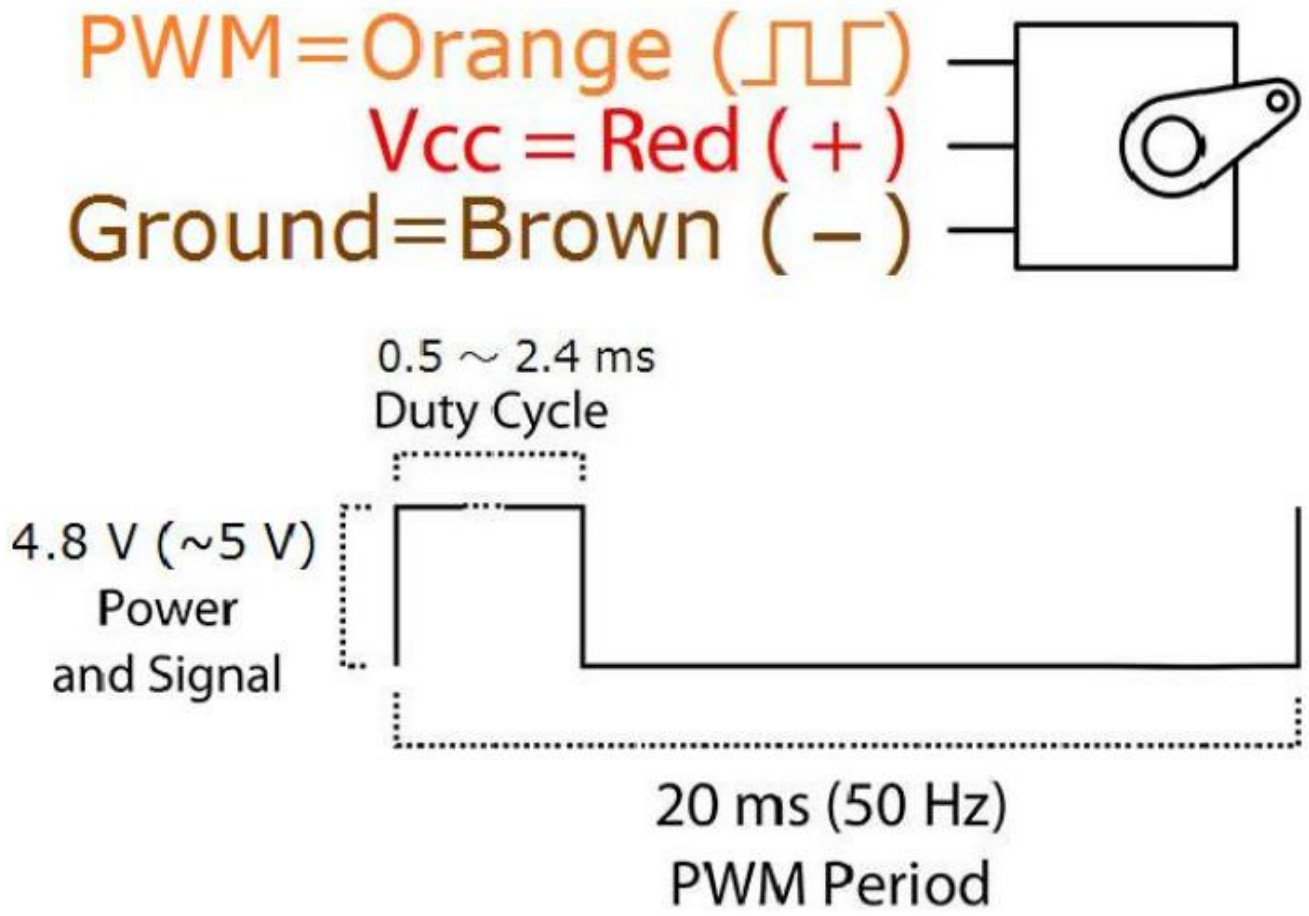
pwm_led.py

```
import RPi.GPIO as GPIO
import time
import sys


Led_pin = 23
GPIO.setmode(GPIO.BCM)
GPIO.setup(Led_pin, GPIO.OUT)
Led = GPIO.PWM(Led_pin, 50)
Led.start(0)
bright = 0
flag=0
while True:
    try:
        Led.ChangeDutyCycle(bright)
        time.sleep(0.01)
        if flag == 0:
            bright +=1
        else: bright -= 1
                if bright >= 100:
            bright = 100
            flag=1
        elif bright <=0:
            bright = 0
            flag=0
    except KeyboardInterrupt:
        Led.stop()
        GPIO.cleanup()
        sys.exit()
```

RCサーボモータの説明書

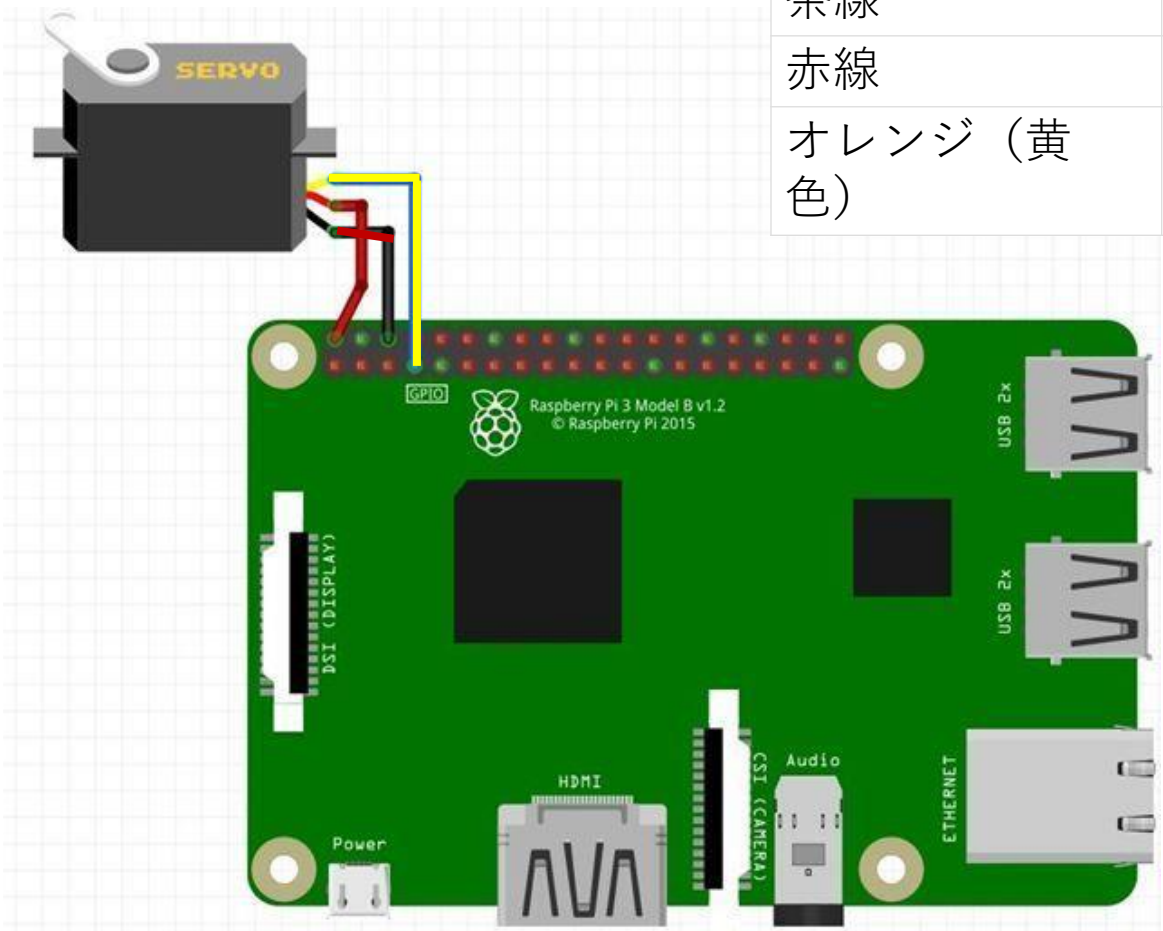
黄色→GPIO4
赤色→5V
茶色→GND



RCサーボモータ

rc.py
rc2.py

コード色	役割
茶線	GND（0V）
赤線	電源+（5V）
オレンジ（黄色）	信号線



黄色→GPIO4
赤色→5V
茶色→GND

rc.py

```
import RPi.GPIO as GPIO
import time
import sys


Led_pin = 23
GPIO.setmode(GPIO.BCM)
GPIO.setup(Led_pin, GPIO.OUT)
Led = GPIO.PWM(Led_pin, 50)
Led.start(0)
bright = 0
flag=0
while True:
    try:
        Led.ChangeDutyCycle(bright)
        time.sleep(0.01)
        if flag == 0:
            bright +=1
        else: bright -= 1
              if bright >= 100:
            bright = 100
            flag=1
        elif bright <=0:
            bright = 0
            flag=0
    except KeyboardInterrupt:
        Led.stop()
        GPIO.cleanup()
        sys.exit()
```

rc2.py

```
import RPi.GPIO as GPIO
import time
import sys
Servo_pin = 4
GPIO.setmode(GPIO.BCM)
GPIO.setup(Servo_pin, GPIO.OUT)

Servo = GPIO.PWM(Servo_pin, 50)
Servo.start(0)
def servo_angle(angle):
    duty = 2.5 + (12.0 - 2.5) * (angle +
90) / 180
    Servo.ChangeDutyCycle(duty)
    time.sleep(0.1)

while True:
    try:
        servo_angle(0)
        servo_angle(90)
        servo_angle(0)
        servo_angle(-90)
        servo_angle(0)
    except KeyboardInterrupt:
        Servo.stop()
        GPIO.cleanup()
        sys.exit()
```

rc.py ポイント

```
while True:
    try:
        Servo.ChangeDutyCycle(2.5)   #デューティ比[2.5%] = -90°
        time.sleep(0.5)              #0.5秒間待つ
        Servo.ChangeDutyCycle(7.25)  #デューティ比[7.25%] = 0°
        time.sleep(0.5)              #0.5秒間待つ
        Servo.ChangeDutyCycle(12)    #デューティ比[12%] = +90°
        time.sleep(0.5)              #0.5秒間待つ
    except KeyboardInterrupt:    #Ctrl+Cキーが押された
        Servo.stop()             #サーボモータをストップ
        GPIO.cleanup()           #GPIOをクリーンアップ
        sys.exit()               #プログラムを終了
```

ソース　rc2.py

```
#必要なモジュールをインポート
import RPi.GPIO as GPIO        #GPIO用のモジュールをインポート
import time                    #時間制御用のモジュールをインポート
import sys                     #sysモジュールをインポート

#ポート番号の定義
Servo_pin = X                  #変数"Servo_pin"にXを格納

#GPIOの設定
GPIO.setmode(GPIO.BCM)         #GPIOのモードを"GPIO.BCM"に設定
GPIO.setup(Servo_pin, GPIO.OUT)    #GPIO Xを出力モードに設定

#PWMの設定
#サーボモータSG90の周波数は50[Hz]
#GPIO.PWM(ポート番号, 周波数[Hz])
Servo = GPIO.PWM(Servo_pin, 50)
```

GPIOをポート番号で扱う方法に設定

ソース　rc2.py

```
Servo.start(0)              #Servo.start(デューティ比[0-100%])

#角度からデューティ比を求める関数
def servo_angle(angle):
    duty = 2.5 + (12.0 - 2.5) * (angle + 90) / 180
                                        #角度からデューティ比を求める
    Servo.ChangeDutyCycle(duty)     #デューティ比を変更
    time.sleep(0.3)                     #0.3秒間待つ
```

[2.5%] = -90°
[7.25%] = 0°
[12%] = +90°

ソース　rc2.py

```
#while文で無限ループ
#サーボモータの角度をデューティ比で制御
#Servo.ChangeDutyCycle(デューティ比[0-100%])
while True:
    try:
        servo_angle(-90)              #サーボモータ -90°
        servo_angle(-60)              #サーボモータ -60°
        servo_angle(-30)              #サーボモータ -30°
        servo_angle(0)               #サーボモータ  0°
        servo_angle(30)               #サーボモータ  30°
        servo_angle(60)               #サーボモータ  60°
        servo_angle(90)               #サーボモータ  90°
    except KeyboardInterrupt:        #Ctrl+Cキーが押された
        Servo.stop()                 #サーボモータをストップ
        GPIO.cleanup()               #GPIOをクリーンアップ
        sys.exit()                   #プログラムを終了
```