令和4年度
Raspberry Pi(ラズパイ)で行うIoT研修

(オンデマンド研修)

埼玉県産業技術総合センター（SAITEC）
電気・電子技術・戦略プロジェクト担当
鈴木 浩之

(Ver2_1)

研修の流れ

配布教材の確認

- ラズパイ本体
- 電源
- スイッチ付きケーブル
- マイクロSDカード
- ヒートシンク
- HDMIケーブル
- ブレッドボード
- フォトセンサー
- ジャンパー線（オスーオス）
- ジャンパー線（オスーメス）
- 赤LED
- 黄LED
- 抵抗
- リードスイッチ（マグネットセンサー）
- 電子ブザー

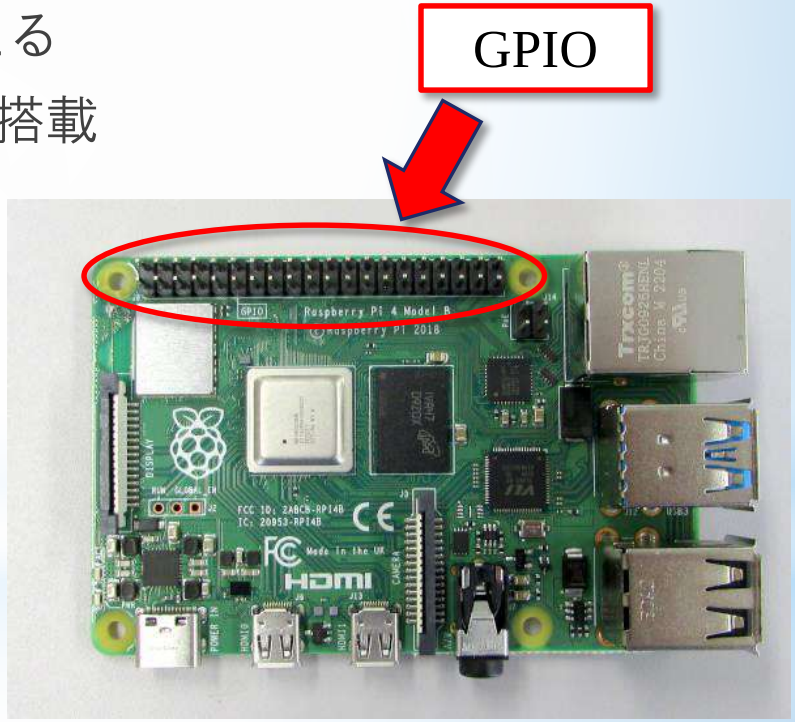
参考書籍
これ1冊でできる！ラズベリー・パイ超入門 改訂第7版
福田和宏 著 2022年3月31日 初版 第1刷発行
以降は単に「参考書籍」と表記します。

1　Raspberry Piの基礎

1　Raspberry Piの基礎

Raspberry Pi（ラズベリーパイ）とは

- 小さなコンピュータ（シングルボードコンピュータ）
- イギリスのラズベリーパイ財団が開発
  - ➢ネット通販や秋葉原の電子部品店などで購入可能
- 小さくても普通のパソコンのようにも使える
- インターフェイス（USB端子など）も多数搭載
- 「GPIO」という入出力端子（ピン）が付いていて、電気的な入力・出力が可能
  - ➢LEDやモーターの制御：出力
  - ➢センサーのデータやスイッチのON-OFFの取り込み：入力
- いろいろなモデルが出ていますが、本研修で使うのは「Raspberry Pi 4　Model B」というモデルで、搭載メモリーが4GBのものです。



✓ モデルの違い等の詳しい情報は、参考書籍 p14～

1　Raspberry Piの基礎

ラズベリーパイ（Raspberry Pi）とは

- 小さいですが、USB接続、有線LAN、無線LANなどが使えます。もちろん画面の出力もできます。
- パソコンのようなハードディスクは持っていません。「マイクロSDカード」をハードディスク代わりに使います。
- マウス、キーボード、ディスプレイをつなげば、小さなパソコンとして動きます。

◆ これ以降**「ラズパイ」**と表記する場合があります。

本研修の目標

・ラズパイによるセンサーデータの取得と表示を体験

・Pythonの簡単な学習をして、データ取得の工夫の仕方を体験

・ラズパイからの出力を使ってブザーやLEDでお知らせ

・Node-RED (ノードレッド)やPythonでデータをグラフ表示

ラズパイを使うための準備と起動について

ラズパイを使うための準備と起動について説明します。

- ヒートシンクを取り付ける。
- マイクロSDカードを挿入する。
- ディスプレイ、マウス、キーボードを接続する。
- 電源を接続する。

発熱対策の詳細は
参考書籍p35

◆ 電源に関して

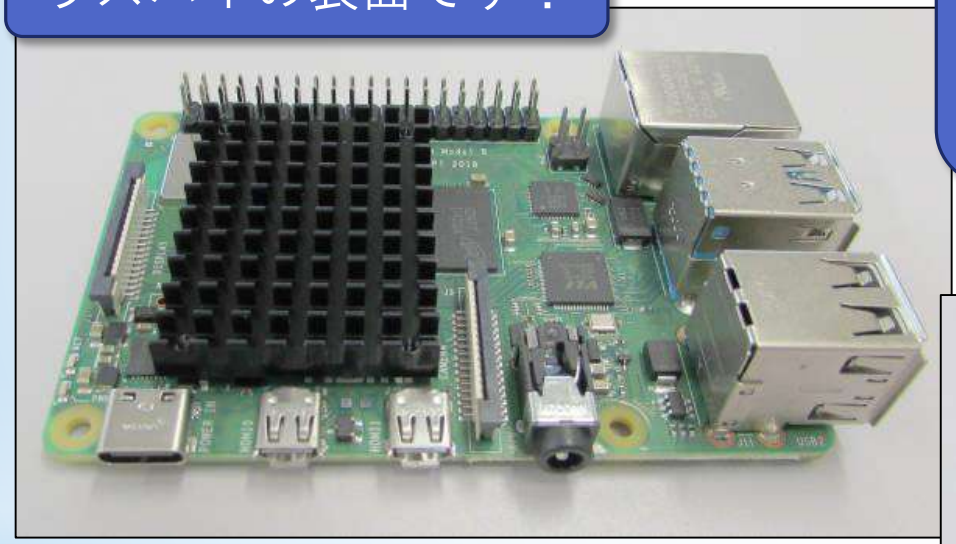
- ラズパイ自体に電源スイッチはありません。電源をつなぐとラズパイがONになります。
- シャットダウンは、ラズパイを操作して行います。
  - ➢ 再度ONにするには？ ⇒ 電源のケーブルを抜いて差しなおす。
- 本研修では「スイッチ付きケーブル」を電源とラズパイの間に入れて使います。スイッチのON-OFFを使えば、ケーブルの抜き差しをせずに、ラズパイの電源をONにすることができます。

1　Raspberry Piの基礎

ラズパイを使うための準備

ヒートシンクを取り付けましょう

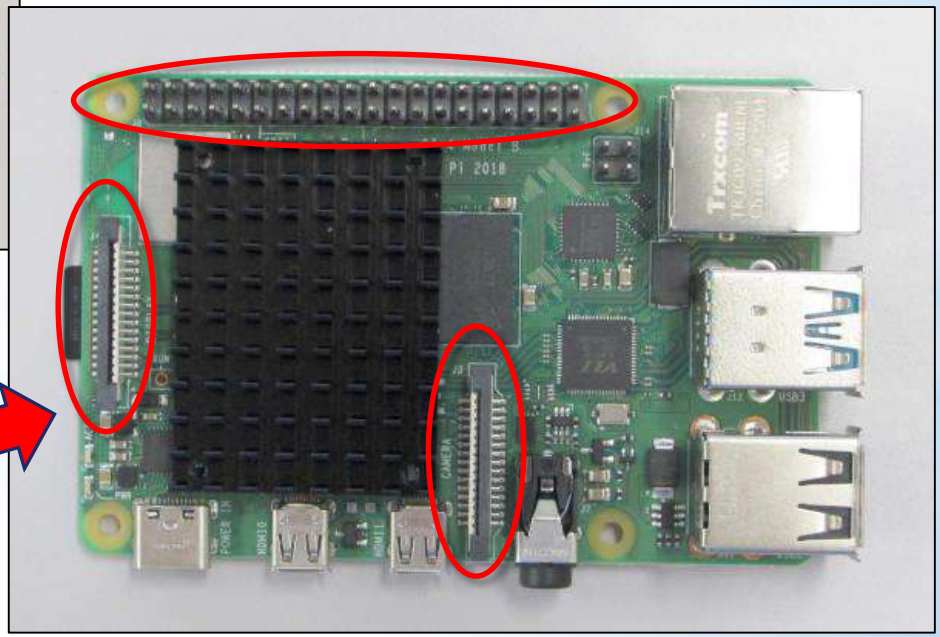
ラズパイの表面です！



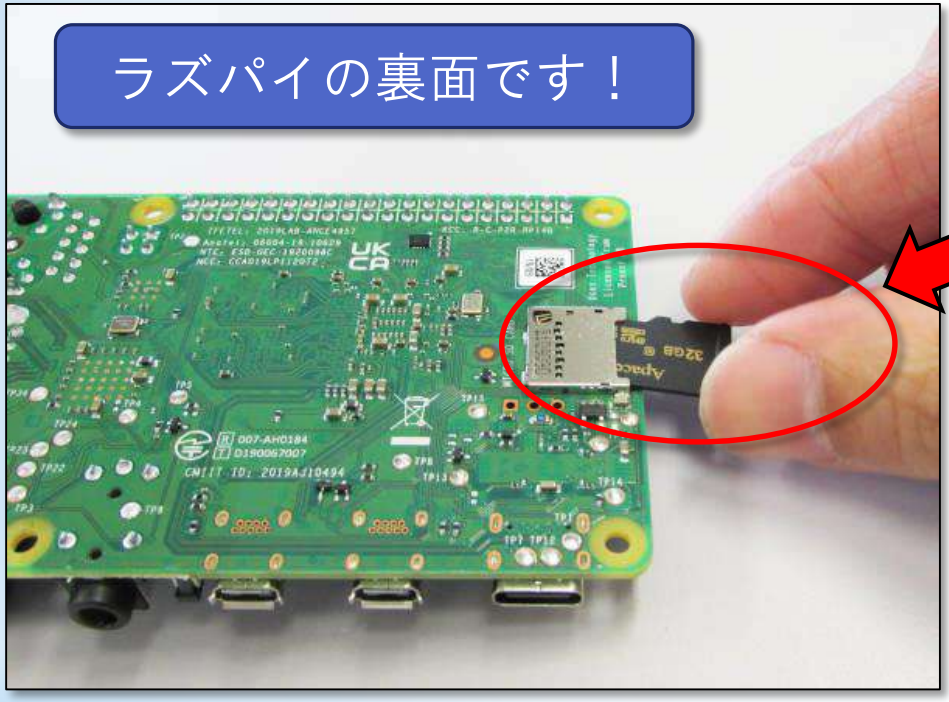
ラズパイ４はCPU性能が良い分、
発熱も増えています。
ファン付きのヒートシンクもありますが、
本研修ではファンのない
ヒートシンクを使います。

付属の両面テープで、CPUに貼り付ける。

ヒートシンクを取り付ける際、GPIOや他のコネクタをふさがないように注意しましょう。



ラズパイを使うための準備



本研修では、必要な設定を済ませたマイクロSDカードを同梱していますので、それを挿入しましょう。

✓ ラズパイでは、マイクロSDカードがハードディスクの代わりです。

ラズパイを使うための準備

続いて、ケーブル類を接続します

ラズパイの表面です！

［マウスとキーボード］
USB端子に接続する
(可能であれば、青でない2つの口に接続する。)

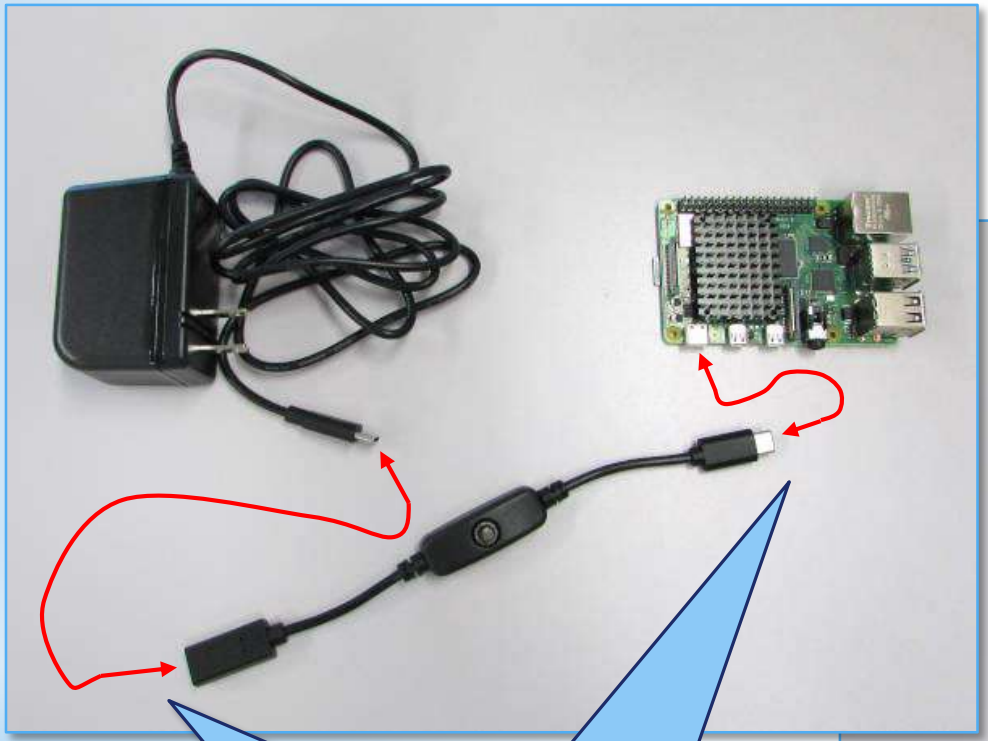
青いUSB端子は、より高速なUSB規格に対応しているので、後で高速な接続が必要になった場合にそなえて、空けておくとよい。
（本研修では使いませんので、無理ならどこでも可）

［ディスプレイ］
マイクロHDMI端子に接続する

マイクロHDMI端子は2つあります。
どちらに差しても構いません。

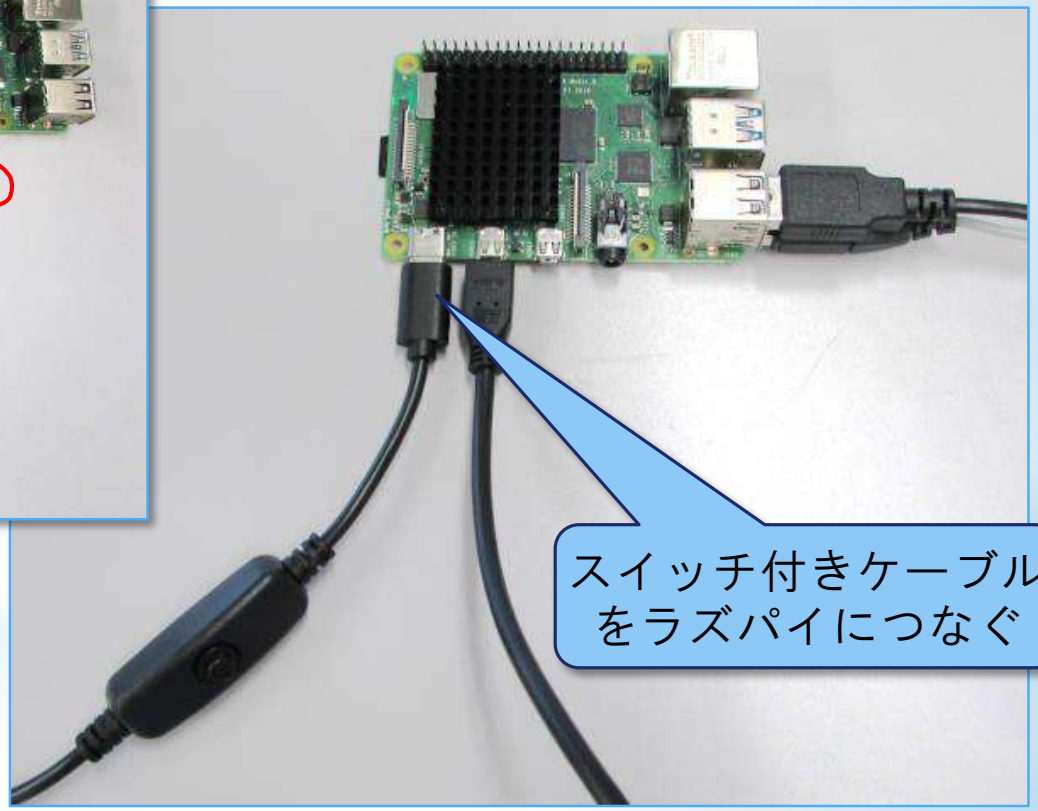
1　Raspberry Piの基礎

ラズパイを使うための準備



電源のケーブルとラズパイの間に、スイッチ付きケーブルを入れてつなぎます。

電源を接続します



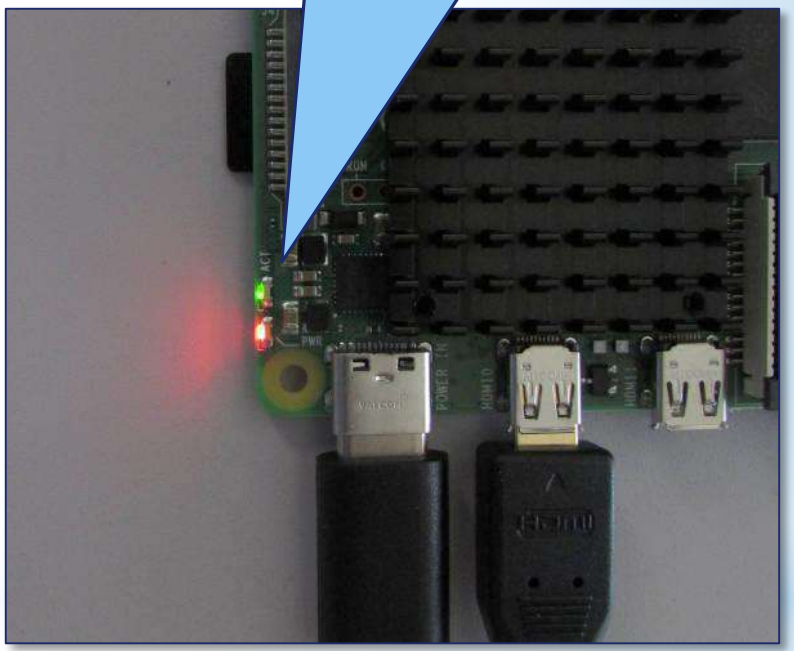
スイッチ付きケーブルをラズパイにつなぐ

1　Raspberry Piの基礎

ラズパイの起動

ケーブルを全部つないだら、モニタの電源を入れ、スイッチ（押しボタン）を押して、ラズパイをONにしましょう。

ONにすると、基板上のLEDが点灯・点滅します。



ケーブル類接続後の全体イメージ

コンセントに接続

赤LED：電源が接続されているときに点灯
緑LED：OSがマイクロSDカードにアクセスしているときに点滅

1　Raspberry Piの基礎

ラズパイの起動

ラズパイが起動すると、Windowsに似た画面が出てきますが、ラズパイ用のOS（『Raspberry Pi OS』）です。(Linuxをベースにしたos)

【注意！】

◆販売店で売っているマイクロSDカードを差しただけで「 Raspberry Pi OS 」が起動するわけではありません。

◆事前の準備がいろいろ必要です。

◆本研修では、事前に準備、設定をしてあります。

◆ラズパイ用のマイクロSDカードを自分で用意してみたい、という方は、参考書籍のp40～を参照してください。

2　小型パソコンとしてのRaspberry Pi

2　小型パソコンとしてのRaspberry Pi

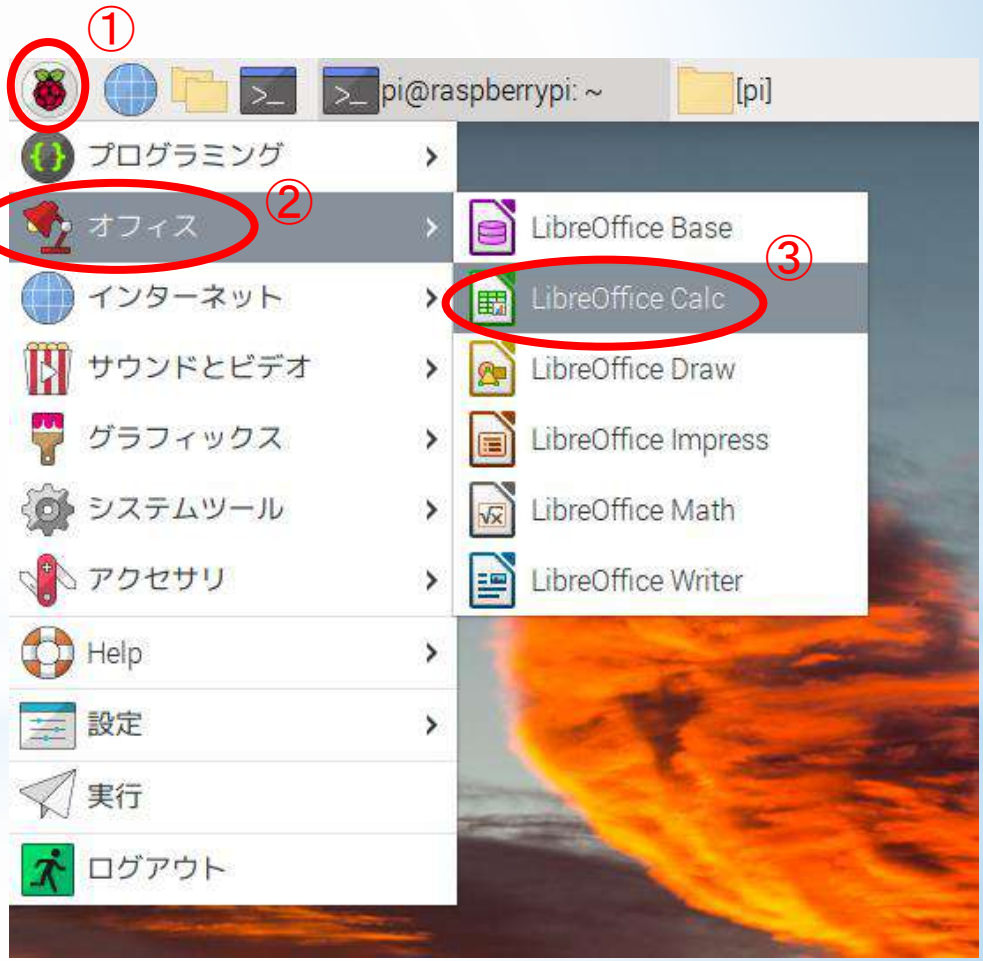
オフィスの起動

- ラズパイは簡単なパソコンとしても使えます。
- 試しに、オフィスを使ってみましょう。例えば、エクセルに似たソフトが使えます。

① 画面の左上の「ラズベリーのアイコン」をクリック

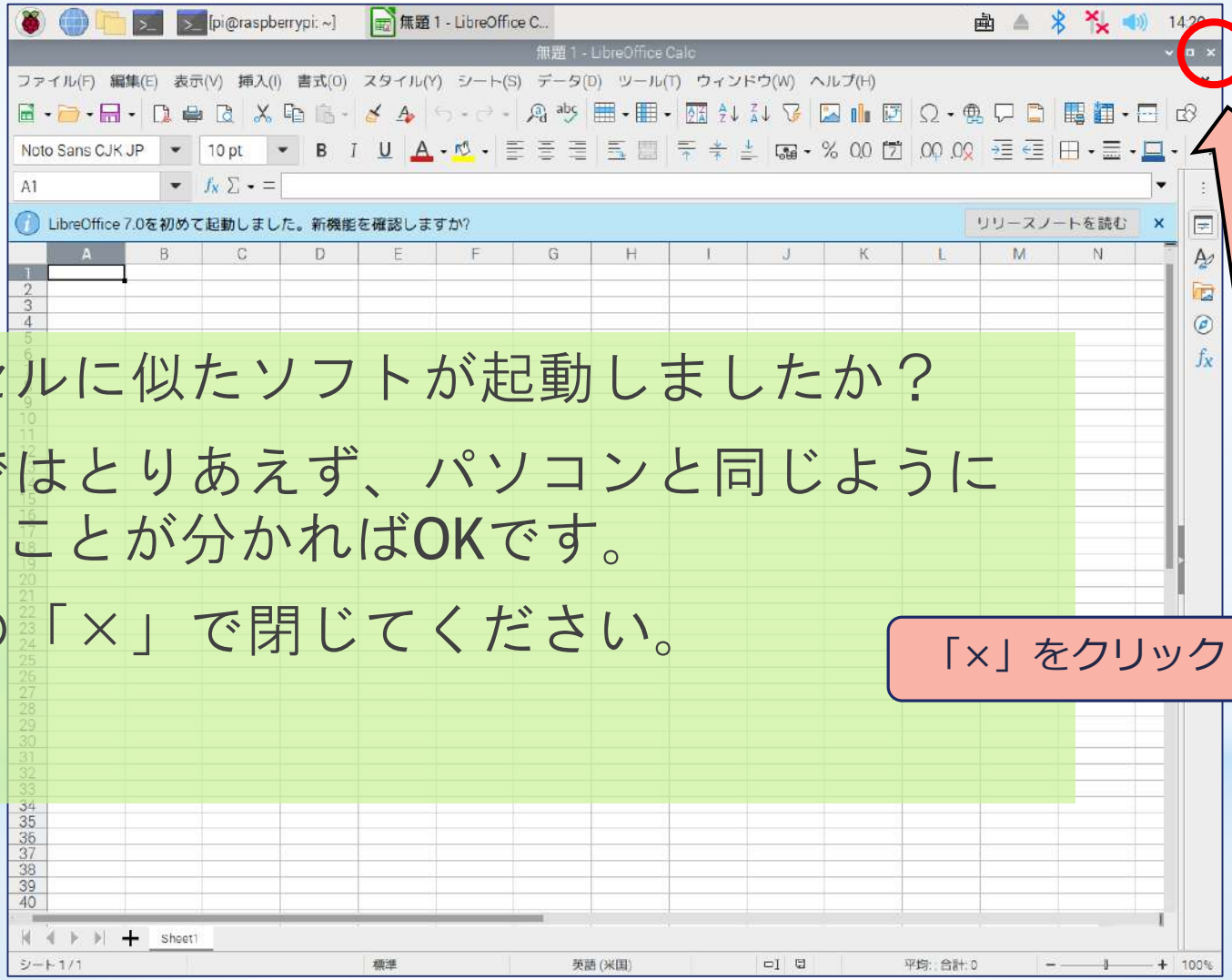
② 「オフィス」にマウスを移動

③ 緑色のアイコン（LibreOffice Calc）をクリック



2　小型パソコンとしてのRaspberry Pi

オフィスの起動

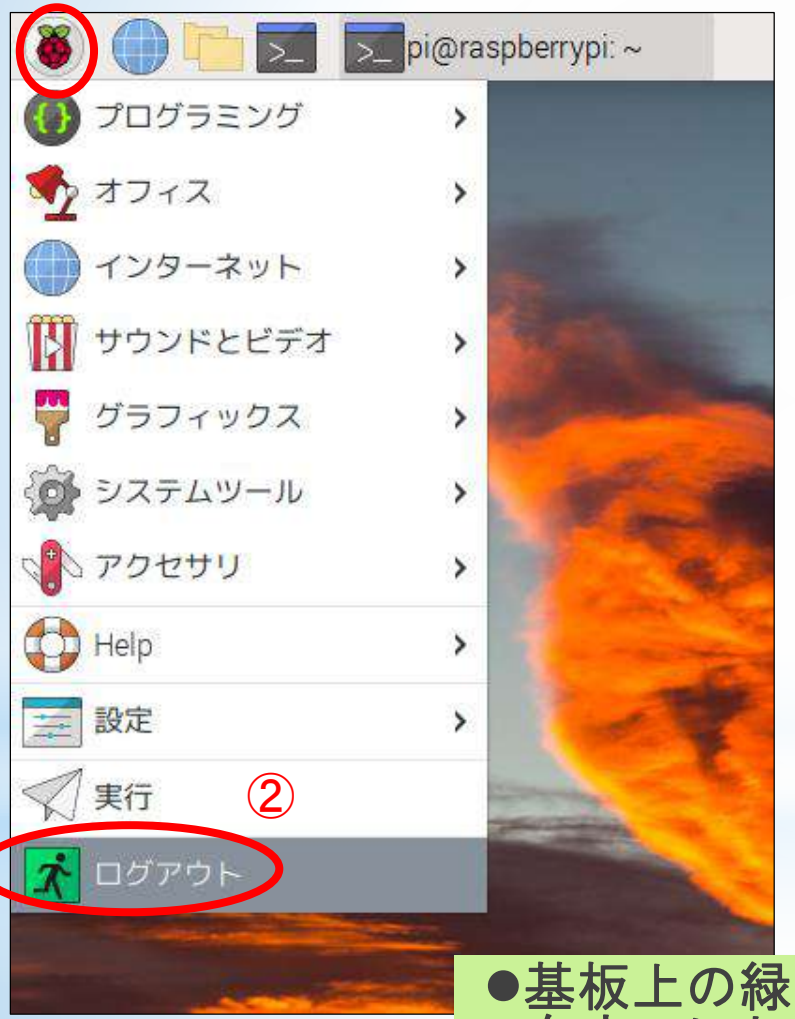


- エクセルに似たソフトが起動しましたか？
- ここではとりあえず、パソコンと同じように使えることが分かればOKです。
- 右上の「×」で閉じてください。

「×」をクリックして閉じる

2　小型パソコンとしてのRaspberry Pi

シャットダウン方法の確認

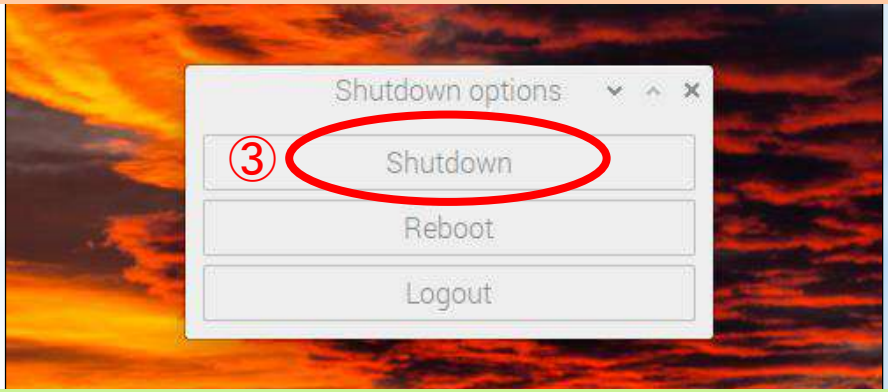


●シャットダウンの方法を確認しましょう

① 画面の左上の「ラズベリーのアイコン」をクリック

② 「ログアウト」をクリック

③ 別のウィンドウが開くので、そこの「Shutdown」をクリック



●基板上の緑LEDの点滅が完全に停止したら、スイッチをオフにしてください。⇒赤LEDが消灯します。

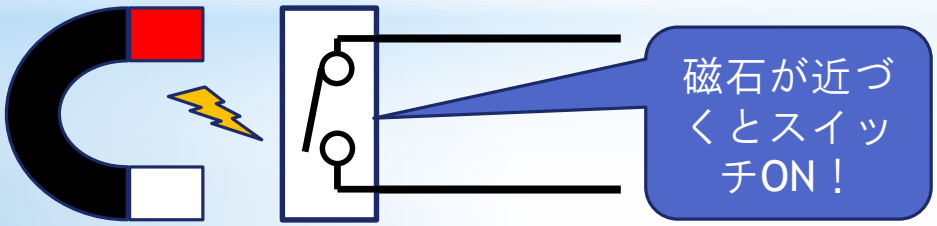
3　マグネットセンサー (リードスイッチ)

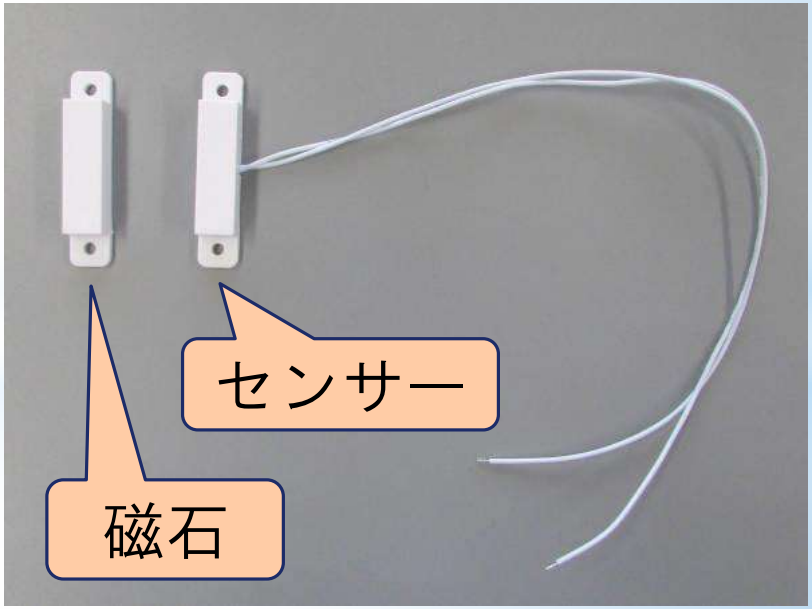
（注）これ以降、各章で実施するブレッドボードとラズパイの電子工作は、追加で実施してください。（前の章で作成した配線は、外さずにそのままにしておいてください。）

3　マグネットセンサー

マグネットセンサー（リードスイッチ）とは

- マグネットセンサーを使った、ON-OFF情報の取得を体験します。
- マグネットセンサーは、磁石が近づくと「ON」になり磁石が離れると「OFF」になるスイッチと考えることができます。
  - 電子部品としては「リードスイッチ」と呼ぶことが多いですが、この資料ではイメージしやすいように「マグネットセンサー」と呼ぶことにします。





3　マグネットセンサー

ラズパイのGPIOについて

- ラズパイには「GPIO」という信号の入出力等に使える端子（ピン）が付いています。
- 40本のピンがあり、用途が決まっています。また、下の図のように端子には番号が付けられています。

- 主な用途は、電源、グランド（GND）、入出力などです。

➢ グランドは電気のマイナス（－）と考えてください。
✓参考：電源の表記について ⇒ 参考書籍p177～

✓GPIOについての詳細は、参考書籍p162～

3　マグネットセンサー

ラズパイのGPIOについて

- 入出力用のピンは、設定を切り替えることで、入力用のピンとして使ったり、出力用のピンとして使ったりします。
- マグネットセンサーのON-OFFの情報をラズパイが受け取るためには、GPIOピンを入力用として使います。
- ラズパイはデジタル信号しか扱えません。GPIOの入出力は、「1」か「0」のどちらかとして扱われます。基準となる電圧より高いと「1」、低いと「0」となります。

✓「0」を「LOW」「OFF」と、「1」を「HIGH」「ON」と表記することもある。詳しくは参考書籍 p183

- 一部の入出力ピンは、用途を切り替えて「I²C」や「SPI」などといった、信号をやりとりする規格に従った通信に使うこともできますが、本研修では取り上げません。
- アナログ値を入力するためには、A/Dコンバーターなどを使う必要があります。本研修では取り上げませんが、詳しく知りたい方は参考書籍 p216～をご覧ください。

3　マグネットセンサー

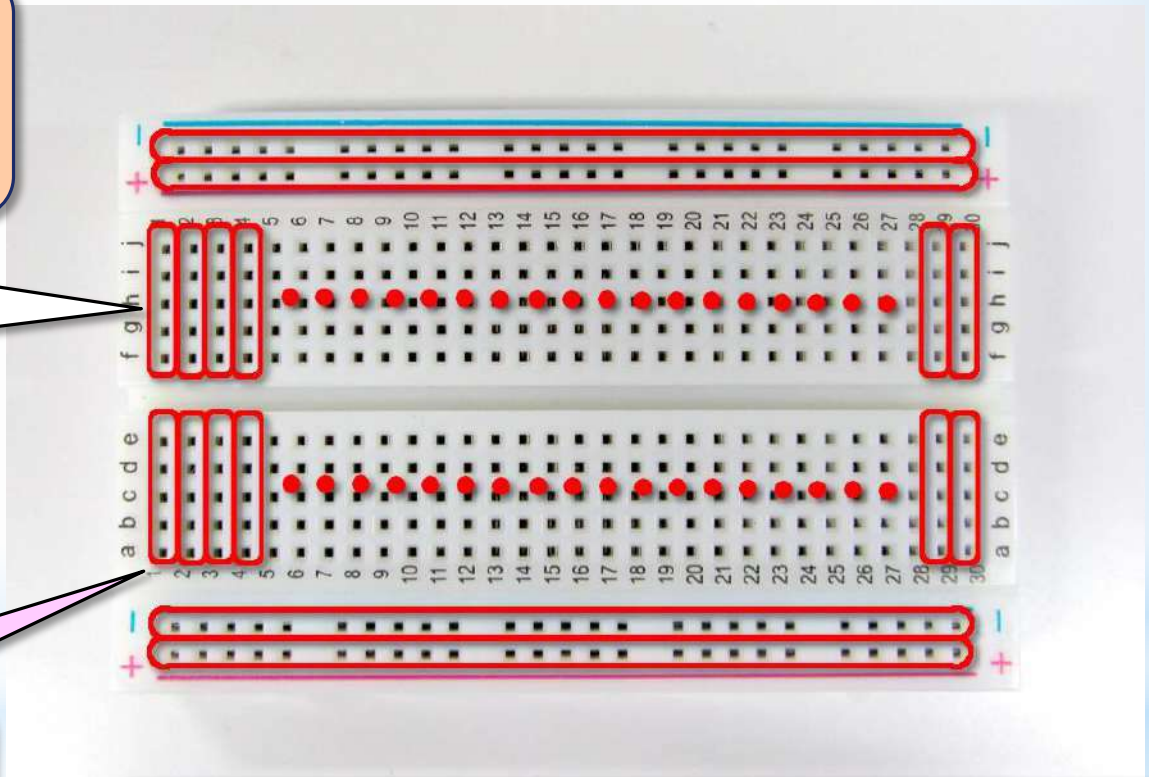
ブレッドボードについて

- 電子回路を作成するために「ブレッドボード」を使います。

穴に部品の足などを差し込むことで、電気的に接続することができる。

赤い枠で囲った範囲が、電気的に接続されている

✓参考書籍 p169も参考にしてください



本研修では、穴の位置を(a,1)のようにアルファベットと数字で表現することがあります。

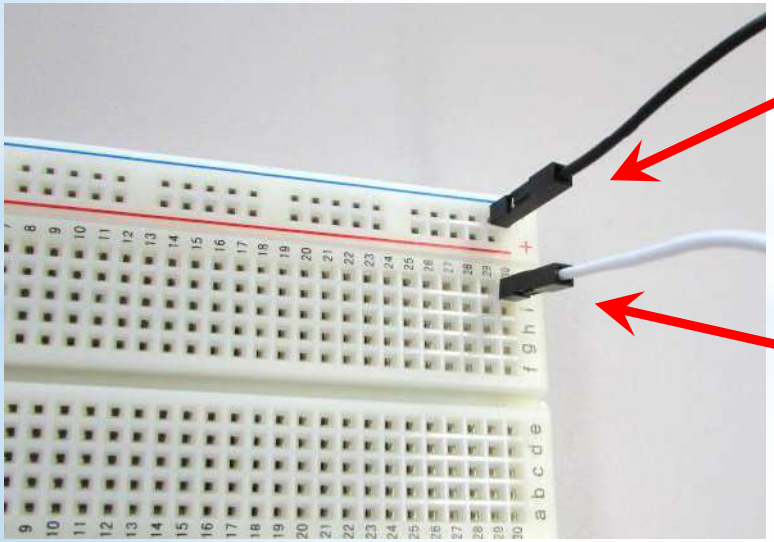
「＋」「－」が書かれた列は、電源やGND用に使うと便利。（電源やGNDは多く使うため）

3　マグネットセンサー

マグネットセンサーの体験

✓ （まだシャットダウンしてない方は）ラズパイを一度シャットダウンしてください。

- オス-メス のジャンパー線を使います。
- 配布したジャンパー線はつながって束になっていますが、外して使ってください。



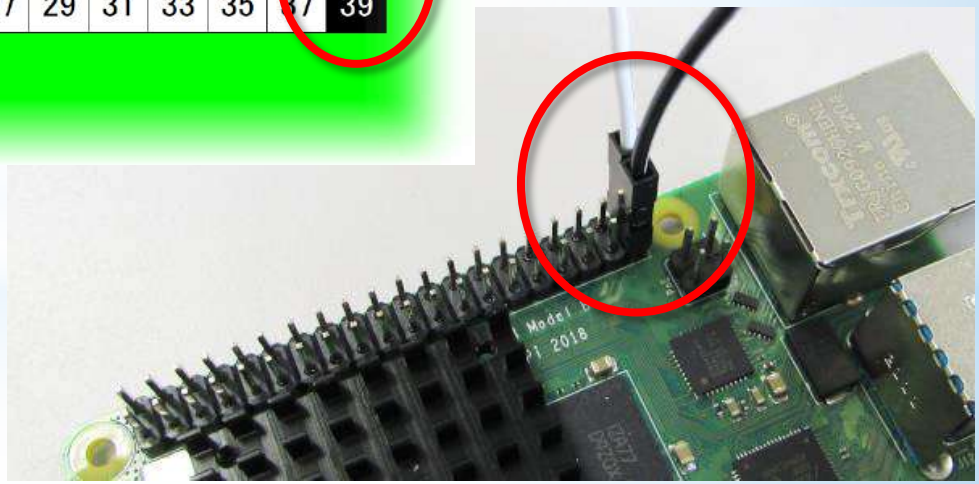
- まずは黒のジャンパー線を一つ取って、ブレッドボードの「－」の列に差します。
- 同様に、白のジャンパー線をブレッドボードの(j,30)の穴に差します。

3　マグネットセンサー

マグネットセンサーの体験

- 黒のジャンパー線の反対側を、ラズパイのGPIOの39番ピンにつなぎます。39番端子はGNDです。
- 同様に、白のジャンパー線の反対側を、 GPIOの40番ピンにつなぎます。

2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39



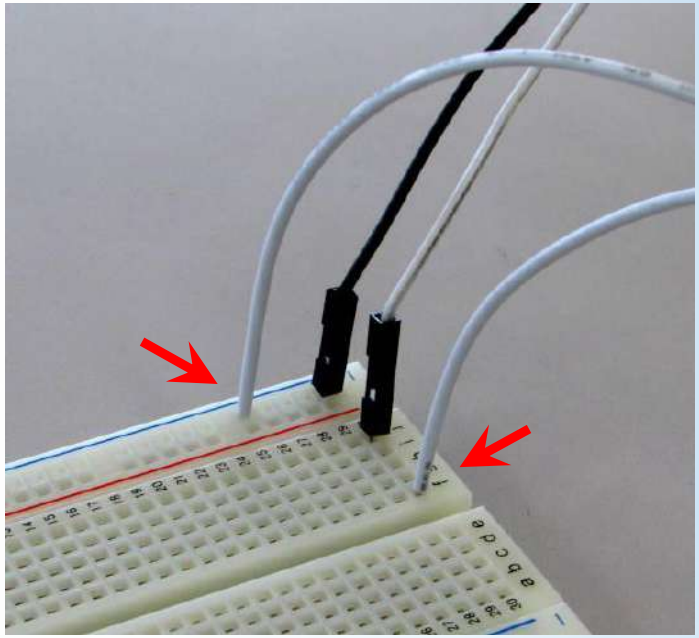
3　マグネットセンサー

マグネットセンサーの体験

- マグネットセンサーの2本の線のうち１本を、先ほど差したブレッドボードの「－」の列に差します。

マグネットセンサーの線に、プラスとマイナスの区別はありません。

- もう片方を、ブレッドボードの白のジャンパー線を差した列に差します。



マグネットセンサーがONになる、つまりスイッチがつながると・・

⇓

40番ピンがGNDにつながる。⇒ 40番ピンに「0」が入力される。

3　マグネットセンサー

マグネットセンサーの体験

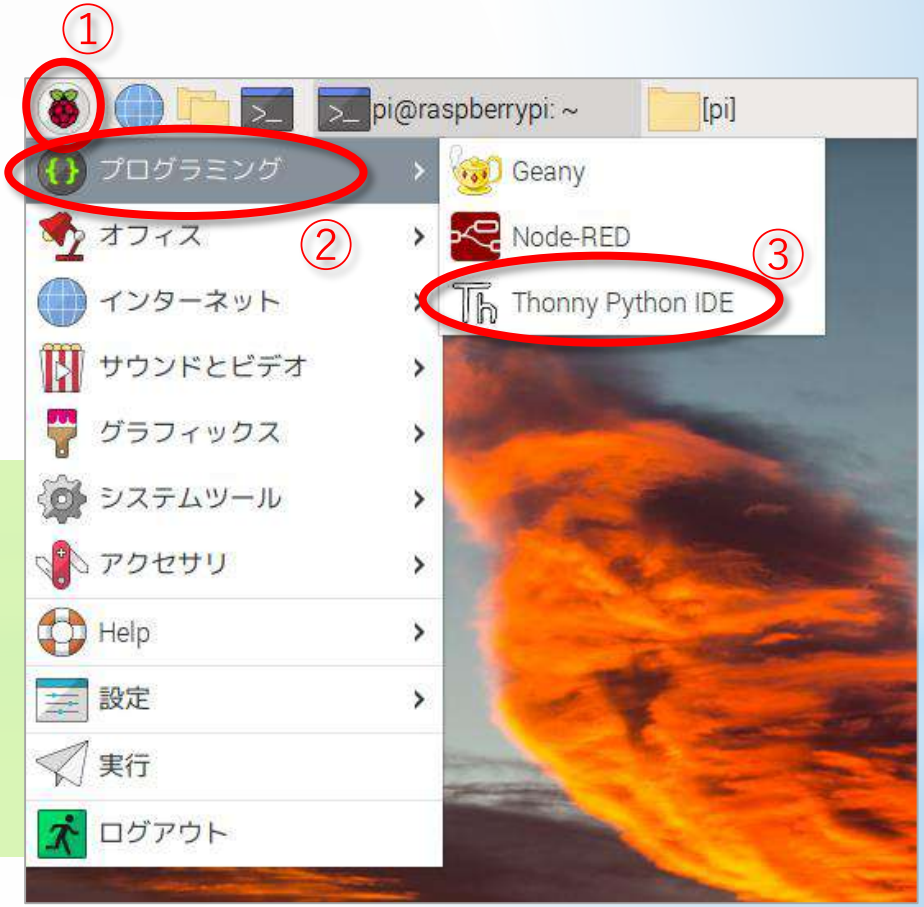
センサーを接続したら、ラズパイの電源を入れましょう。

- センサーの状態を読み取るためにはプログラムが必要です。
- “プログラミング言語”には様々なものがありますが、ラズパイでは、『Python』(パイソン)というプログラミング言語がよく使われます。
- 事前にPythonで作ったプログラムを用意してありますので、まずは動作を確認してみましょう。

3　マグネットセンサー

マグネットセンサーの体験

- プログラムの実行や編集などをするために、『Thonny』(トニー)というソフトを使います。
- 『 Thonny 』とは？
  - プログラムを作ったり、修正したり、実行したりするためのソフトです。
  - Thonny自体はプログラミング言語ではありません。



Thonnyの起動

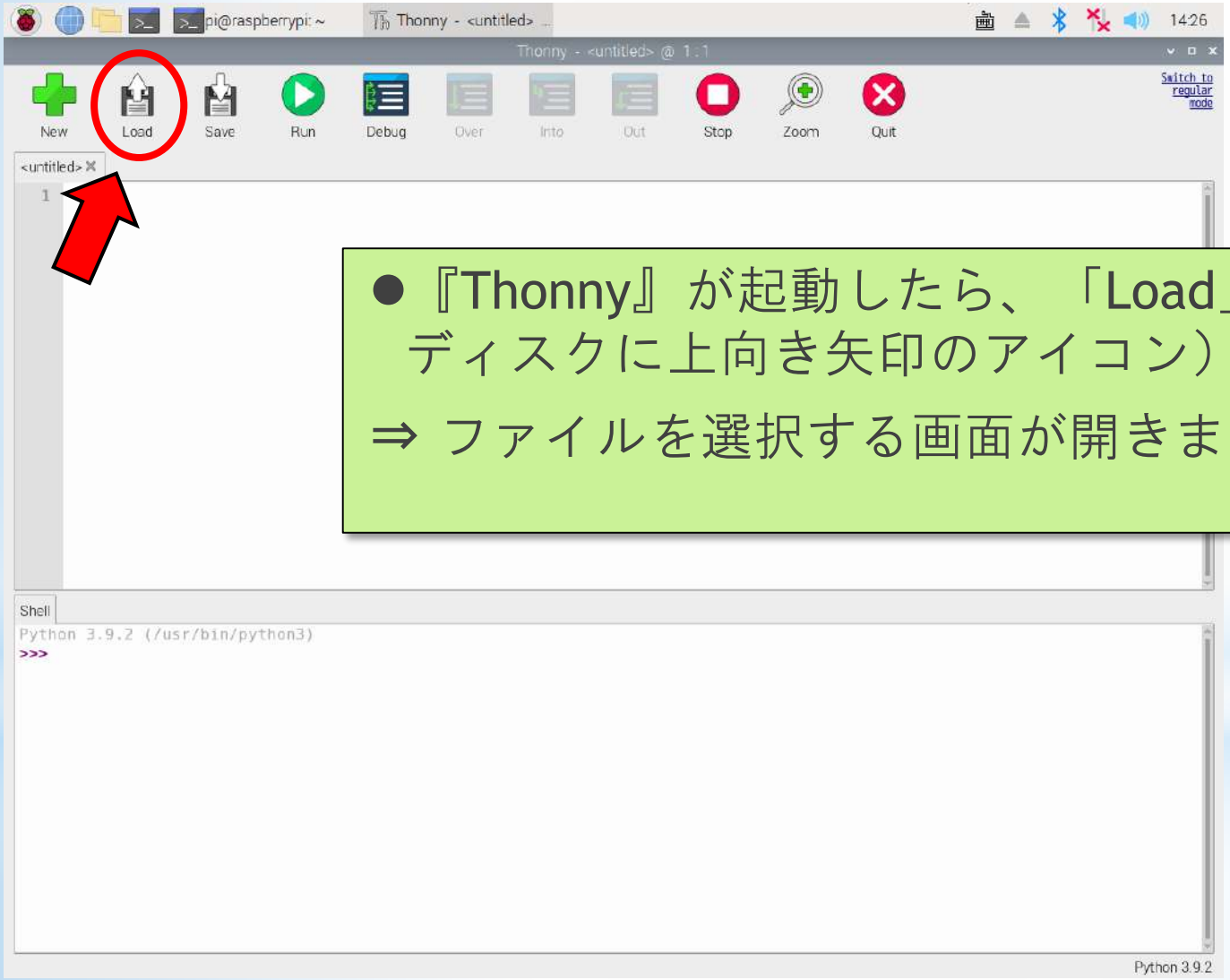
① 画面の左上の「ラズベリーのアイコン」をクリック

② 「プログラミング」へマウスを移動

③ 「Thonny Python IDE」をクリック

3　マグネットセンサー

マグネットセンサーの体験

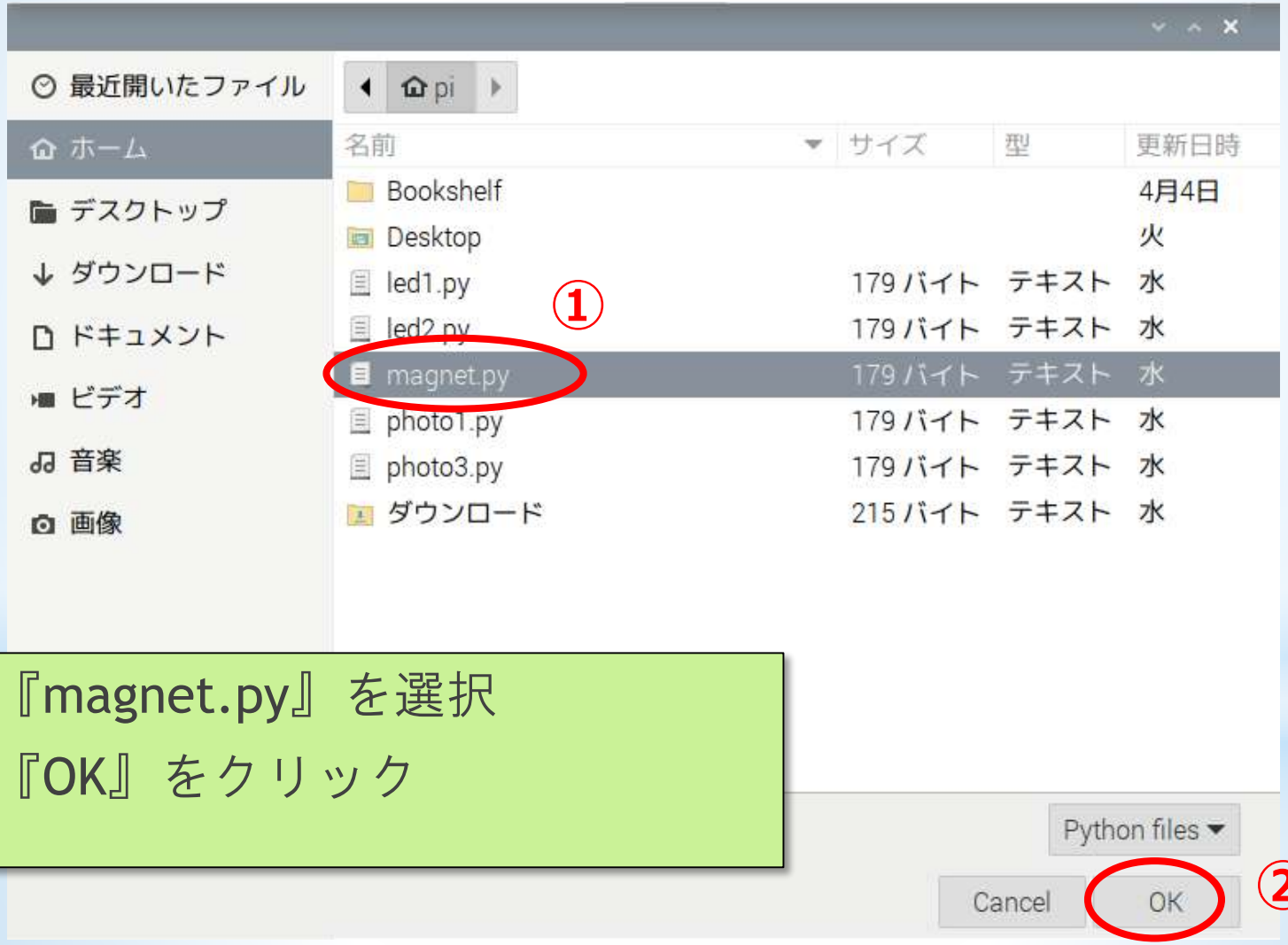


- 『Thonny』が起動したら、「Load」（フロッピーディスクに上向き矢印のアイコン）をクリック

⇒ ファイルを選択する画面が開きます。

3　マグネットセンサー

マグネットセンサーの体験

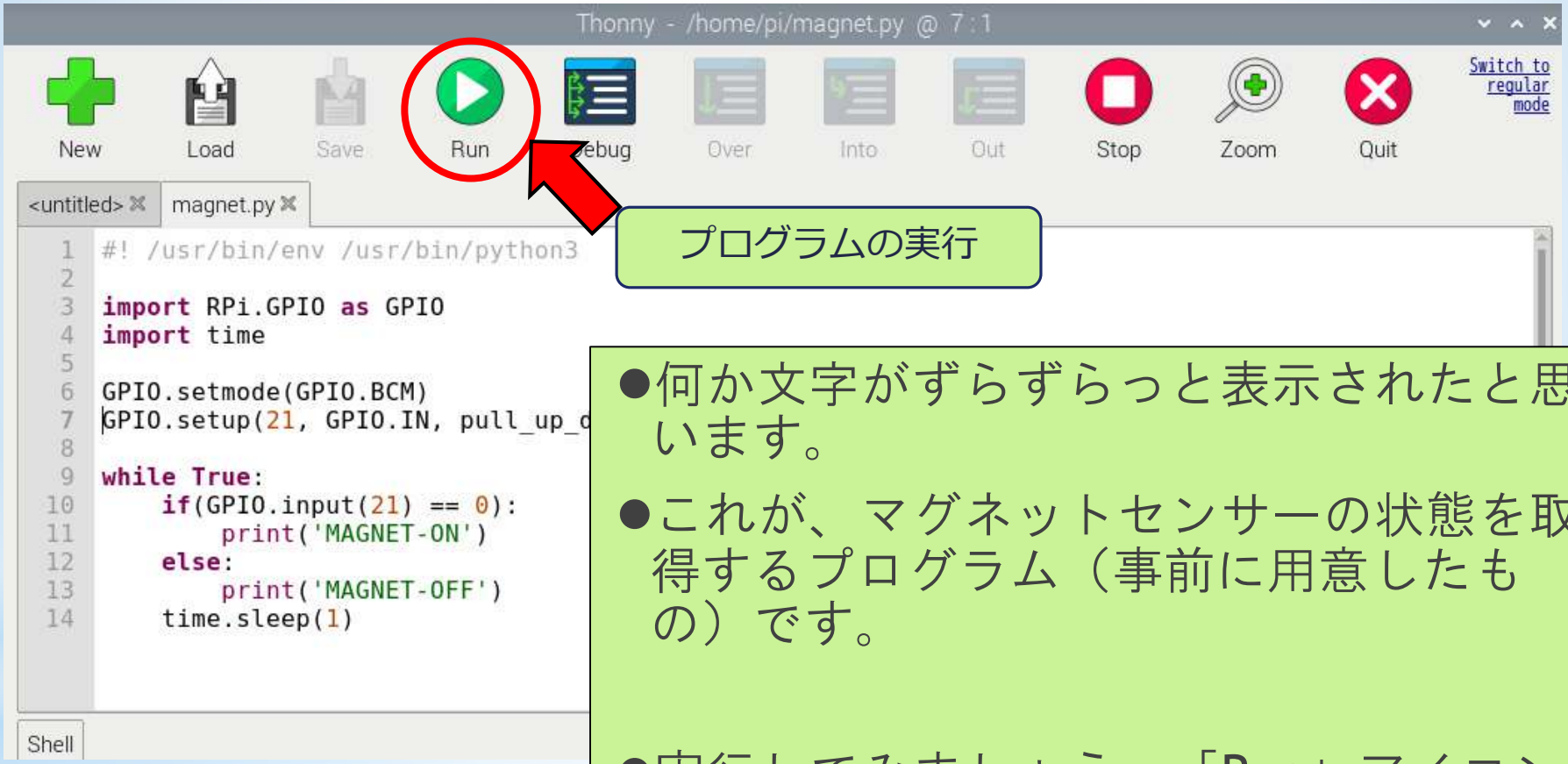


① 『magnet.py』を選択

② 『OK』をクリック

3　マグネットセンサー

マグネットセンサーの体験

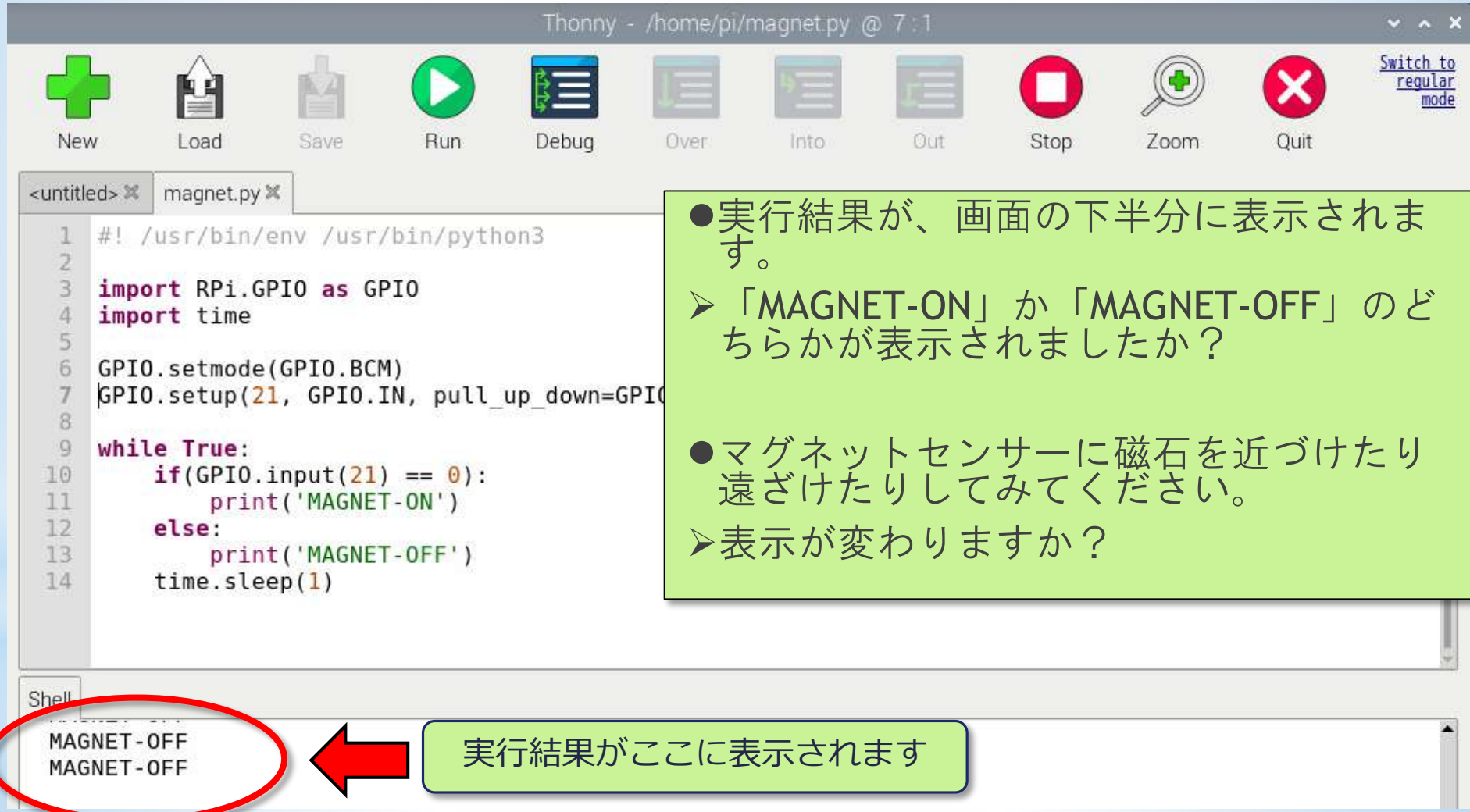


プログラムの実行

- 何か文字がずらずらっと表示されたと思います。
- これが、マグネットセンサーの状態を取得するプログラム（事前に用意したもの）です。
- 実行してみましょう。「Run」アイコンをクリックすると、プログラムが実行されます。

3　マグネットセンサー

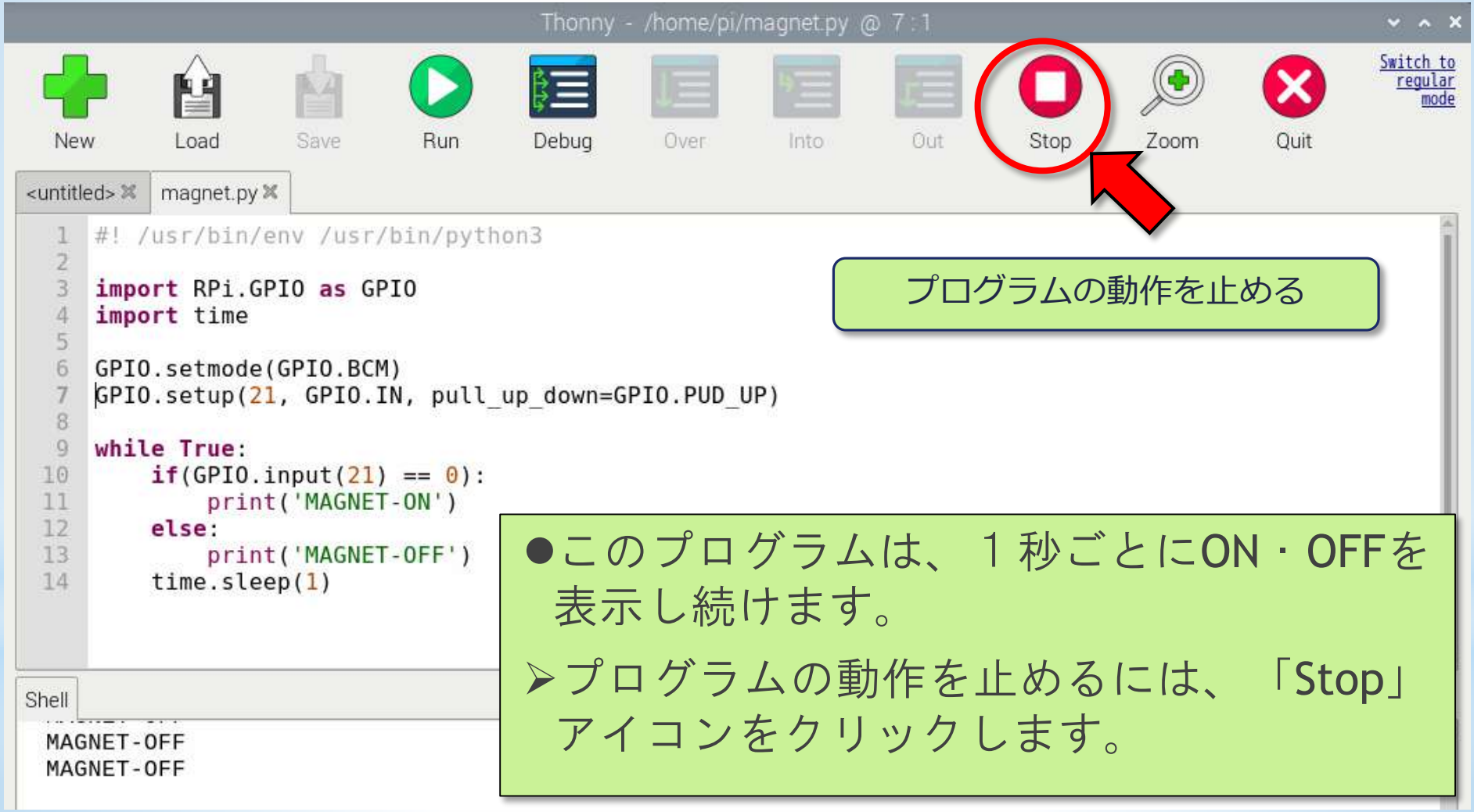
マグネットセンサーの体験



- 実行結果が、画面の下半分に表示されます。
  - 「MAGNET-ON」か「MAGNET-OFF」のどちらかが表示されましたか？
- マグネットセンサーに磁石を近づけたり遠ざけたりしてみてください。
  - 表示が変わりますか？

3　マグネットセンサー

マグネットセンサーの体験



プログラムの動作を止める

- このプログラムは、１秒ごとにON・OFFを表示し続けます。
- プログラムの動作を止めるには、「Stop」アイコンをクリックします。

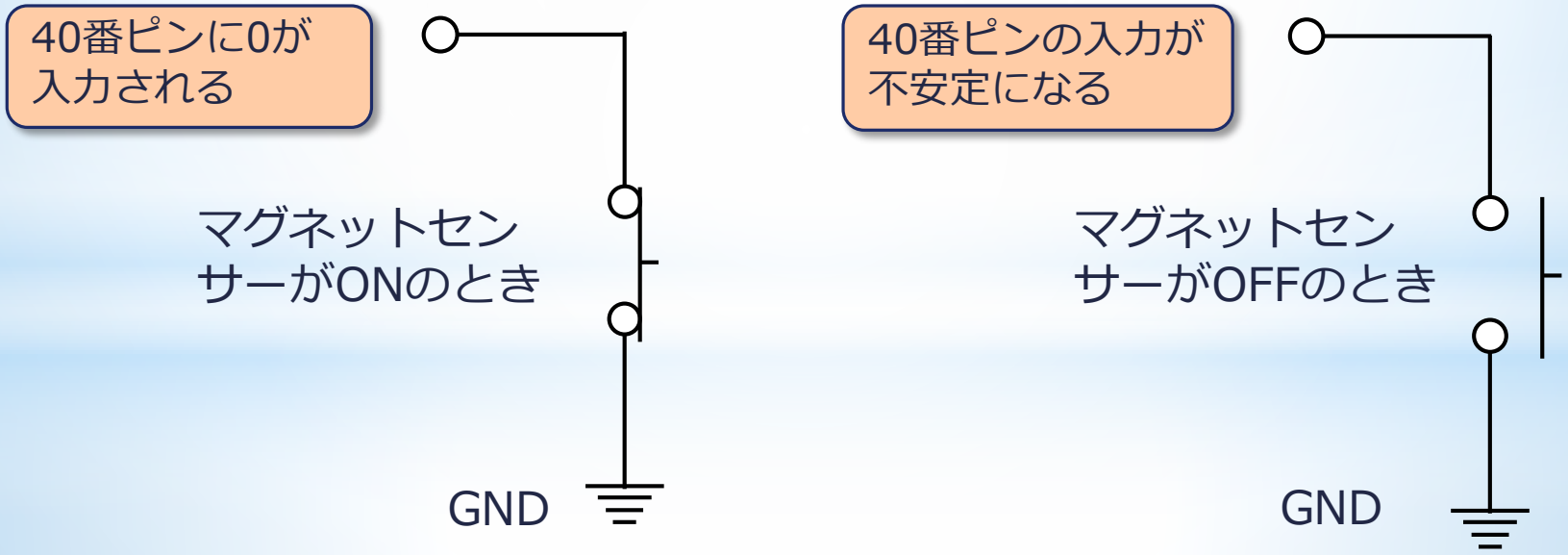
3　マグネットセンサー

GPIOの入力を安定させる方法

- マグネットセンサーがONになる。⇒ 40番ピンがGNDにつながる。

  ⇒ 40番ピンの入力は0になる。
- マグネットセンサーがOFFのときは？ ⇒ 40番ピンはどこもつながっていない。⇒ 入力は「1」と認識される？「0」と認識される？

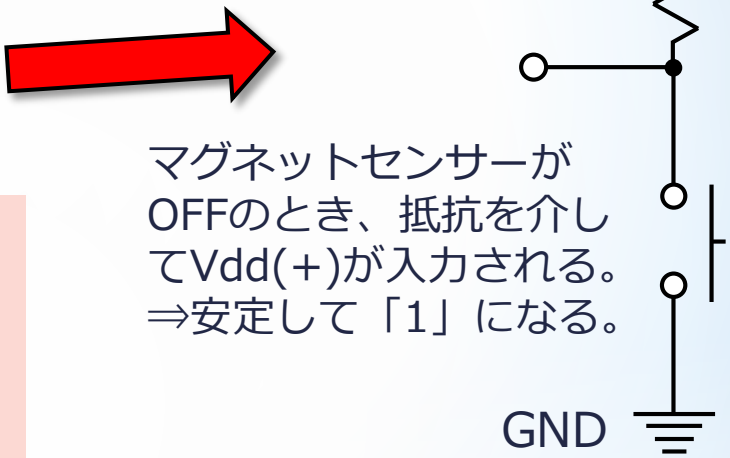
  ・・・ノイズなどを拾って不安定な状態になる。



3　マグネットセンサー

GPIOの入力を安定させる方法

- 安定させるためには「プルアップ」と呼ばれる方法を使う必要がある。
  - ✓ 参考書籍p199～
- プルアップは、抵抗と＋電源を使うのが本来の方法。



- ラズパイでは、プルアップの回路が内蔵されているので、プログラムでプルアップの設定をすれば抵抗などの接続を省略できる。

スイッチ（マグネットセンサー等）の接続方法によっては、「＋」と「－」を接続が逆になる「プルダウン」を使うこともあります。
プルダウンの説明は割愛しますが、詳しくは参考書籍のp199～をご覧ください。

４　カウンター

（注）各章で実施するブレッドボードとラズパイの電子工作は、追加で実施してください。（前の章で作成した配線は、外さずにそのままにしておいてください。）

4　カウンター

まずはプログラムの簡単な説明

```
1  #! /usr/bin/env /usr/bin/python3
2
3  import RPi.GPIO as GPIO
4  import time
5
6  GPIO.setmode(GPIO.BCM)
7  GPIO.setup(21, GPIO.IN, pull_up_down=GPIO.PUD_UP)
8
9  while True:
10     if(GPIO.input(21) == 0):
11         print('MAGNET-ON')
12     else:
13         print('MAGNET-OFF')
14     time.sleep(1)
```

GPIO を使うための準備

GPIO 21（40番ピン）を「入力」に設定する。

永続的に繰り返す

GPIO 21の入力を調べて処理を分岐する

「MAGNET-ON」と表示する

「MAGNET-OFF」と表示する

1 秒待機する

4　カウンター

Pythonの学習（プログラムの説明）

- **import**　プログラムで使ういろいろな機能がセットになった「ライブラリ」と呼ばれるものを使うための準備のようなもの。

import time

時間関係の処理をするためのライブラリを使う準備

import RPi.GPIO as GPIO

ラズパイのGPIO関係の処理をするためのライブラリを使う準備。(import ～)

プログラム中ではその機能を、「GPIO」という名前を使って呼び出す。(as ～)

理解を深めるために・・・

最初の「import time」を消して実行してみましょう。

✓ 「time.sleep(1)」のところにエラーがあるという趣旨の表示が出る。

✓ 「time.sleep」を使うのには、「import time」という準備が必要。

⇒もう一度「import time」を入力してエラーが出ないようにしましょう

4　カウンター

Pythonの学習（プログラムの説明）

- **GPIO.setmode()**　GPIOを使う際のモードのセット

GPIO.setmode(BCM)

GPIOを使う際のモードをBCMにする。✓詳しくは参考書籍p166

- **GPIO.setup()**　入出力ピンの設定

GPIO.setup(21, GPIO.IN, pull_up_down=GPIO.PUD_UP)

“GPIO 21”を、入力用として使い、プルアップを設定する

端子（ピン）の物理的な番号と、GPIOの番号とは異なります。
ここまでの「GPIO」は、40本全体のコネクタを指すことが多かったですが、個々のピンを指して「GPIO」と呼ぶこともあります。紛らわしいですが、注意して進めてください。

「GPIO 21」 は、ピンの物理的な番号では40番ピンです。

✓ 物理的なピン番号とGPIO番号の対応は、参考書籍p163を参照

4　カウンター

Pythonの学習（プログラムの説明）

【プログラムの流れの制御】

- **while ○○：**　○○の条件が成立する間、繰り返し続ける

while True:

永続的に繰り返す。

✓条件式（○○の部分）が「True」（「真」である。）

⇒「常に条件成立」と作り手が明記。つまり永続的に繰り返す

- pythonでは、この繰り返しの範囲をインデントで指定する。

```
 9 while True:
10     if(GPIO.input(21) == 0):
11         print('MAGNET-ON')
12     else:
13         print('MAGNET-OFF')
14     time.sleep(1)
```

インデントされているこの範囲が、永続的に繰り返す範囲

4　カウンター

Pythonの学習（プログラムの説明）

【プログラムの流れの制御】

- **if (〇〇)：**　〇〇の条件が成立する場合は、次の処理をする
- **else：**　〇〇の条件が成立しなかった場合は、次の処理をする

```
if(GPIO.input(21) == 0):
        GPIO 21の入力が0だったら、次の処理をする。
        ✓「等しい」は、「=」を2個つなげて表します。
else:

        GPIO 21の入力が0でなかったら、次の処理をする。
```

- “条件によって処理が変わる範囲”も、インデントで指定する。

```
 9  while True:
10      if(GPIO.input(21) == 0):
11          print('MAGNET-ON')
12      else:
13          print('MAGNET-OFF')
14      time.sleep(1)
```

条件が成立したときに処理する範囲

条件が成立しなかったときに処理する範囲

4　カウンター

Pythonの学習（プログラムの説明）

- **print(’〇 〇’)**　「〇〇」と表示する

print(’MAGNET-ON’)
　　「MAGNET-ON」と表示する。

print(’MAGNET-OFF’)
　　「MAGNET-OFF」と表示する。

- **time.sleep(〇 〇)**　〇〇秒待機する

time.sleep(1)
　　1秒待機する

理解を深めるために・・・
「time.sleep()」の()の数字を変えて、実行してみましょう。
表示の間隔が変わりましたか？（0.5など、小数を入力しても構いません。）

4　カウンター

Pythonの学習

理解を深めるために・・・
　最後のtime.sleep(1)の次の行に、もう1行「time.sleep(5)」と追加して実行してください。

```
 9 while True:
10     if(GPIO.input(21) == 0):
11         print('MAGNET-ON')
12     else:
13         print('MAGNET-OFF')
14     time.sleep(1)
15     time.sleep(5)
```

⇒1秒待機後、続いて5秒待機するため、
　計6秒待機して、表示が繰り返されます。

追加した「time.sleep(5)」のインデントを先頭に戻して、実行してください。

```
 9 while True:
10     if(GPIO.input(21) == 0):
11         print('MAGNET-ON')
12     else:
13         print('MAGNET-OFF')
14     time.sleep(1)
15 time.sleep(5)
```

“1秒待機”は繰り返しの範囲内
　⇔“5秒待機”は繰り返しの範囲外

⇒表示の繰り返しは、1秒間隔

※この例では、繰り返しを抜けることはないので、実際に5秒待機することはありません。

4　カウンター

Pythonの学習

理解を深めるために・・・
　追加した「time.sleep(5)」を、「if ～」の次の行に移動して実行してください。

```
 9  while True:
10      if(GPIO.input(21) == 0):
11          time.sleep(5)
12          print('MAGNET-ON')
13      else:
14          print('MAGNET-OFF')
15      time.sleep(1)
```

⇒ ifの条件が成立したとき（マグネットセンサーがONのとき）だけ、5秒待機してから表示します。

もし、「if」と「else」の間の行のインデントを前に戻したら・・・？

if～の範囲から抜けた処理があるのに、その後に「else」が書かれているため、「else」のところでエラーになる。

```
 9  while True:
10      if(GPIO.input(21) == 0):
11          time.sleep(5)
12      print('MAGNET-ON')
13      else:
14          print('MAGNET-OFF')
15      time.sleep(1)
```

Shell

```
Traceback (most recent call last):
  File "/home/pi/magnet.py", line 13
    else:
    ^
SyntaxError: invalid syntax
```

4　カウンター

カウンタープログラムの作成

- 「ONかOFFか」を表示するのではなく、ONの数やOFFの数を数えるには？

➢「変数」を使います。

- 変数とは？ ⇒ プログラムの中でデータを入れておく入れ物
- カウントしたい数を変数に入れておき、条件に応じて加算していく。

変数を使うには、
　変数名 = 値
と記述します。

◆「=」は等しいという意味ではなく、代入を意味します。

例：
cnt = 0
cnt = cnt + 1

※プログラムの中で最初に記述した場所
これ以降「cnt」という名前の変数が使える。
まずは、cntに0を代入する。

cntには0が入っていたので、0+1=1となり、cntには1が代入される。

4　カウンター

カウンタープログラムの作成

例えば、マグネットセンサーでドアの開閉回数を数えるには？
マグネットセンサーをドアに付けて、その状態を調べて、「ドアが開いた」と判断したら、変数を１増やす。

例：

```
cnt = 0
if(GPIO.input(21) == 1):
    cnt = cnt + 1
print (cnt)
time.sleep(1)
```

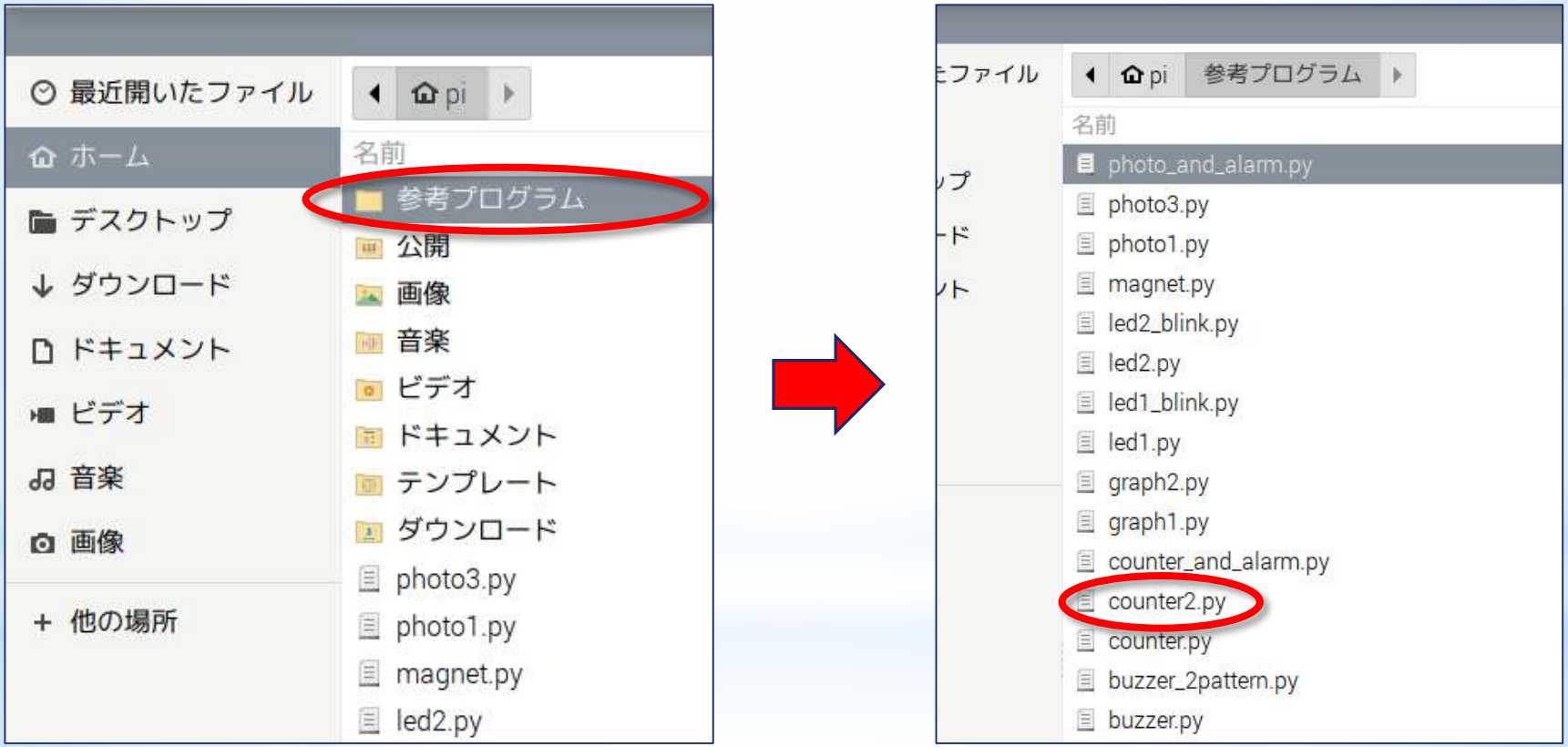
ドアが閉じているときに磁石がセンサーに付くような設置をイメージ。
ドアが開く⇒磁石がセンサーから離れる⇒センサーOFF⇒GPIOの入力は1

- 「counter.py」のというプログラムを開いてください。
- 中身は「magnet.py」と同じです。これを修正して、動かしてみましょう。
- 磁石を近づけたり遠ざけたりして、カウントされるか試してください。

4　カウンター

作った（修正した）プログラムが上手く動かない場合は

◆Pythonで作った完成版のファイルが「参考プログラム」というサブフォルダの中に、「counter2.py」というファイル名で入っています。



◆自分で作ったプログラムと比べて、修正してみましょう。

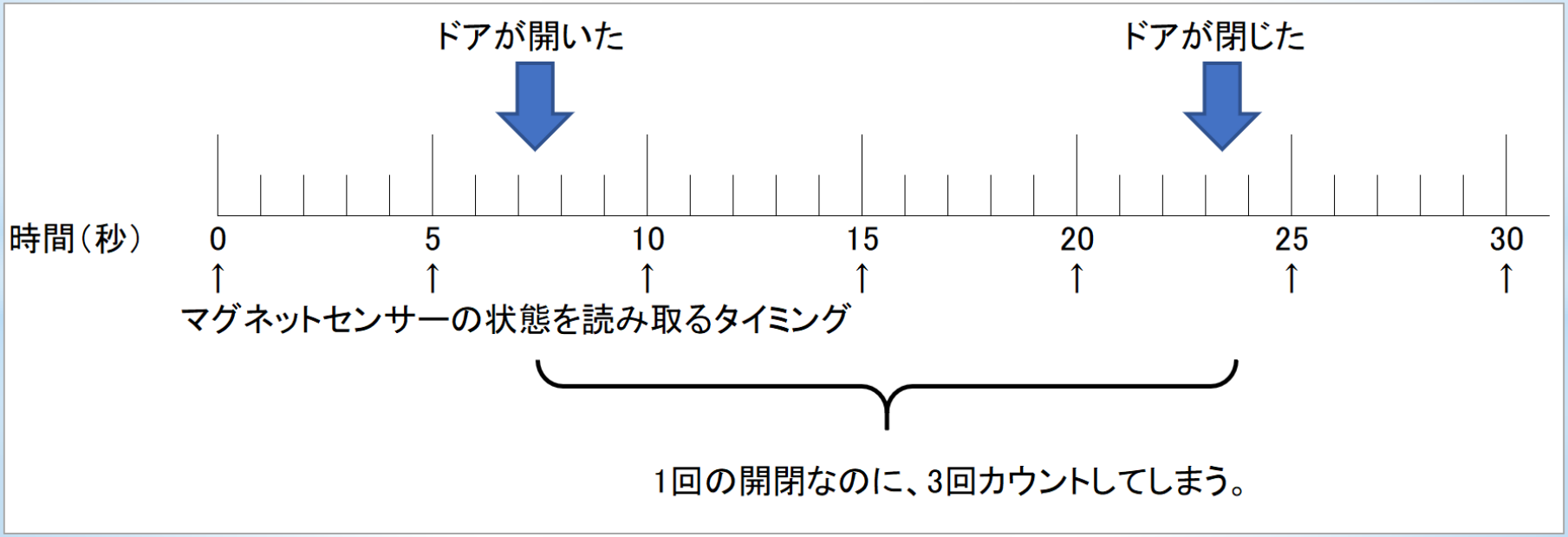
4　カウンター

カウンタープログラムを作る時の検討事項

マグネットセンサーの状態を調べる頻度はどのくらいが適切か？
仮に、５秒に１回調べたら・・・？
※ time.sleepを5秒に変更すれば、5秒に1回調べる動作になります。

●これで、意図したとおりにドアが開いた回数を数えられるでしょうか？

⇒ 一度開いてから閉じるまで１５秒くらいかかったら、１回しか開いていないのに、３～４回分カウントしてしまうかも。

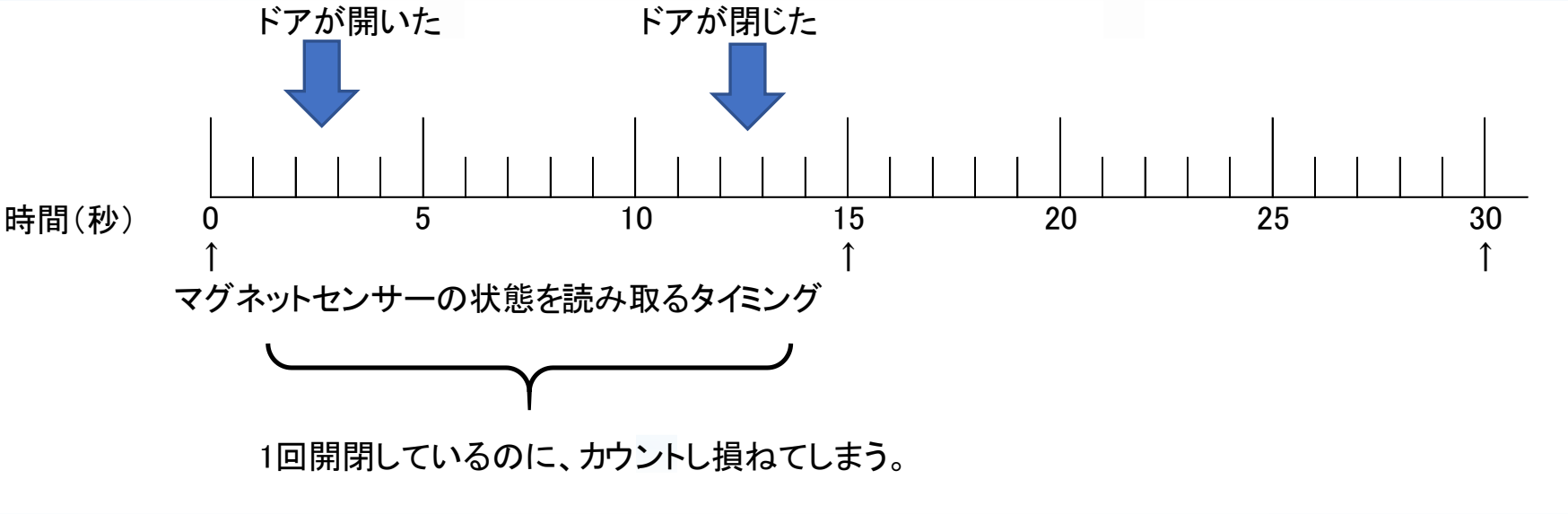


4　カウンター

カウンタープログラムを作る時の検討事項

●重複してカウントしないように、頻度を15秒に１回にしたら・・・？

⇒ 一度開いてから閉じるまで10秒で済んでしまったら、カウントし損ねてしまう。



この例の場合、状態を調べる頻度を変えるだけでは、
正しくカウントするのは難しい。

4　カウンター

カウンタープログラムを作る時の検討事項

●では、どのようなプログラムにすればよいでしょう？

⇒短時間に何度も状態を調べ、閉じた状態から開いた状態に変化したらカウントする。

具体的なプログラム例は、後日掲載します。

もし、機械の動作回数を数えたかったら・・・

- 1回の動作にかかる時間は？
- 動作と動作の間隔は？
- どこにセンサーを付けたらカウントしやすいか？

・・・

プログラムの知識だけでなく、機械に対する知識や観察力、つまり現場のノウハウが重要です！

5　フォトセンサー

（注）各章で実施するブレッドボードとラズパイの電子工作は、追加で実施してください。（前の章で作成した配線は、外さずにそのままにしておいてください。）

5　フォトセンサー

フォトセンサーについて

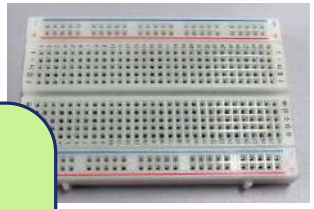
- 今度は、フォトセンサーを使ってみます。
- フォトセンサーは「明るさ」を検知するセンサーです。
  - ➢本来は、“どのくらい明るいか”という“値”を取得できるセンサーです。（アナログセンサー）
  - ➢今回は、明るいか・暗いか、だけを判断する用途で使います。（デジタル的に使う。）
- 今回使うセンサーには「グランド (GND) 」と「電源」、「信号線」をつなぎます。（同様のつなぎ方をするセンサーは多い。）
- どのようにつなぐかは、センサーによって（あるいはセンサーが準拠している規格によって）決まっています。
- ラズベリーパイから、センサー用の電源も供給できます。(+5Vと+3.3Vを供給できる。）
  - ➢今回は3.3Vを使います。

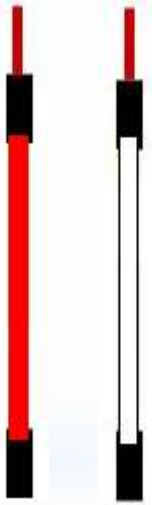
5　フォトセンサー

フォトセンサーの状態を取得

フォトセンサーに使う線をラズパイからブレッドボードまで接続する

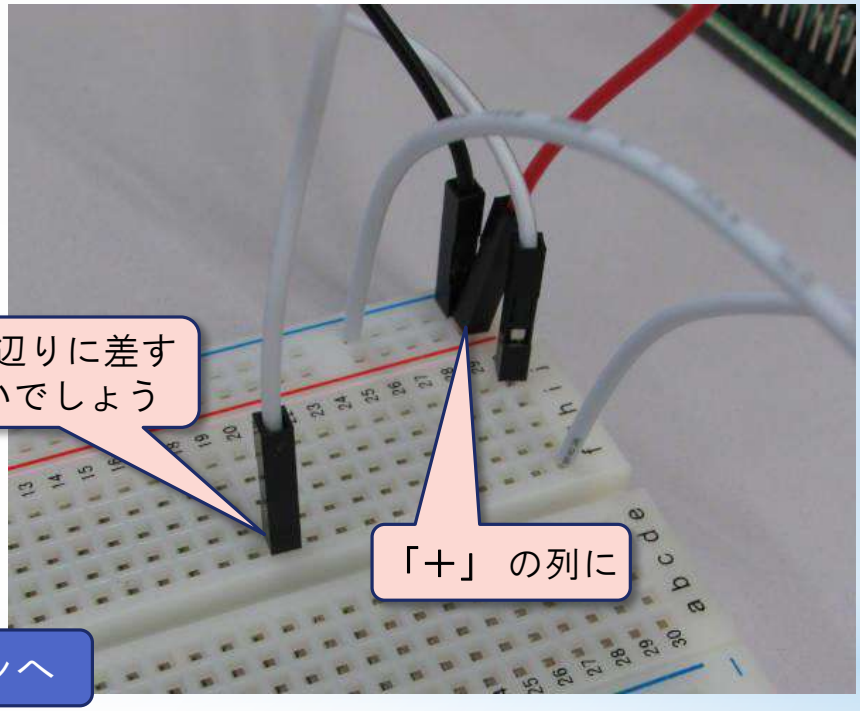
白：センサーの信号線との接続用（ブレッドボードの(f,20) 辺りが目安）
赤：センサーの電源との接続用（ブレッドボードの「＋」の列に差す）
GNDは？
⇒既にブレッドボードに接続済み





17番ピンへ

21番ピンへ



2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39

GPIOのピン配置と接続箇所

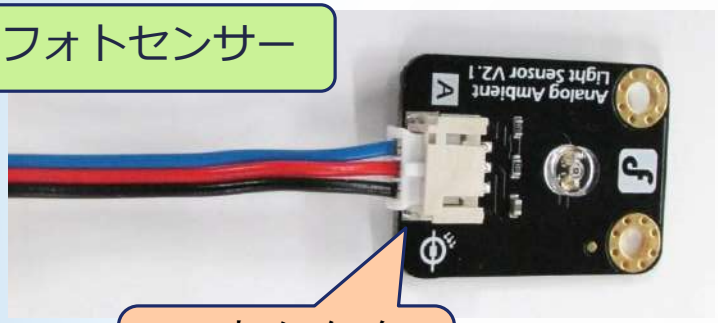
復習：
GPIOの各端子の番号と用途は、参考書籍p163を参照

5　フォトセンサー

フォトセンサーの状態を取得

フォトセンサーに、ブレッドボードにつなぐジャンパー線を接続する

フォトセンサー

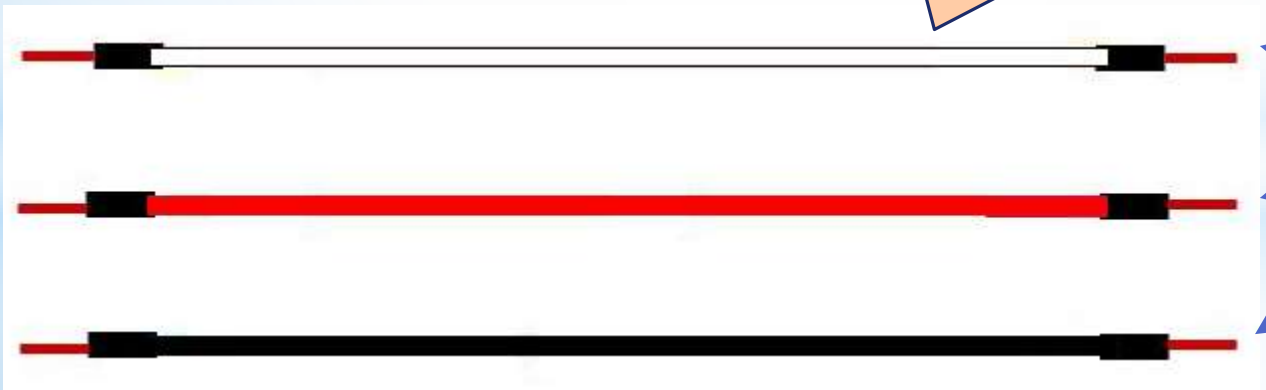


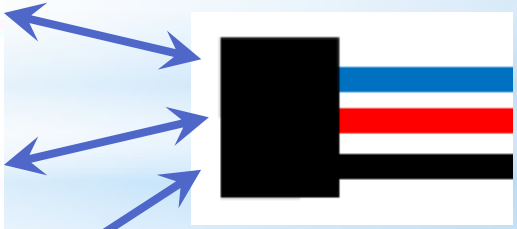
コネクタを
接続

白のジャンパー線
⇒ センサーの青（信号線）に接続
赤のジャンパー線
⇒ センサーの赤（電源）に接続
黒のジャンパー線
⇒ センサーの黒（グランド）に接続

ケーブルの
色を確認

ブレッドボード側



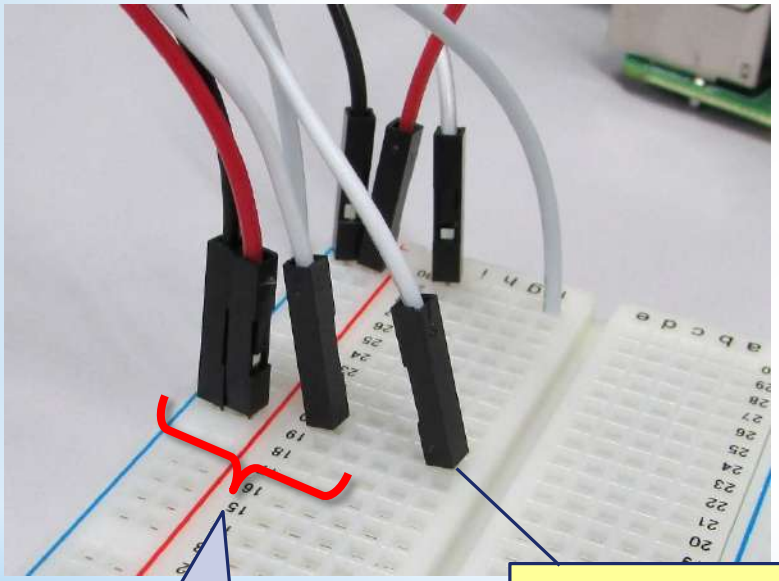


フォトセン
サーの線

5　フォトセンサー

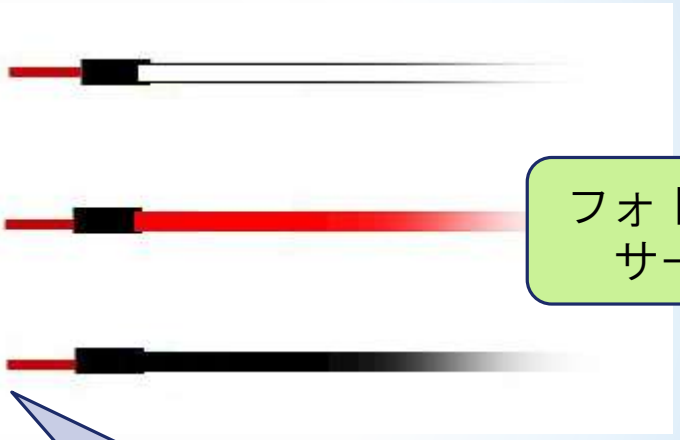
フォトセンサーの状態を取得

フォトセンサーにつないだジャンパー線の反対側をブレッドボードに差す



ラズパイに
つなげた線

フォトセンサーからの
ジャンパー線を差す



ブレッドボード側

フォトセンサー側

白のジャンパー線
⇒ 先に白のジャンパー線を差した列に
赤のジャンパー線
⇒ 「＋」の列に
黒のジャンパー線
⇒ 「－」の列に

5　フォトセンサー

フォトセンサーの状態を取得

- フォトセンサーを接続したら、ラズパイの電源をONにしてください。
- プログラムの確認のために再びThonnyを使います。
  ✓Thonnyの使い方は？　⇒　スライドのNo26～No31を復習
- 今度は「**photo1.py**」というプログラムファイルを選びます。
- このプログラムは未完成です。マグネットセンサーのプログラムを参考にして、完成させてください。

```
1  #! /usr/bin/env /usr/bin/python3
2
3  import RPi.GPIO as GPIO
4  import time
5
6  GPIO.setmode(GPIO.BCM)
7  GPIO.setup(9, GPIO.IN)
8
9  while True:
10     if(GPIO.input() == 1): #未完成の部分を埋めてください
11         print('PHOTO-ON')
12
13 #これより下を作ってください。
14
```

入力を調べるGPIOの番号を入れる。

ifの条件が成立しなかったときの処理を書く。

待機の処理を書く

5　フォトセンサー

フォトセンサーの状態を取得

- 完成例は次のようになります。

```
1  #! /usr/bin/env /usr/bin/python3
2
3  import RPi.GPIO as GPIO
4  import time
5
6  GPIO.setmode(GPIO.BCM)
7  GPIO.setup(9, GPIO.IN)
8
9  while True:
10     if(GPIO.input(9) == 1): #未完成の部分を埋めてください
11         print('PHOTO-ON')
12     else:
13         print('PHOTO-OFF')
14     time.sleep(1)
```

補足

プルアップ（プルダウン）の設定をしなくていいの？
⇒“GPIOピンがどこにもつながっていない”という状態にはならないので、プルアップ（プルダウン）の設定は不要です。

補足

「#」より後ろの記載は『コメント』と呼ばれ、プログラムとはみなされません。人間がプログラムを見るときに分かりやすいようにメモ、注釈等を記載するために使われます。

完成したら実行してみてください。「PHOTO-ON」「PHOTO-OFF」が表示されましたか？（センサーを明るい方に向けたり、手で覆って暗くしたりしてください。）

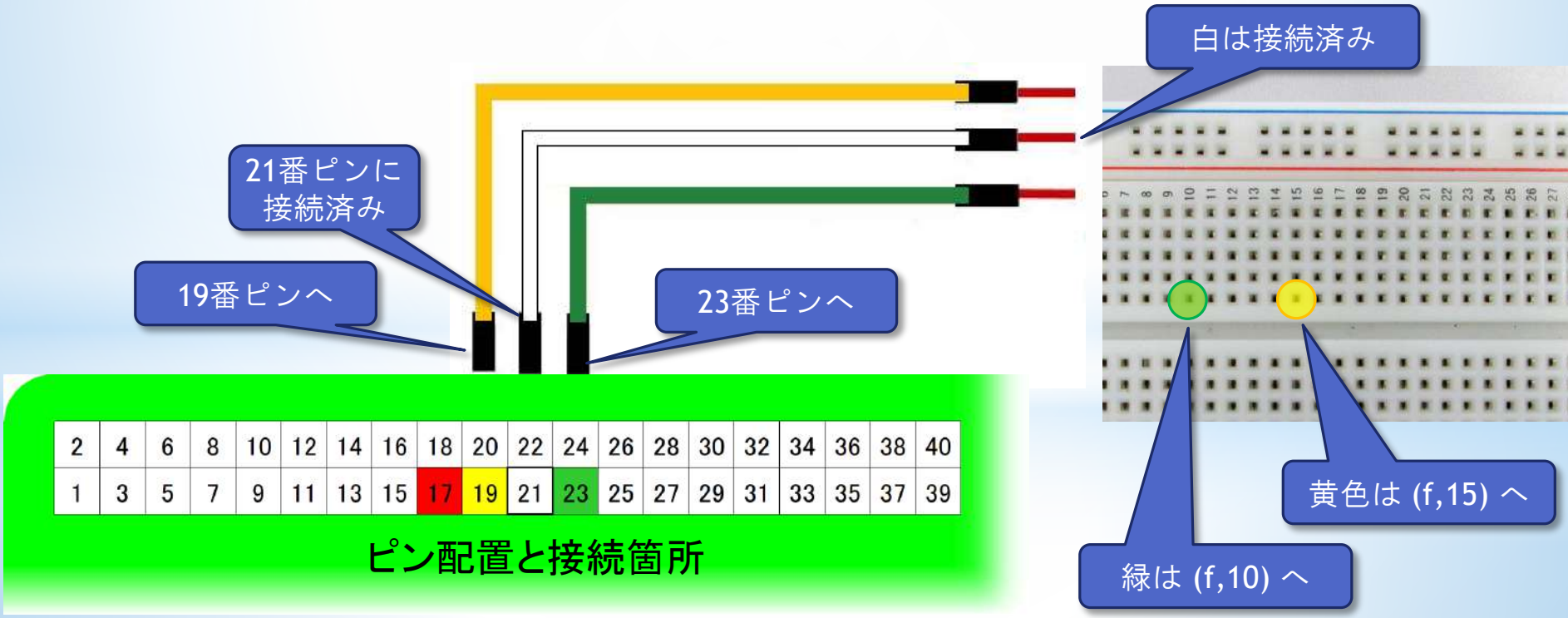
5　フォトセンサー

フォトセンサーを増やす

●続いてフォトセンサーを2個追加し、計3個にしてみましょう。

✓ ラズパイは一度シャットダウンしてください。

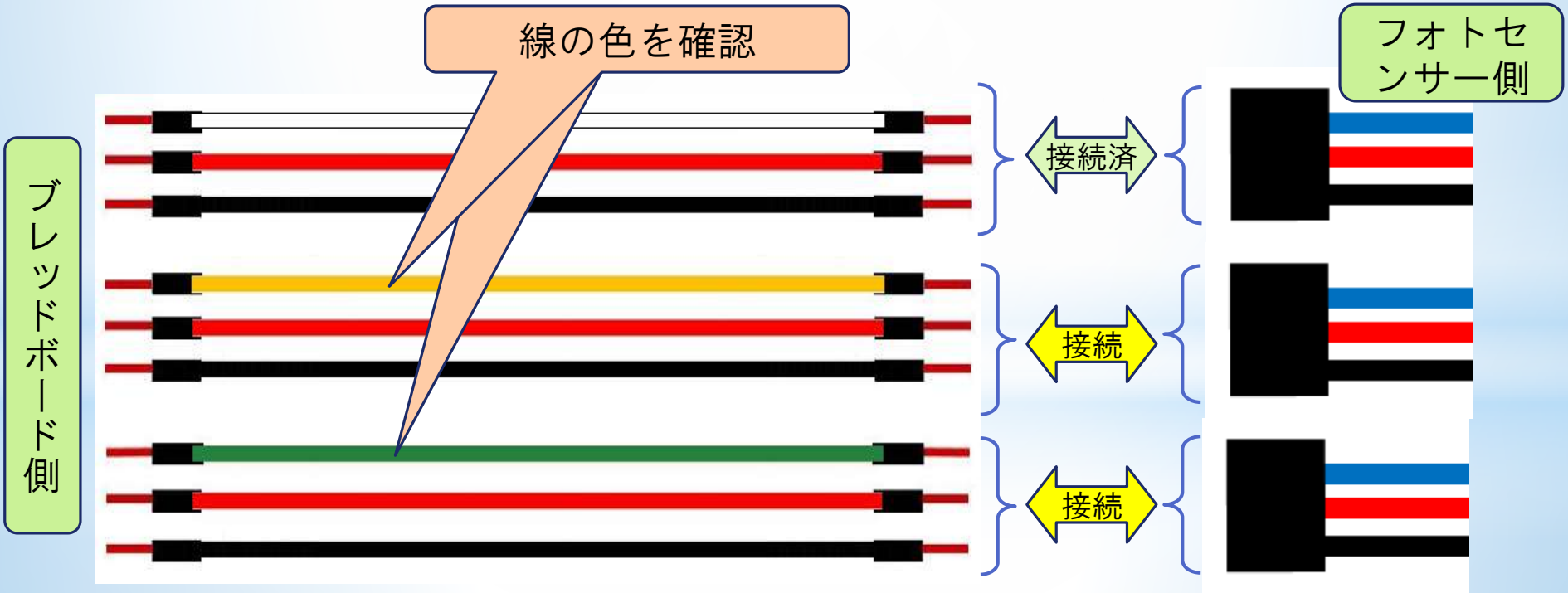
- 信号用の線をラズパイからブレッドボードまでつなぎますが、先ほどと違う色を使いましょう。（黄色と緑を使ってください。）
- 黄色は19番ピンへ、緑は23番ピンへ差してください。



5　フォトセンサー

フォトセンサーを増やす

- フォトセンサーとブレッドボードの間も、先ほどと同様に接続します。
- 信号用の線は、ラズパイ⇔ブレッドボードをつないだ色と合わせましょう。



5　フォトセンサー

フォトセンサーを増やす

- フォトセンサーの追加を接続したら、電源をONにしてください。
- プログラムの確認のために再びThonnyを使います。
  - ✓Thonnyの使い方は？
    - ⇒　スライドのNo26～No31を復習
- 今度は「**photo3.py**」というプログラムファイルを選びます。
- これも未完成です。入力済みの部分を参考に、フォトセンサー3個分の処理を完成させてください。
  - ✓今までよりも繰り返しの範囲が長くなります。
- 完成したら実行しましょう。
- 今度は、「RED-OFF」や「YELLOW- ON」のように色とON・OFFが表示されます。

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(9, GPIO.IN)
GPIO.setup(10, GPIO.IN)
GPIO.setup(11, GPIO.IN)

while True:
    if(GPIO.input(9) == 1):
        print('RED-ON')
    else:
        print('RED-OFF')
    if(GPIO.input(10) == 1):
        print('YELLOW-ON')
    else:
        print('YELLOW-OFF')
    if(GPIO.input(11) == 1):
        print('GREEN-ON')
    else:
        print('GREEN-OFF')
    time.sleep(3)
```

Shell

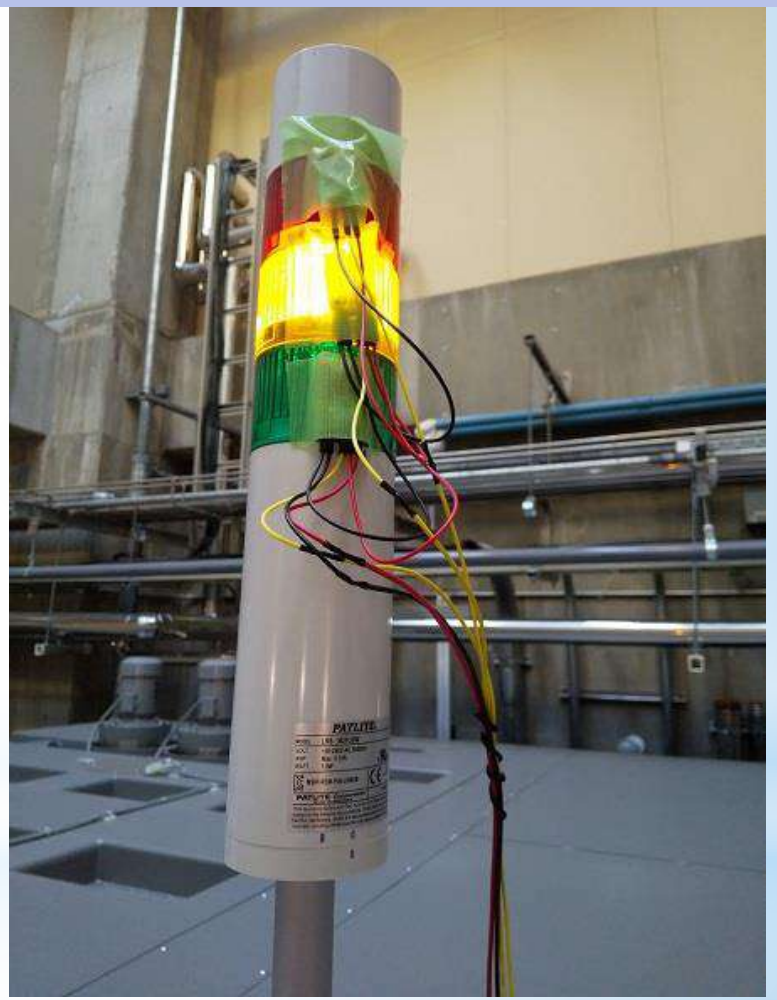
```
>>> %Run photo3.py

RED-OFF
YELLOW-ON
GREEN-ON
RED-OFF
YELLOW-ON
GREEN-ON
```

5　フォトセンサー

フォトセンサーの設置イメージ

- 今回使っているフォトセンサーが色を感知しているのではありません。
- 白のジャンパー線のセンサーは、「RED-ON」または「RED-OFF」と表示されます。
- 同様に黄色のジャンパー線のものは「YELLOW-ON」または「YELLOW-OFF」と表示され、緑のジャンパー線のものは「GREEN-ON」または「GREEN-OFF」と表示されます。
- 3色の表示灯にセンサーを付けて、状態を監視する使い方をイメージして、このような表示にしています。



フォトセンサーの設置イメージ

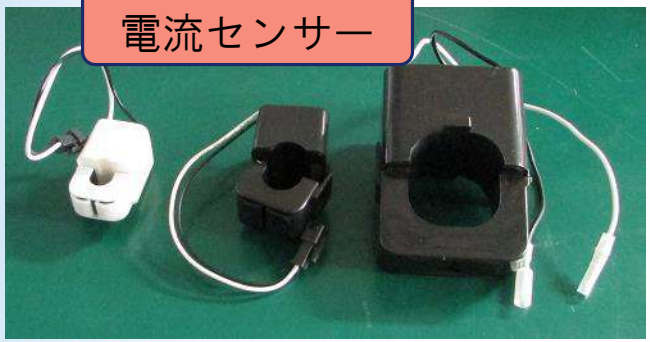
補足（その他のセンサーの例）

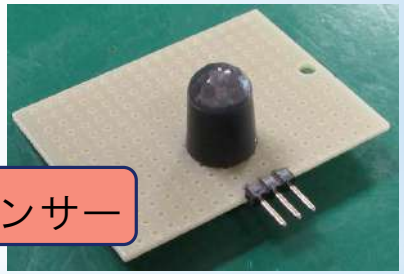
◆本研修では、マグネットセンサーとフォトセンサーを取り上げました。

◆その他にも様々なセンサーがあります。

◆温度センサー、電流センサー、振動センサー、加速度センサー、超音波センサー、人感センサー・・・など。

◆カメラも付けられます。





ラズパイで使う上での注意点

●アナログ出力のセンサーの場合

⇒ A/Dコンバーターでデジタル信号への変換が必要

※ 詳細は参考書籍 p216～を参照

●ON-OFFではないデジタル通信で出力するセンサーの場合

⇒ 規格を確認し、ラズパイでその規格の設定が必要

※ 詳細は参考書籍p210～ 及び p21のKEY WORD を参照

6　ブザーによるお知らせ

（注）各章で実施するブレッドボードとラズパイの電子工作は、追加で実施してください。（前の章で作成した配線は、外さずにそのままにしておいてください。）

6　ブザーによるお知らせ

電子ブザーについて

- ここまで、マグネットセンサーとフォトセンサーで情報を取得することをやりました。GPIOを入力として使いました。
- 今度は、電子ブザーによるお知らせをやります。この場合、GPIOを出力として使います。
- GPIOを出力として使う場合の注意点

✓GPIOから出力できるのは16mAまで。少しの電流しか流せない。

✓GPIOからの出力電圧は3.3V

⇒ 本研修で使う電子ブザーは、動作電圧3～18V、消費電流10mAとなっているものを選定しましたので、そのまま使えます。

※本研修で使う電子ブザーには極性(＋－の別)があります。注意してください。

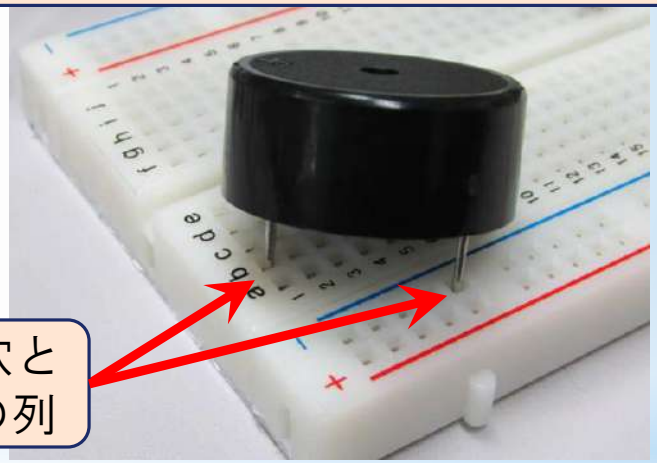
6　ブザーによるお知らせ

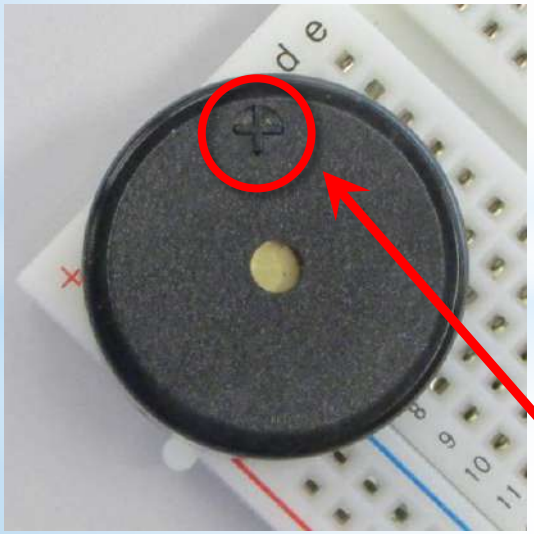
電子ブザーを鳴らす

まず、ブレッドボードに電子ブザーを差しましょう。

※写真のように、(b,1) の穴と「－」の列を使うと、うまく差せると思います。

この写真は穴の位置の説明用です。
電子ブザーは奥までしっかり差し込んでください。



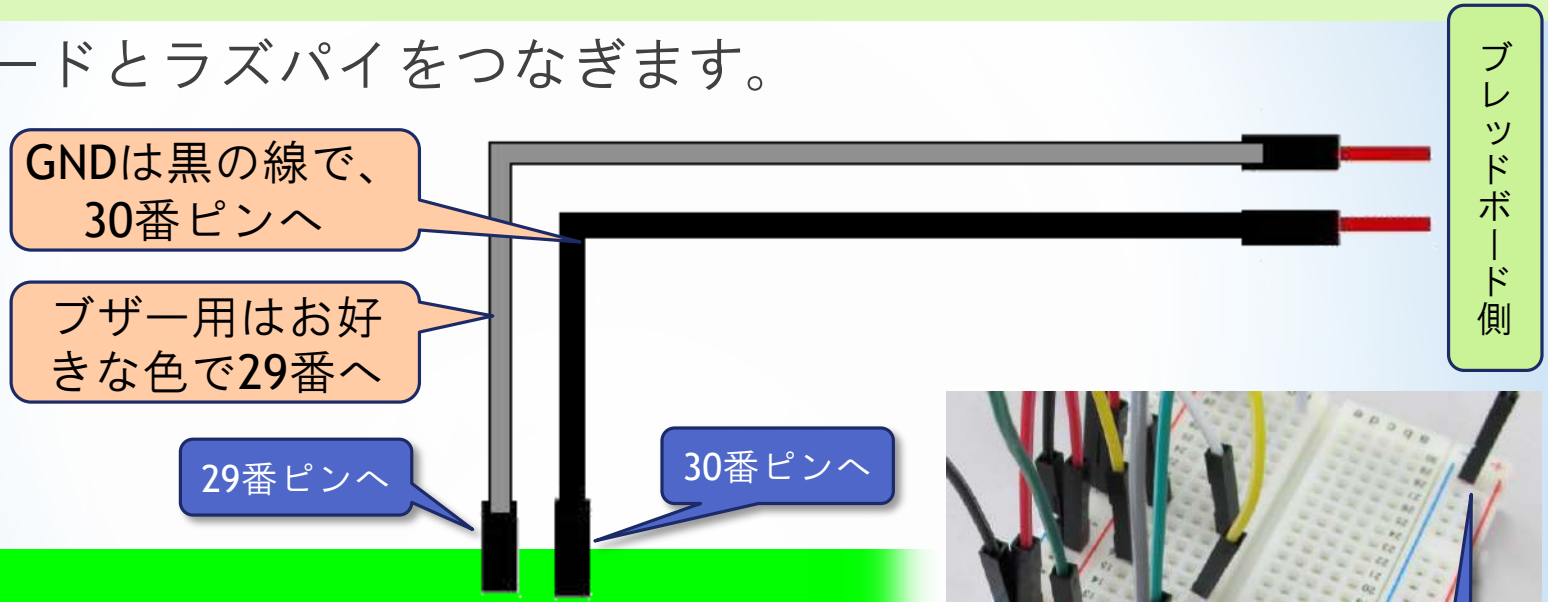


※極性に注意してください。
(b,1)には「＋」マークのある方のピンを差します。

6　ブザーによるお知らせ

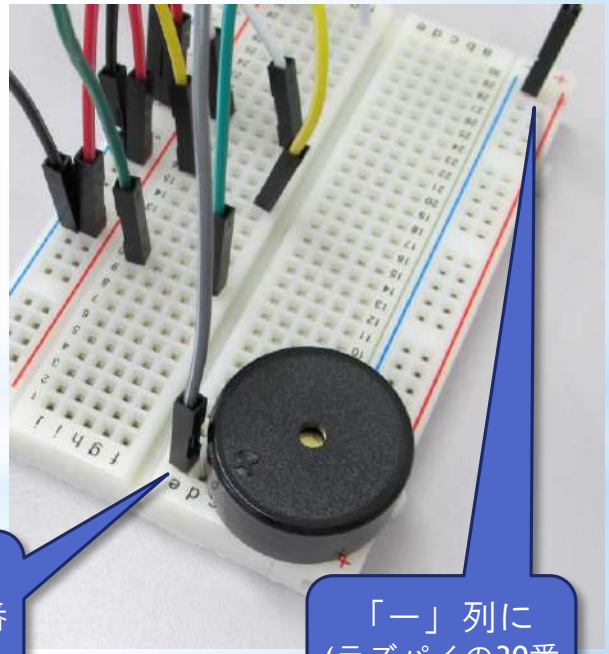
電子ブザーを鳴らす

ブレッドボードとラズパイをつなぎます。



2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39

ピン配置と接続箇所



- ✓ ブレッドボードの左右にある「－」の列同士、「＋」の列同士はつながっていません。
- ✓ 先ほどまでと反対の「－」列を使うので、改めてラズパイのGNDと接続します。

6　ブザーによるお知らせ

電子ブザーを鳴らす

ブザーを接続したら、ラズパイの電源を入れましょう。

●ブザー用のプログラム「buzzer.py」を開く。

```
1  #! /usr/bin/env /usr/bin/python3
2
3  import RPi.GPIO as GPIO
4  import time
5
6  GPIO.setmode(GPIO.BCM)
7  GPIO.setup(5, GPIO.OUT)
8
9  GPIO.output(5, GPIO.HIGH)
10 time.sleep(3)
11 GPIO.output(5, GPIO.LOW)
12 GPIO.cleanup()
13
```

●このプログラムはこのままで動作します。Runで実行させてください。

●大きな音でブザーが鳴るので注意！

●3秒ブザーが鳴って、停止します。

6　ブザーによるお知らせ

電子ブザーを鳴らす（プログラムの説明）

- **GPIO.setup()**　入出力ピンの設定

```
GPIO.setmode(5, GPIO.OUT)
```

“GPIO 21”を、出力用として使う

- **GPIO.output()**　出力を設定する

```
GPIO.output(5, GPIO.HIGH)
```

“GPIO 5”の出力を「HIGH」（0 or 1 の「1」）にする。

⇒この場合はブザーが鳴る。

```
GPIO.output(5, GPIO.LOW)
```

“GPIO 5”の出力を「LOW」（0 or 1 の「0」）にする。

⇒この場合はブザーが止まる。

6　ブザーによるお知らせ

電子ブザーを鳴らす（プログラムの説明）

- GPIO.cleanup()　GPIOの状態を初期化する

初期化しないままプログラムが終了すると、
- GPIOの出力がそのままとなり、ほかの処理に影響を及ぼすこともある。（ブザーが鳴りっぱなしになるなど。）
- 次にプログラムを実行したとき、警告が表示される。（実際には問題なく動くことがほとんどです。）

〔参考〕『参考プログラム』の中の『00_GPIO_cleanup.py』を実行すると、GPIOが初期化されます。ブザーが鳴りっぱなしになってとにかく止めたい場合などには、このプログラムを実行してください。

先ほどのマグネットセンサーやフォトセンサーと組み合わせて、
・カウントが○回になったらブザーを鳴らす。
・フォトセンサーがONになったらブザーを鳴らす。
などの応用が考えられます。

７　ＬＥＤによるお知らせ

（注）各章で実施するブレッドボードとラズパイの電子工作は、追加で実施してください。（前の章で作成した配線は、外さずにそのままにしておいてください。）

7　ＬＥＤによるお知らせ

- 次はLEDを点灯させてみましょう。
- LEDは小電流でも点灯可能ですが、GPIOに直接つなぐと、電流が流れすぎてしまう恐れもあります。
- 抵抗を入れて、電流を制限します。

✓今回使うLEDは、赤と黄です。

✓どちらも、LEDにかかる電圧は約2V

✓抵抗は100Ωを選定。電流は13mA程度。

- LEDを使う場合の抵抗の計算は、参考書籍p182を参照。

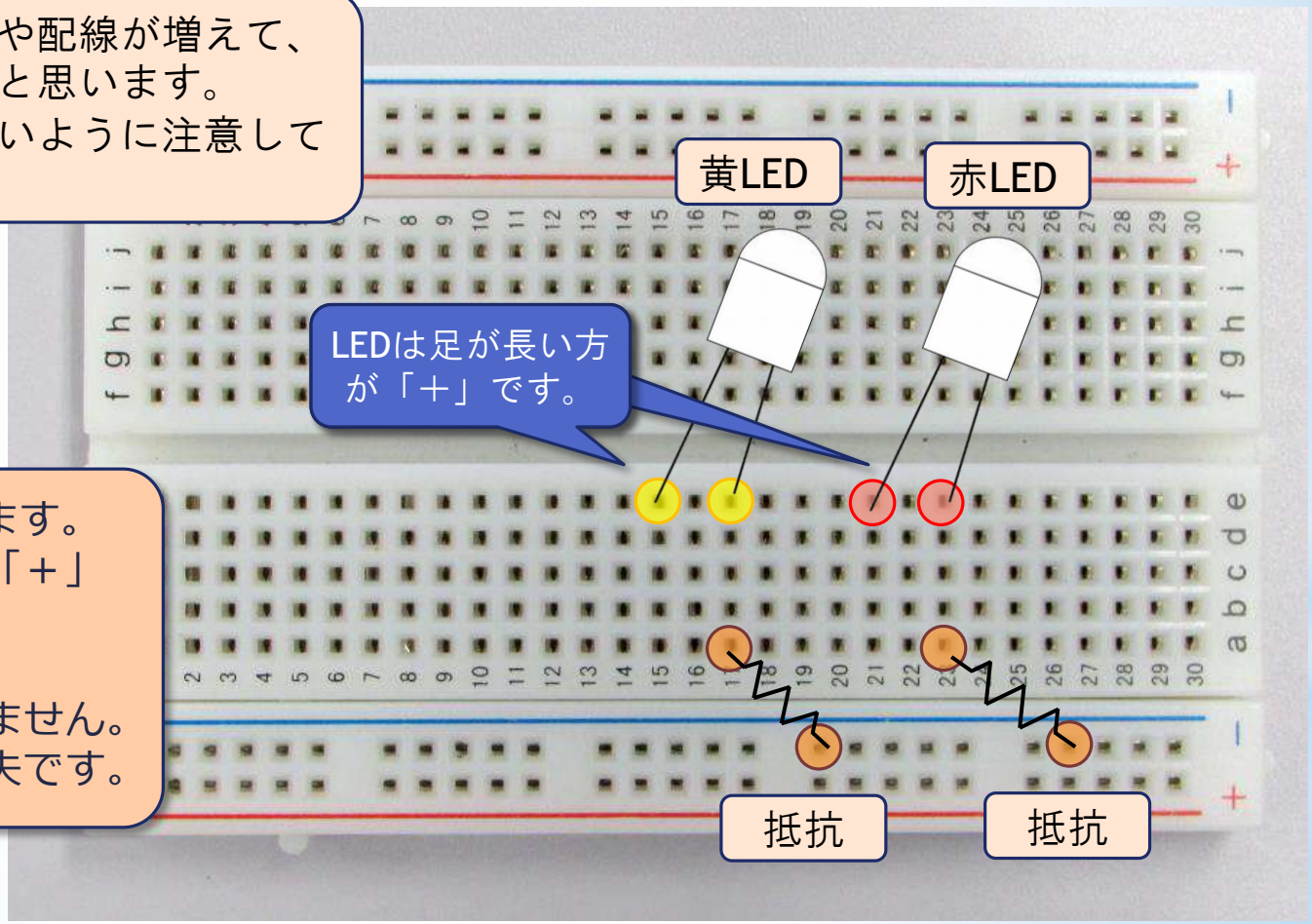
- 本研修で使うLEDは、先端に白いスモークが入っているものが黄色、先端までクリアなものが赤です。
- 分からなくなった場合は、とりあえずつないで、実際に点灯させてから確認しても構いません。
- 抵抗は、2本とも100Ωのものです。区別はありません。

7　ＬＥＤによるお知らせ

ＬＥＤを点灯させる

LEDと抵抗をブレッドボードに付けましょう。

ブレッドボード上の部品や配線が増えて、分かりにくくなってきたと思います。
この図を見て、間違えないように注意して部品を差してください。

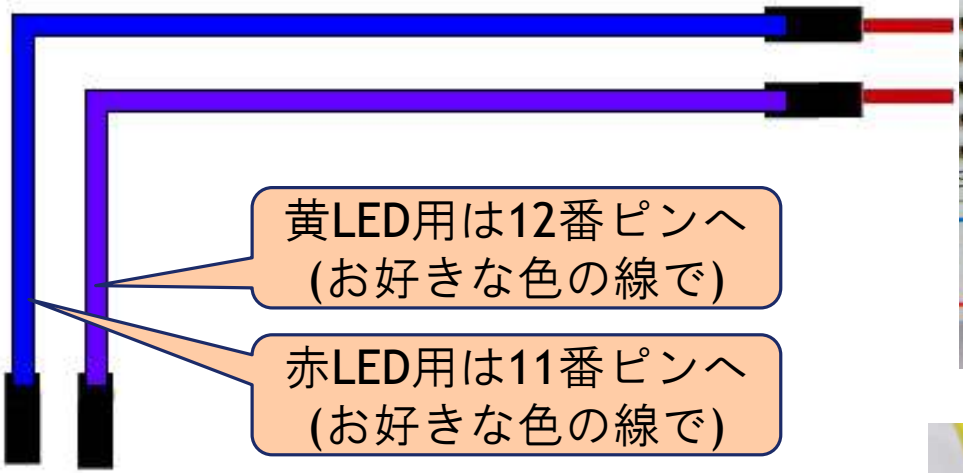


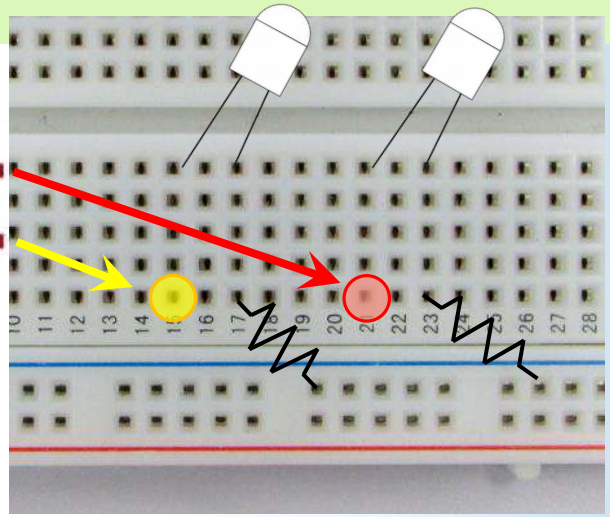
- ✓ LEDには極性があります。部品の足が長い方が「＋」です。
- ✓ 抵抗には極性がありません。どちら向きでも大丈夫です。

7　ＬＥＤによるお知らせ

ＬＥＤを点灯させる

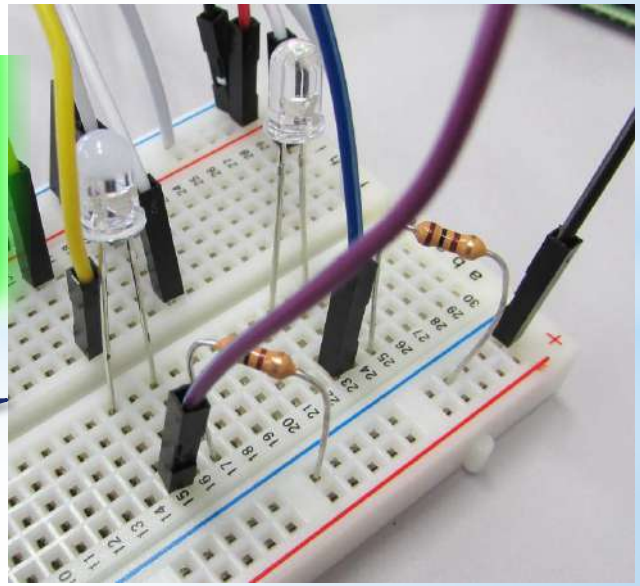
ブレッドボードとラズパイをつなぎます。





2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39

ピン配置と接続箇所



ブレッドボードの上がだいぶ
混雑してきたと思いますが、
こんな感じになります。

7　ＬＥＤによるお知らせ

ＬＥＤを点灯させる

LEDを接続したら、ラズパイの電源を入れましょう。

- LED用のプログラム「led1.py」を開いてください。
- このプログラムは未完成です。ブザーのプログラムを参考にして、 LEDを1個（赤LED）点灯させるプログラムを完成させてください。

```
1  #! /usr/bin/env /usr/bin/python3
2
3  import RPi.GPIO as GPIO
4  import time
5
6  GPIO.setmode(GPIO.BCM)
7  GPIO.setup(17, GPIO.OUT)
8
9  GPIO.output( , ) #未完成の部分を埋めてください
10 time.sleep() #LEDを点灯させたい時間（お好きな時間）を入力
11 GPIO.cleanup()
12
```

7　ＬＥＤによるお知らせ

ＬＥＤを点灯させる

完成例は以下のようになります。

```
1  #! /usr/bin/env /usr/bin/python3
2
3  import RPi.GPIO as GPIO
4  import time
5
6  GPIO.setmode(GPIO.BCM)
7  GPIO.setup(17, GPIO.OUT)
8
9  GPIO.output(17, GPIO.HIGH)
10 time.sleep(3)
11 GPIO.cleanup()
12
```

完成したら実行してみてください。
LEDが点灯しましたか？

補足

GPIO.output(17, GPIO.LOW) を実行しなくていいの？
⇒ GPIO.cleanup() を実行すれば、出力は初期化され、LEDは消えます。ブザーのプログラムでは、確実に音を止めるため、GPIO.output(_ , GPIO.LOW) を入れました。

7　ＬＥＤによるお知らせ

ＬＥＤを点灯させる

- 次に、2色のLEDを交互に点滅させてみましょう。
- 「led2.py」を開いてください。このまま実行すると、2色のLEDが同時に点灯し、3秒後に消灯します。
- これを、交互に数回点滅するよう、変えてみましょう。
- ここでは繰り返しを制御する for文 を使ってみましょう。

【プログラムの流れの制御】

- **for ○ in range(△)：**　変数○の値を0から△-1まで変えて△回繰り返す。

```
for i in range(5):
```

5回繰り返す。

この場合、変数iを0から4まで増やしながら、5回繰り返します。必ずしも、繰り返しの中でiを使う必要はありません。

繰り返しの範囲はインデントで指定

7　ＬＥＤによるお知らせ

ＬＥＤを点灯させる

LEDを点灯させるプログラムの箇所を次のように変えてみましょう。

例：

```
for i in range(5):
    GPIO.output(17, GPIO.HIGH)
    GPIO.output(18, GPIO.LOW)
    time.sleep(0.5)
    GPIO.output(17, GPIO.LOW)
    GPIO.output(18, GPIO.HIGH)
```

繰り返しの回数や、点滅の間隔も、好みで変えてみてください。

交互に点滅させたいので、一方をHIGHにしたら他方をLOWにする。

●完成したらプログラムを実行。

✓「参考プログラム」フォルダに、完成版として「led2_blink.py」というファイルが入っています。これも参考にしてください。

7　ＬＥＤによるお知らせ

LEDや電子ブザーの活用について

本研修では、ブザーと2色のLEDを用意しました。

LED2色の使い方として、例えば次のような工夫ができます。

- ◆赤ランプに付けたフォトセンサーがONになったらブザーを鳴らして、赤LEDを点灯させる。
- ◆黄色ランプに付けたフォトセンサーがONになったらブザーを鳴らして、黄色LEDを点灯させる。

このようにすると、ブザー音で気が付いた作業者は、
LEDを見ることで2種類の状態のどちらか判断ができます。

✓「参考プログラム」フォルダに入れてあるプログラム「photo_and_alarm.py」も参考にしてください。

参考

ブザー1個で2種類のお知らせ

LEDを2色用意することで、2種類のお知らせを判断できるという説明をしました。しかし、ブザー1個でも工夫次第で、例えば以下のように、異なるお知らせをすることができます。

- ◆赤ランプに付けたフォトセンサーがONになったら、ブザーが「プ,プー、プ,プー、プ,プー、 」と鳴る。
- ◆黄色ランプに付けたフォトセンサーがONになったら、ブザーが「プー、プー、プー」と鳴る。

プログラムを工夫することで、鳴らし方を変えられます。

✓「プ」と「プー」はブザーをONにしておく時間を変えることで実現できます。

✓興味のある方はチャレンジしてみてください。

✓「参考プログラム」フォルダに入れてあるプログラム「buzzer_2pattern.py」も参考にしてください。

8　グラフ表示

8-1　Node-RED

（注）「４．カウンター」で作成した配線とプログラムを利用し、そのグラフ表示を実施します。配線とプログラム（/home/pi/counter.py）が完成している前提での説明となっていますので、ご了承ください。

8-1　グラフ表示（Node-RED）

- センサーのデータは取れるようになったが・・・
  - 数字の羅列などで表示がわかりにくい
- 見やすくするために、グラフで表示したい。
- まずは「Node-RED」(ノードレッド)というしくみを使います。
  - プログラムの“コード” を入力するのではなく、グラフィカルな操作でプログラミングができる。
  - 「パレット」（**プログラムを乗せるボードのようなもの**）に、「ノード」（**プログラム上の部品のようなもの**）を乗せてつないでいく。
  - 様々な機能のノードが用意されており、適したノードがあれば、自分で作らなくてもいろいろな処理が可能

参考Webサイト
・https://nodered.jp/docs/user-guide/　「ユーザガイド：Node-RED日本ユーザ会」

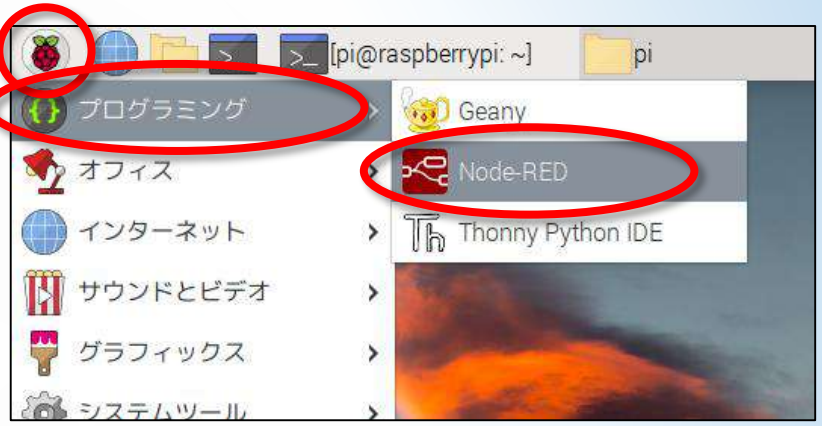
8-1　グラフ表示（Node-RED）

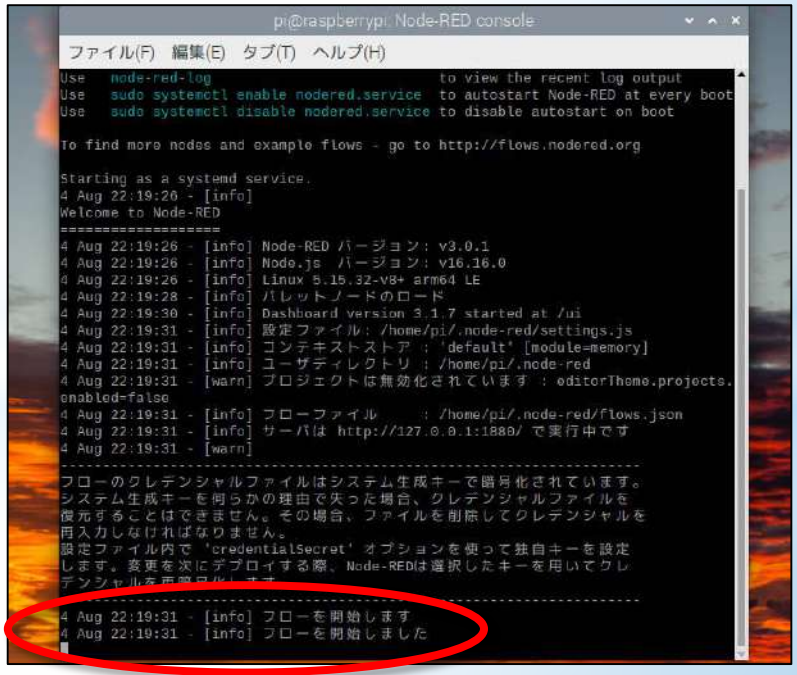
Node-REDの起動

Node-REDを使うためには、Node-REDを起動させる必要があります。

●Node-REDの起動

- ➢メニュー⇒プログラミング⇒Node-RED　の順にクリック
- ➢コンソール（黒いウィンドウ）に表示が出てくるので少し待つ。
- ➢「フローを開始しました」という表示が出たら、コンソールは閉じても構いません。





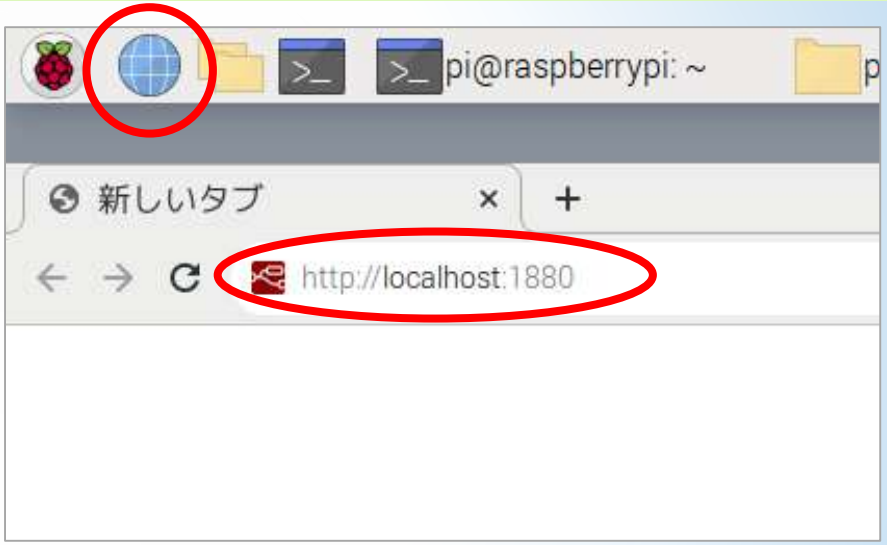
✓Node-REDは、Webブラウザを使ってシステムを構築、利用できる仕組みになっています。

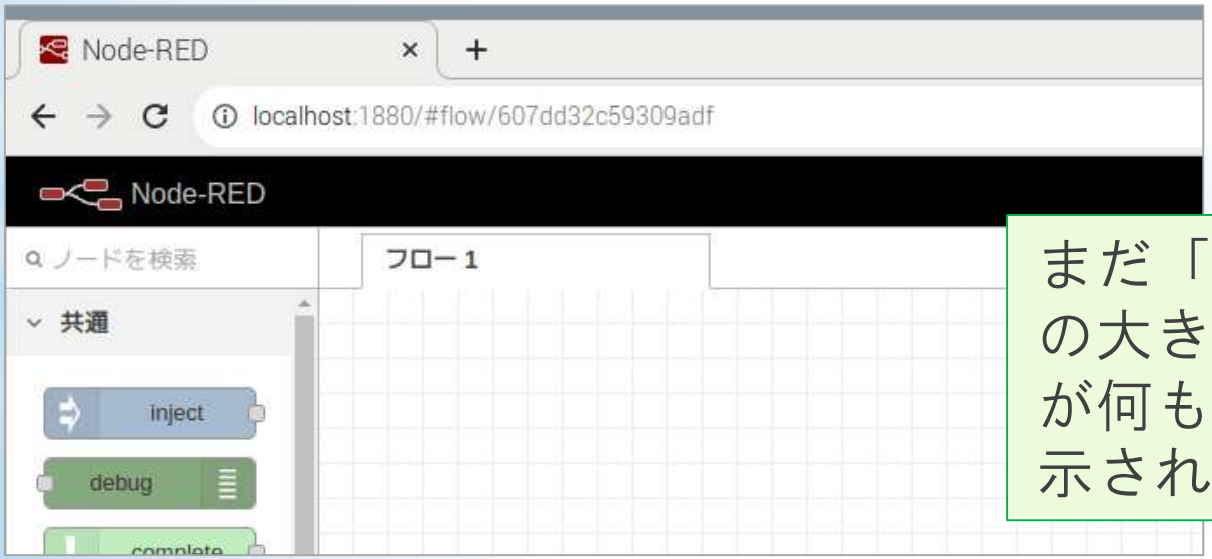
✓ここで実行した作業は、それを使うための準備です。

8-1　グラフ表示（Node-RED）

Node-REDの起動

- 続いて「ブラウザ」を起動する。（地球のマークのアイコン）
- アドレス欄に「http://localhost:1880」と入力してEnterキーを押す





まだ「パレット」（画面中央の大きな白い部分）にノードが何も乗っていない画面が表示されます。

8-1　グラフ表示（Node-RED）

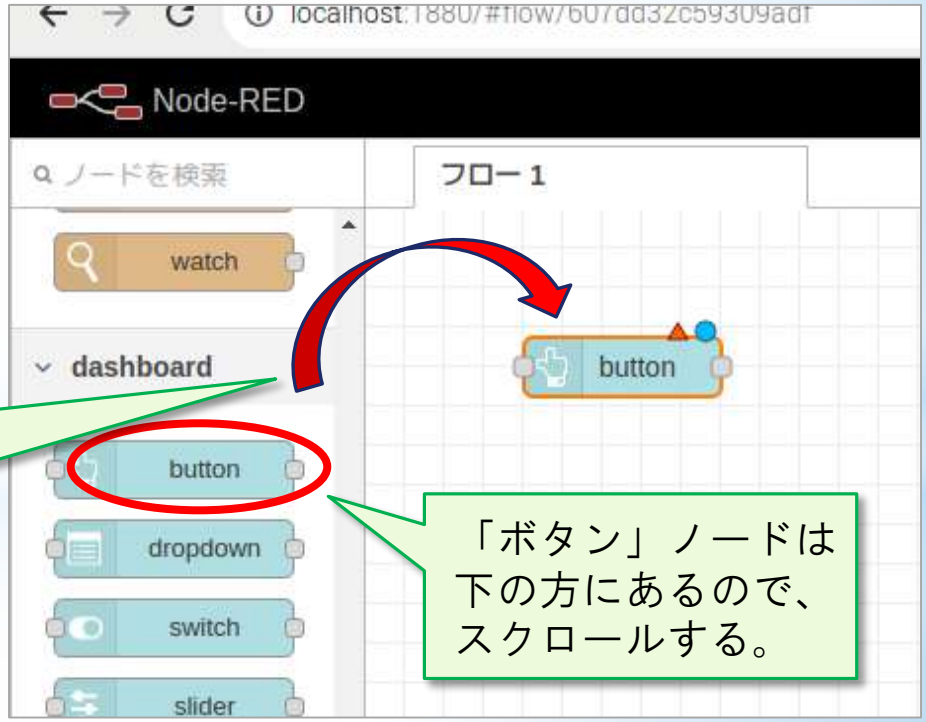
Node-REDによるグラフ表示

Node-REDで実現させたいこと

- 画面にボタンを配置し、ボタンを押したら、Pythonのプログラム（マグネットセンサーによるカウンター）が実行される。
- カウンターの実行結果（今の値）をグラフで表示する。

ノードの配置

- 左側に表示されているノードの中から使いたいノードをドラッグ＆ドロップでパレットに持ってくると、ノードを配置できる。



最初に「ボタン」ノードを配置しましょう。
（「button」と書かれたノードをドラッグ＆ドロップ）

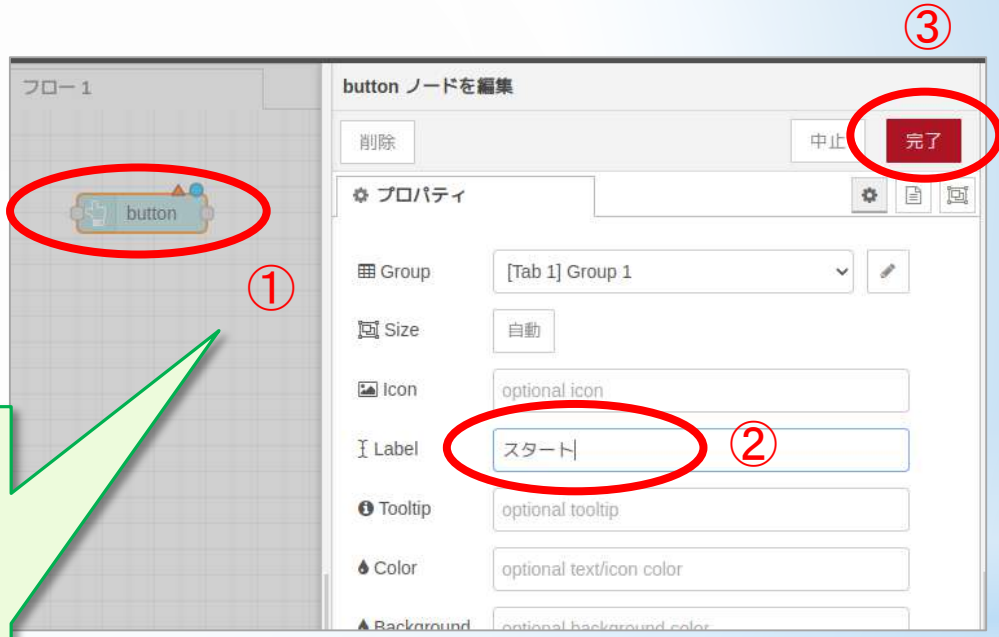
「ボタン」ノードは下の方にあるので、スクロールする。

8-1　グラフ表示（Node-RED）

Node-REDによるグラフ表示

ノードの設定

- それぞれのノードには様々な機能があり、機能に応じていろいろな設定ができる。
- ノードをダブルクリックすると設定画面が開く。



ボタンに表示する文字を設定しましょう。

① 配置した「ボタン」ノードをダブルクリックする。

② 設定画面が開くので、「Label」のところに「スタート」と入力する。

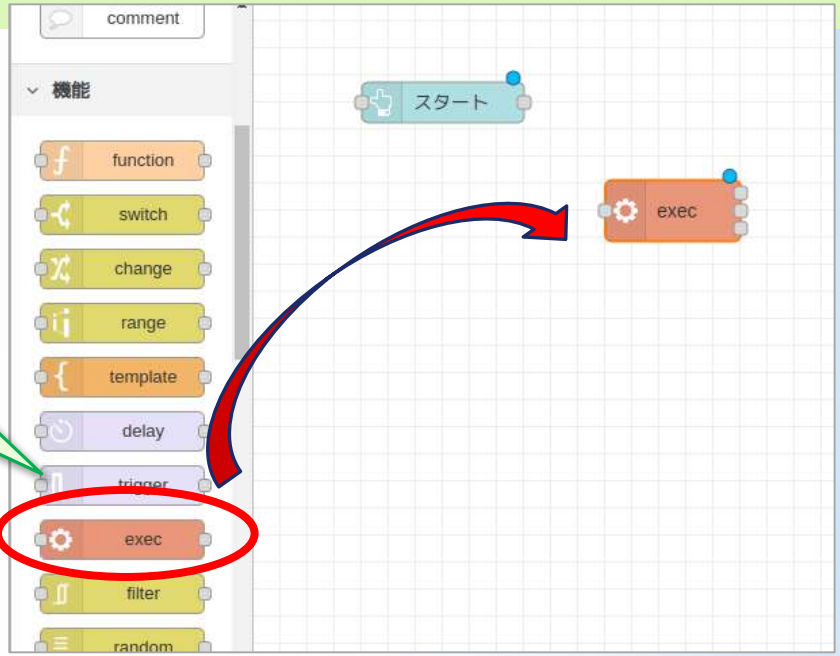
③ 右上の「完了」ボタンをクリックする。

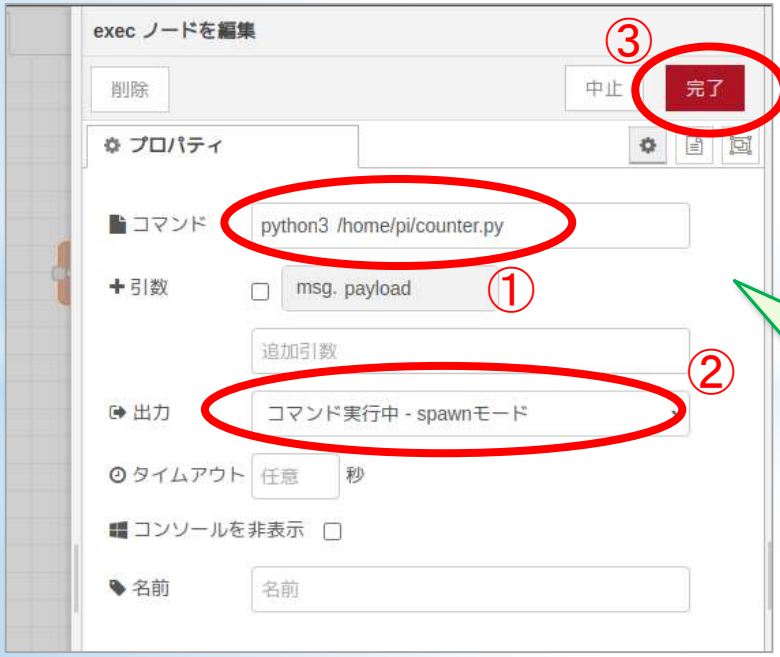
8-1　グラフ表示（Node-RED）

Node-REDによるグラフ表示

Pythonのプログラムを実行させる
ノードを配置し、設定をしましょう。

プログラムを実行させる
ノードは「exec」ノードで
す。これを配置しましょう。





(注)半角スペースが入る

配置したノードをダブルクリックして、設定をしましょう。
① コマンド欄に
「python3 /home/pi/counter.py」と入力
② 出力欄から「コマンド実行中-spawnモード」を選択
③ 「完了」ボタンをクリック

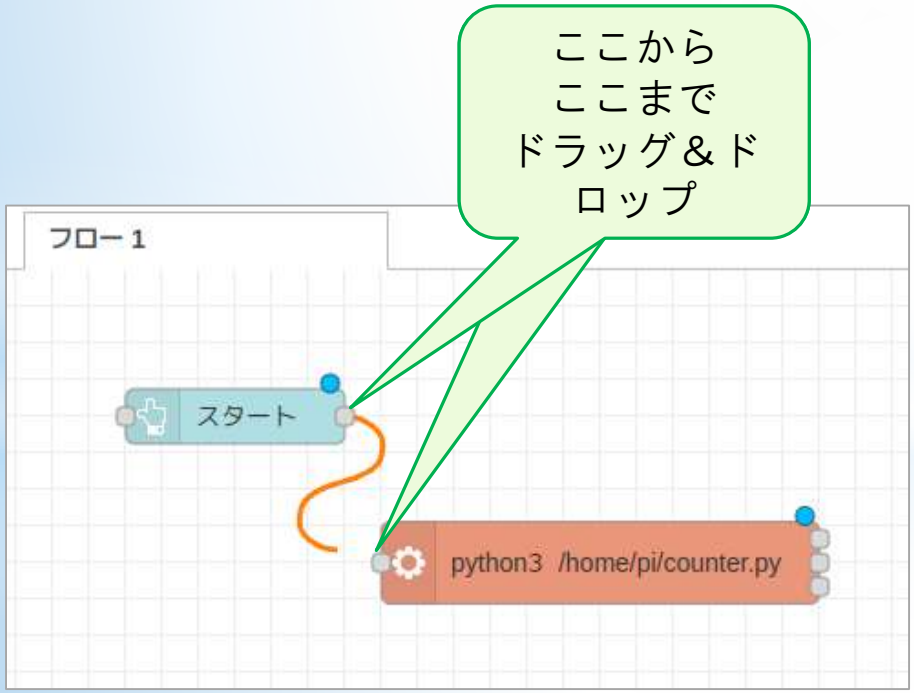
8-1　グラフ表示（Node-RED）

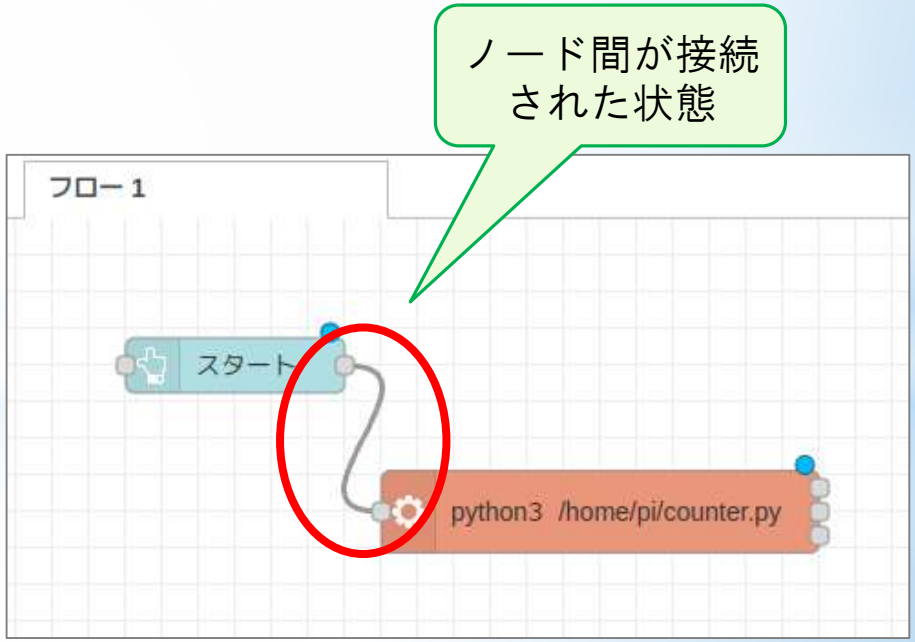
Node-REDによるグラフ表示

ボタンが押されたらプログラムが実行されるようにするには？

⇒ 「ボタン」ノードの出力を「exec」ノードの入力に、ドラッグ＆ドロップでつなぐ。

✓Node-REDは、ノード間をつないでいくことで処理やデータの流れを作っていく。



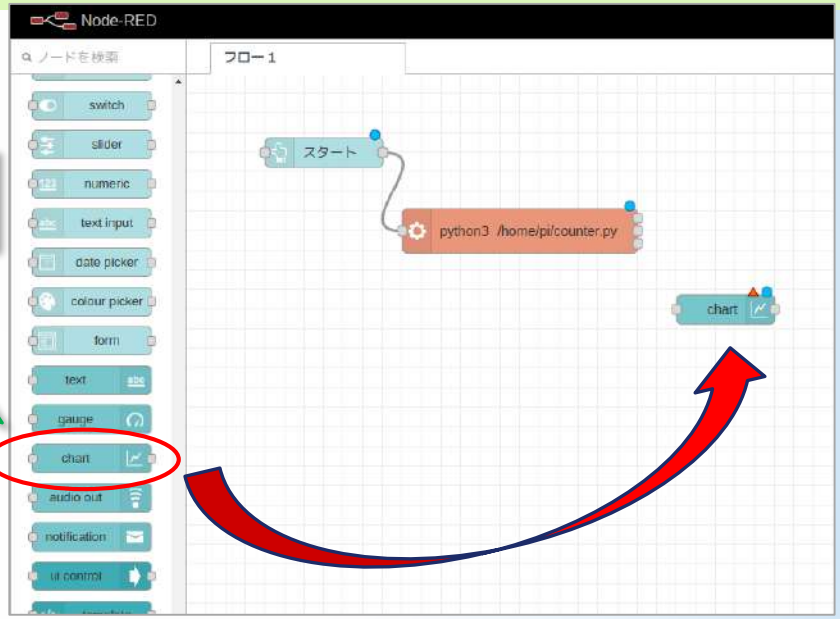


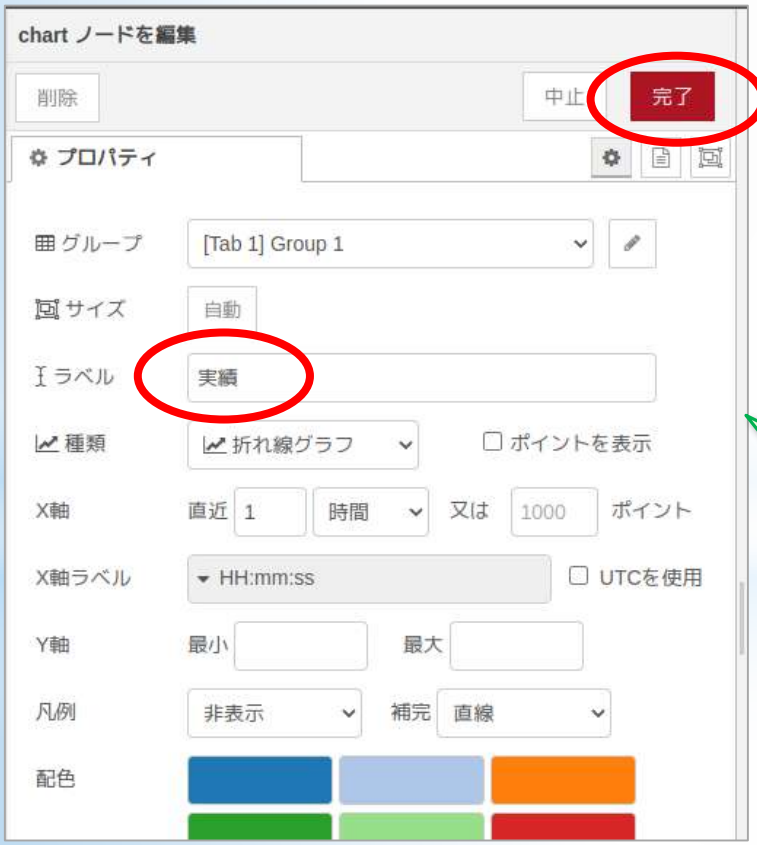
8-1　グラフ表示（Node-RED）

Node-REDによるグラフ表示

グラフを描くには「chart」ノードを使います。

「chart」ノードを配置





配置したノードをダブルクリックして、設定をしましょう。

- ラベル欄にグラフタイトルを入力します。ここでは「実績」としてみましょう。
- 入力したら、「完了」ボタンをクリック

✓ グラフの横軸・縦軸や配色、凡例などの設定もできます。ここでは特に変更しません。

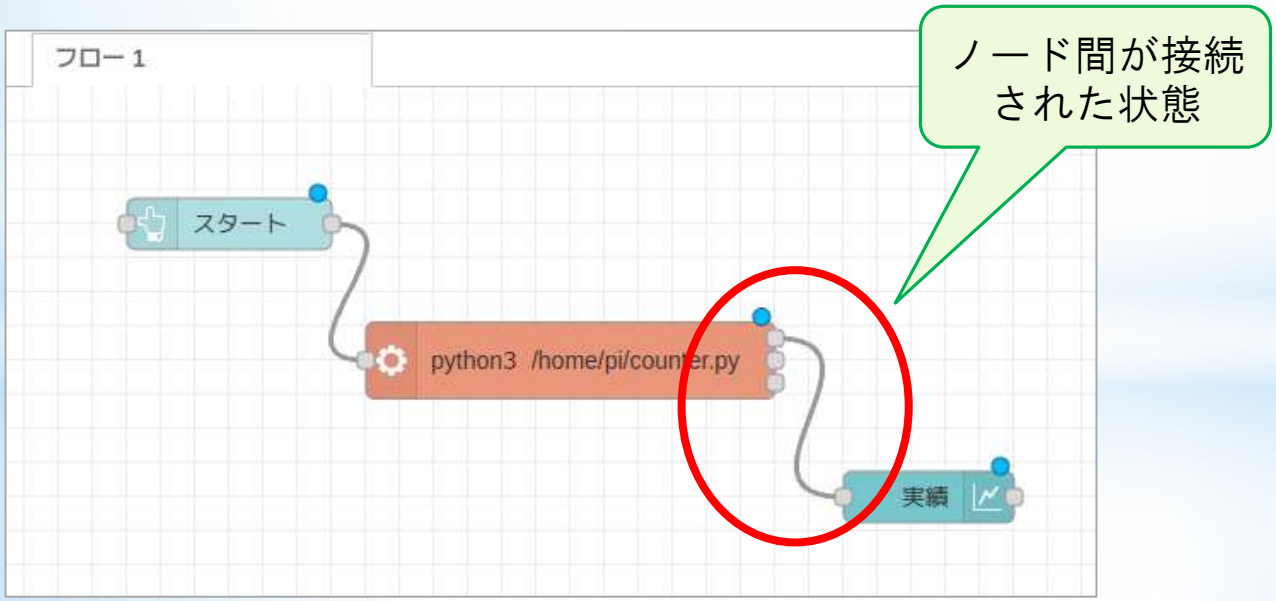
8-1　グラフ表示（Node-RED）

Node-REDによるグラフ表示

「chart」ノードの設定が終わったら、「exec」ノードの出力から「chart」ノードの入力までをつなぎましょう。

⇒ こうすることで、「exec」のノードの実行結果をグラフで表示できます。

✓「exec」の実行結果が数値だけで出力される場合は、そのまま「chart」に渡せる。



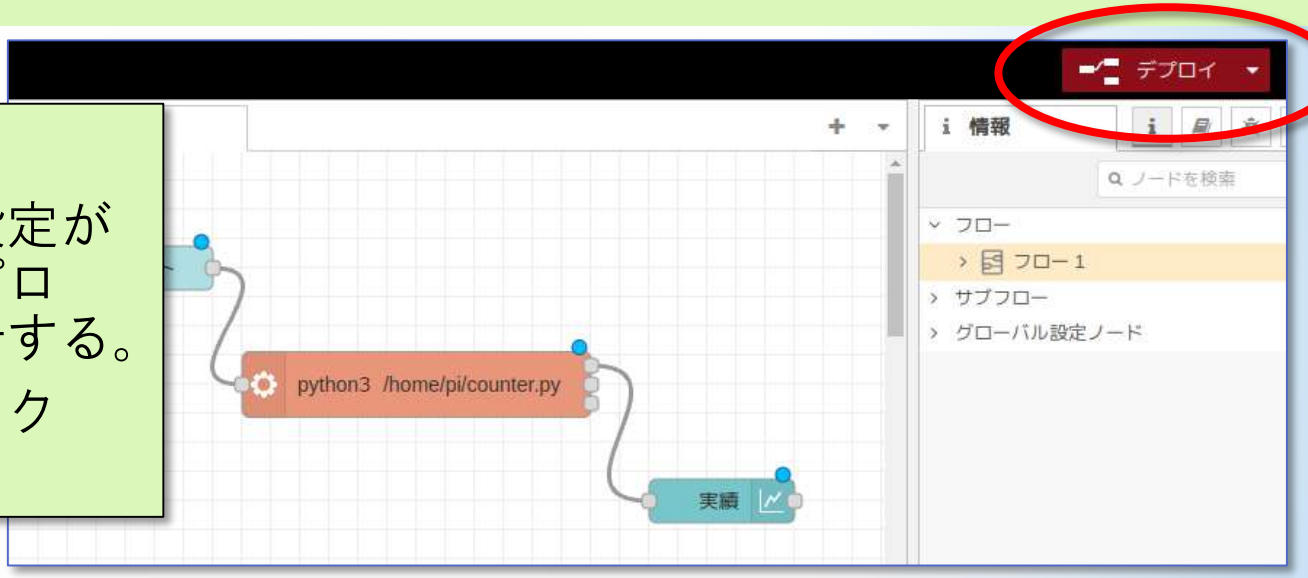
8-1　グラフ表示（Node-RED）

Node-REDによるグラフ表示

デプロイ

- ノードレッドでは、設定が全てできたら、「デプロイ」という処理を実行する。

➢右上の「デプロイ」をクリック



デプロイが成功すると、「デプロイが成功しました」という表示が出る。

✓この表示はしばらくすると自動的に消えます。

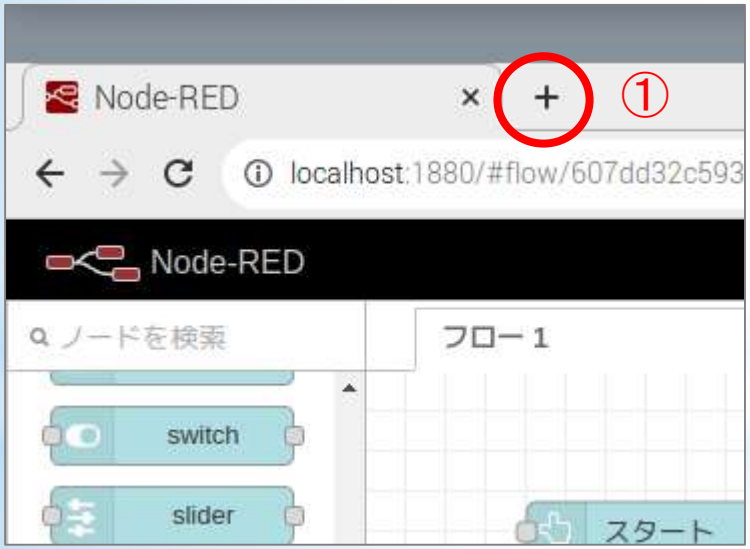
8-1　グラフ表示（Node-RED）

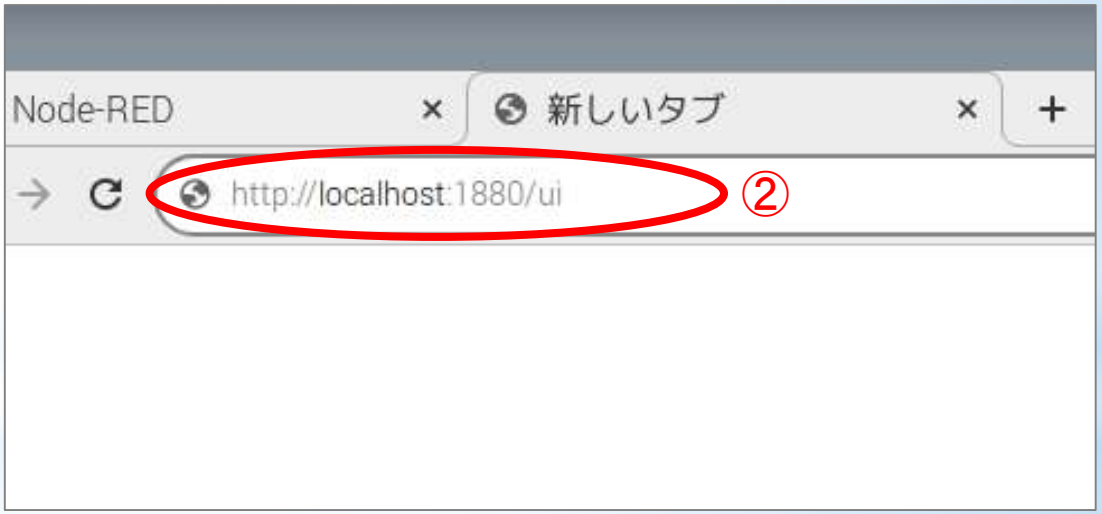
Node-REDによるグラフ表示

「デプロイ」が成功したら、グラフを見てみましょう

① ブラウザの「＋」ボタンを押して、新しいタブを追加

② アドレス欄に「http://localhost:1880/ui」と入力してEnterキーを押す





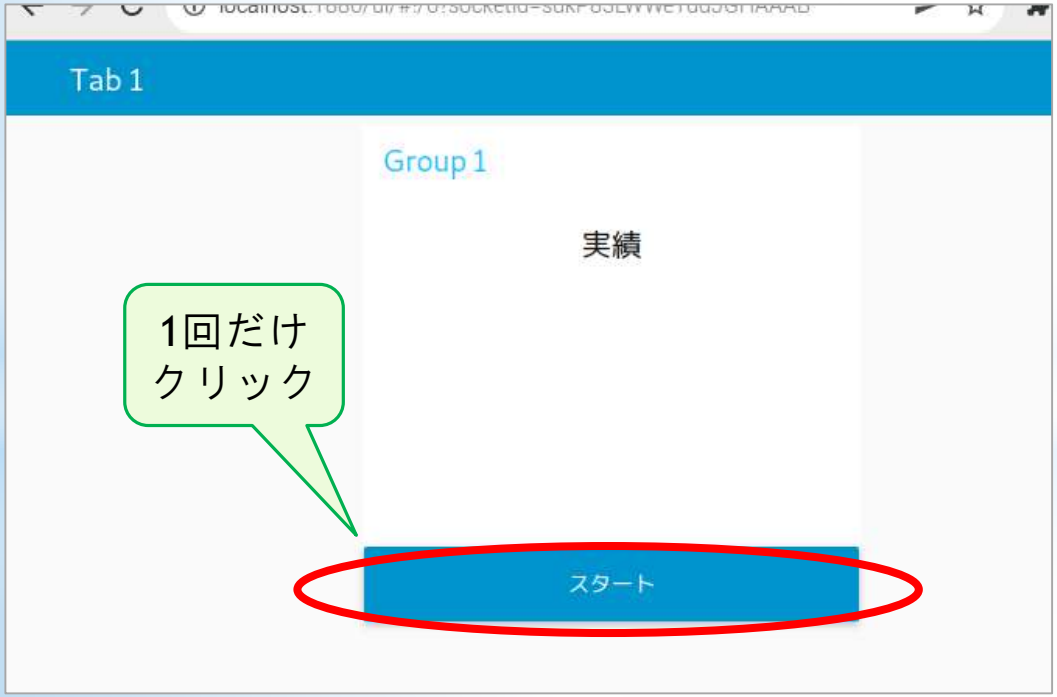
8-1　グラフ表示（Node-RED）

Node-REDによるグラフ表示

下の画面が表示されたら「スタート」ボタンをクリックします。

✓クリックは1回だけ。何度もクリックすると、Pythonが二重、三重に実行されてしまいます。

◆マグネットセンサーに磁石を近づけたり遠ざけたりして、グラフが描かれていくか確認してください。

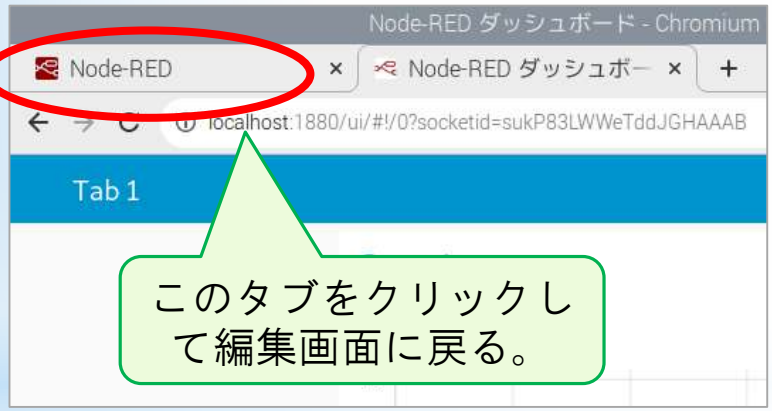


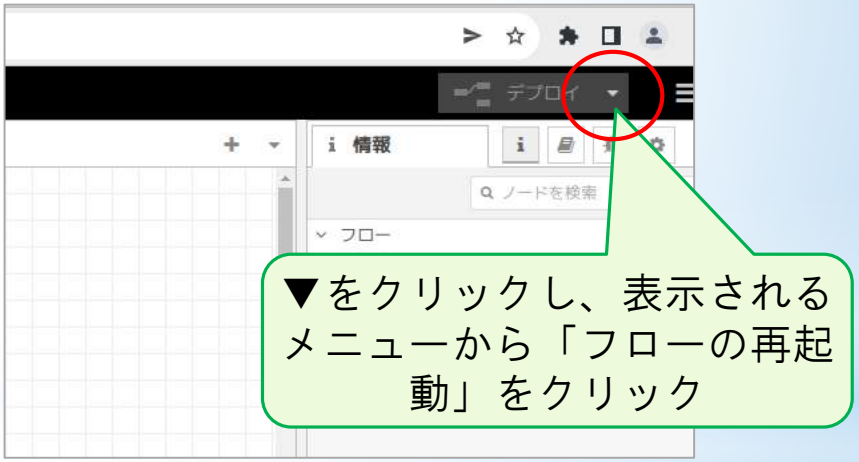
8-1　グラフ表示（Node-RED）

Node-REDによるグラフ表示

補足

◆今回やったグラフ表示では、カウンターのプログラムやグラフの描画を止める仕組みを作っていません。ブラウザのタブをクリックしてNode-REDの編集画面に戻り、「デプロイ」の『▼』ボタンから「フローの再起動」をクリックすると、動作していた処理が停止し、最初の状態に戻ります。





◆グラフに表示される時刻はラズパイの時刻です。

✓時刻については、次頁の参考情報もご覧ください。

ネットワークに接続する方法と時刻合わせ

- ラズパイをネットワークに接続する方法は、参考書籍のp81～をご覧ください。
- 社内等のLANで “DHCPサーバー” が稼働している場合、有線接続のLANであればすぐにつながります。
- 本研修でお渡しするラズパイは無線LAN機能も搭載しています。
- 無線LANの環境がある場合、p84～の手順で無線LANに接続できます。

✓ラズパイは、停止している間に自分自身で時刻を保持する仕組みがありません。停止中は、シャットダウンした時刻のまま止まっています。

✓ラズパイは、起動時などのタイミングでネットワークから時刻情報をもらって正しい時刻に修正します。ネットワーク接続できない場合、ラズパイは正しい時刻に合わせられず、シャットダウンしたときの時刻から進んでいきます。

8　グラフ表示
8-2　Pythonプログラム

（注）「４．カウンター」で作成した配線とプログラムを利用し、そのグラフ表示を実施します。配線とプログラムが完成している前提での説明となっていますので、ご了承ください。

8-2　グラフ表示（Pythonプログラム）

Pythonプログラム

- Node-REDによるグラフ表示をやりましたが、Pythonでもグラフ表示のプログラムを作ることができます。
- またThonnyを使って、「graph1.py」を開いてください。
- このプログラムはそのまま実行できます。まずは実行してみてください。

  ※ グラフが表示されるまで少し時間がかかります。

✓ 停止するときはThonnyの停止ボタンで止めます。停止してもグラフのウィンドウが残っている場合は、「×」でグラフのウィンドウも閉じてください。

```
#! /usr/bin/env /usr/bin/python3

import RPi.GPIO as GPIO
import time
import matplotlib.pyplot as plt

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

fig = plt.figure()
ax = fig.add_subplot(111)

x = [0,]
y = [0,]

i = 1
cnt = 0

try:
    while True:
        if(GPIO.input(21) == 1):
            cnt = cnt + 1
        i = i + 1

        plt.cla()
        plt.title('Jisseki')
        ax.set_xlim((0, 60))
        ax.set_ylim((0, 50))

        x.append(i)
        y.append(cnt)
        ax.plot(x, y)

        plt.pause(1)

except KeyboardInterrupt:
    plt.cla()
    plt.close()

    print('End')
```

8-2　グラフ表示（Pythonプログラム）

Pythonプログラム

- GPIOの設定や、ifによって入力を判断してカウンターの値を増やすのは、前にやった内容と同じです。
- グラフ描画のために新しく出てきた項目を中心に、順に説明します。

```
5  import matplotlib.pyplot as plt
```

「matplotlib」というグラフ描画に使用するライブラリを、「plt」という名前で使う準備

```
10 fig = plt.figure()
11 ax = fig.add_subplot(111)
```

グラフの描画領域を確保する

8-2　グラフ表示（Pythonプログラム）

Pythonプログラム

```
13 x = [0,]
14 y = [0,]
```

グラフにプロットするx,yの値を入れておく“配列”を用意し、最初の値として0を入れておく。

xとyの値の組をグラフに与えてプロットします。
折れ線グラフを描くには複数の(x,y)の組が必要であり、そのために複数の値を入れておける「配列」を使います。
このように書くと、配列を用意し、最初の値だけ入れた状態になります。

```
25          plt.cla()
```

一度、描画領域をクリアする。

```
26          plt.title('Jisseki')
27          ax.set_xlim((0, 60))
28          ax.set_ylim((0, 50))
```

グラフのタイトルと、横軸、縦軸の設定をする。

8-2　グラフ表示（Pythonプログラム）

Pythonプログラム

```
30              x.append(i)
31              y.append(cnt)
```

配列xに、次の値として i (の値)を追加する。
配列yに、次の値として cnt (の値)を追加する

ここの処理をするたびに、配列に入っている値の数が1つずつ増えていきます。

```
32              ax.plot(x, y)
```

グラフに描画するx,yの値の組として、配列xと配列yを与える。

```
34          plt.pause(1)
```

グラフを描画し、1秒待機する。

1秒待機したあと、while True:のところに戻り、同じ処理を繰り返します。

９　事例紹介

9　事例紹介

「いろいろな事例を知りたい」というご意見を聞く
⇒企業秘密等もあって、支援企業の例は現場を紹介できないことも多い

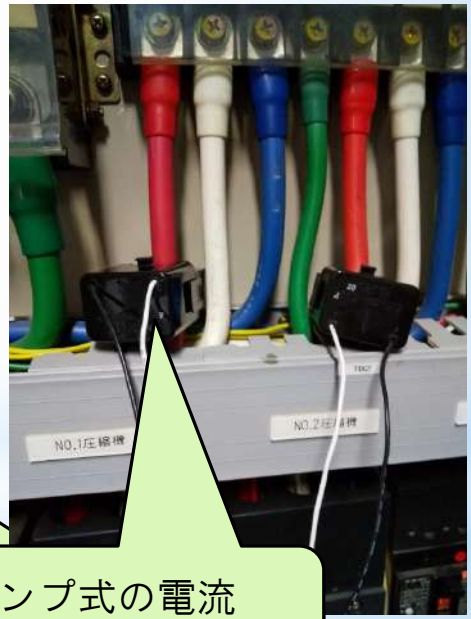
SAITECでの事例を紹介します。

9　事例紹介

SAITEC内の事例①

SAITEC内でもIoT化に取り組み

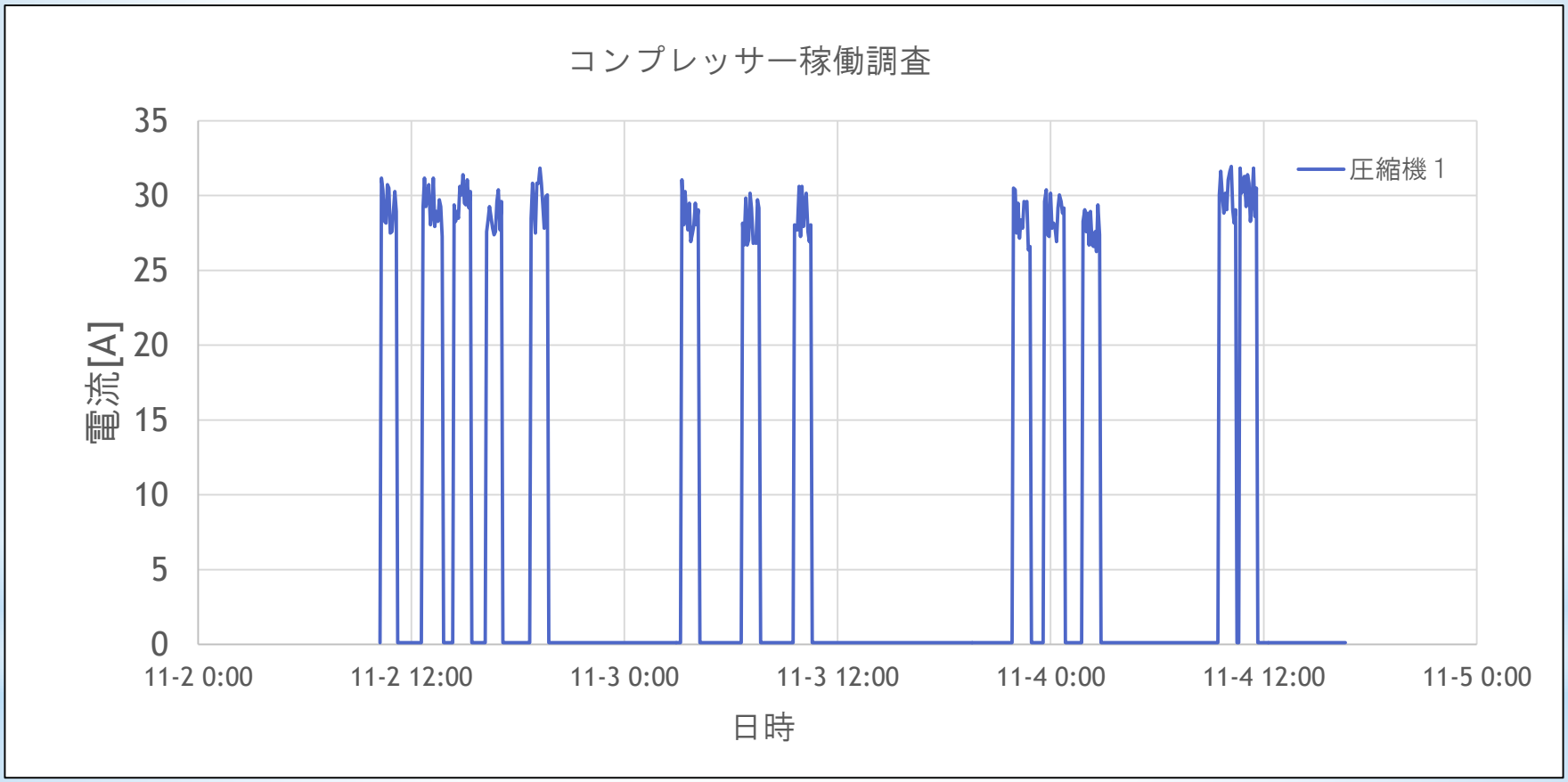
館内用圧縮空気のコンプレッサーの稼働状況調査



コンプレッサーの更新を見据えて、現在の稼働状況の調査をした。電流変化を調べることで、稼働状況を把握。

クランプ式の電流センサーを使用

SAITEC内の事例①



コンプレッサーの稼働状況が分かった

SAITEC内の事例②

塩水噴霧試験機の稼働監視

トラブル例

試験中に機器が故障し、JIS規格通りの試験が行えていない状態で試験を続けていた

▶ **貴重なサンプルの損失、無駄な試験時間**

対策

機器の異常に早急に気付くことができていれば適切な対処ができる。
機器があるのは地下１階であり、頻繁な確認は困難。

↓

機器の稼働状況を６階の事務室から遠隔監視できる
システムを構築し、トラブルを未然に防ぐ

SAITEC内の事例②

噴霧流量・噴霧圧力の監視

機器の問題点

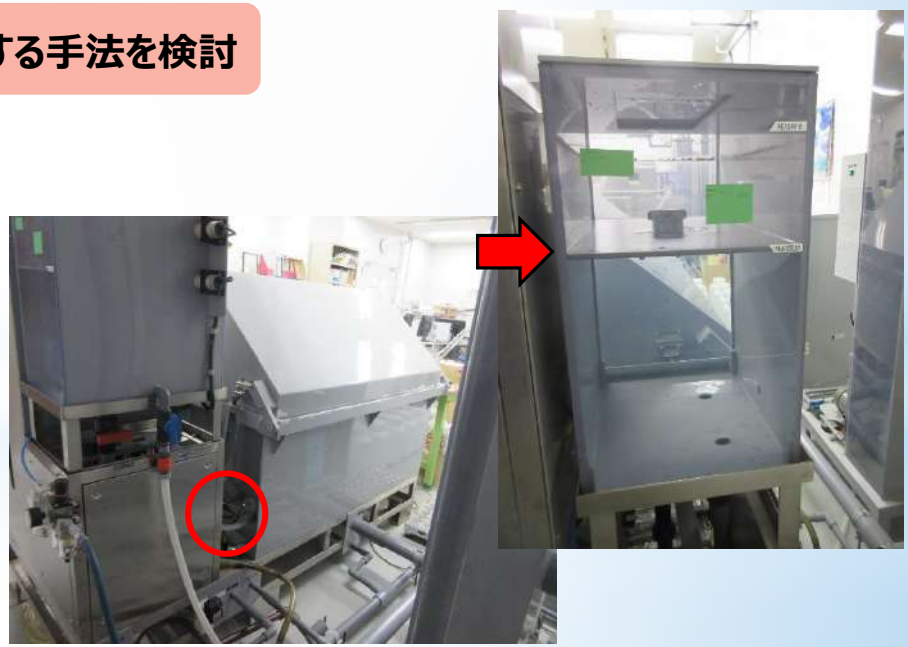
●試験液は、常時規定量が噴霧されている必要があるが流量が少ない。
流量の異常を適切なタイミングで検知できるか？

流量、噴霧圧力を常時監視する手法を検討

検討内容

●試験液が通過するホースに流量計を設置する
●距離センサを用いてタンクの蓋から液面までの距離を測定する
●タンクの液面を撮影→低下していく様子を画像認識

●噴霧圧力計を撮影→指針を画像認識



9　事例紹介

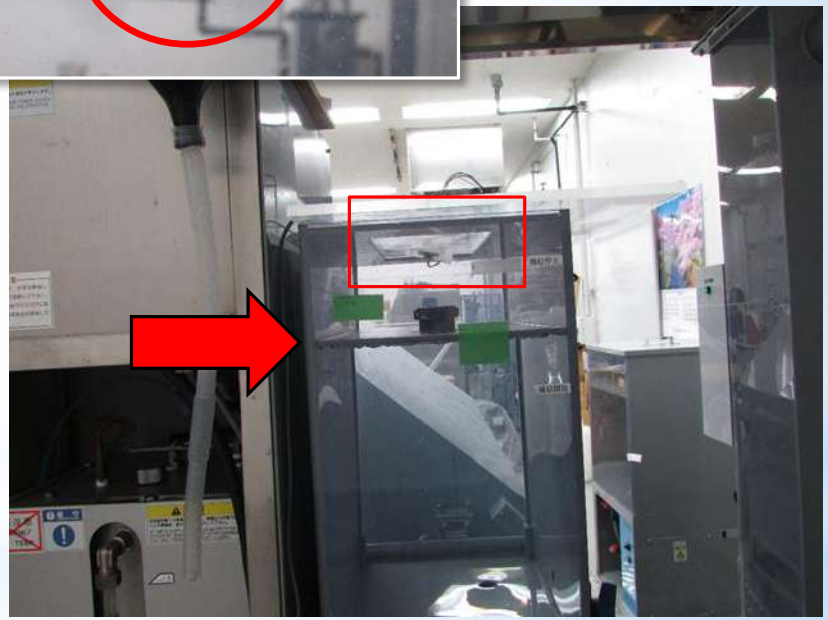
SAITEC内の事例②

噴霧圧力の監視

圧力のメーターをカメラで撮像し、画像認識で読み取る。

噴霧流量の監視

距離センサーで液面までの距離を計測



9　事例紹介

SAITEC内の事例②

監視画面

６階事務室で、試験機の状況が分かる。

最後に

- 本研修は、オンデマンド研修となっています。オンデマンドの利点を生かし、繰り返し見ていただき、理解を深めていただければ幸いです。
- お配りした機材は、自社の実際の現場で、自社の機器・設備に対して使っていただき、IoT化のノウハウ・効果を体感してほしいと考えています。
- センサーからのデータ取得、出力を使った通知機能、グラフ表示などを取り上げました。
- もっといろいろとやってみたい、という方は、配布した参考書籍もありますので、ぜひチャレンジしてください。また、高度な活用方法を目指す企業様には、SAITECが支援いたしますので、ご相談ください。

参考文献・参考Webサイト

参考書籍

・これ1冊でできる！ラズベリー・パイ超入門 改訂第7版
福田和宏 著 2022年3月31日 初版 第1刷発行

参考Webサイト

・ https://nodered.jp/docs/user-guide/ 「ユーザガイド：Node-RED日本ユーザ会」