プログラミングPBL

「Raspberry Piでデータロガーを作ろう！」

～第1回～

マイコン「Raspberry Pi」を使って
データを記録する装置（データロガー）を作ってみましょう！
プログラミングの基礎を学びながら、試行錯誤していこう！

1．RaspberryPi ってなんだ！？

Raspberry Pi（ラズパイ）は、もともとプログラミング教育用に開発された安価なコンピュータです。
しかし、その使い勝手の良さから多くのエンジニアに愛用され、様々な電子工作に使われています。
Raspberry Pi は「小さなパソコン」だと思ってください。

2．RaspberryPi を起動しよう！

まずは先生の注意を良く聞いて、RaspberryPi を起動させよう。**すべての機器を接続して、最後に電源を入れること！**
電源を入れたまま各種ケーブルの抜き差しをしてはいけません。以下の手順で接続してみよう。

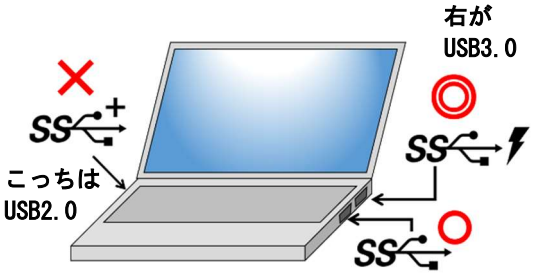
❶セットを確認しよう。

❷裏に回ってディスプレイとつなげ！

❸ディスプレイを切り替える。

❹周辺機器をすべて
ラズパイに取り付ける。

❺電源用 USB ケーブルをパソコンの USB3.0
コネクタに繋ぐ。



USB を繋ぐだけでラズパイは起動します。
必ず最後に接続すること！

きちんと接続できていれば、画面に文字がたくさん表示されてラズパイが起動する。

３．RaspberryPi にログインしよう！

ラズパイが起動したら、ログイン ID とパスワードを聞かれます。これは共通です。太字の部分を入力してください。

Raspberrypi login:**ナイショ**
Password:**ナイショ**　←注意！パスワードは直接表示されないので、見えない状態で入力します。
pi@raspberrypi:~ $　←このようなマークが表示されたら準備完了！

今回使用する「Raspberry Pi」に入っている OS（オペレーションシステム）は、Windows のような GUI（Graphical User Interface）ではなく CUI（Character-based User Interface）なのだ！…なんだそりゃ？
Windows のようにマウスでアイコンをクリックして操作するのではなく、**全てのやりとりを文字だけで行う方式が CUI です。**大学で研究を始めると、CUI のコンピュータを使う機会は多い。頑張って慣れよう！
ちなみに、ラズパイに入っている OS は Linux（リナックス）という種類の OS が入っています。
下記のようなコマンド入力で操作できます。

Linux の基本的なコマンド

すべて文字入力で操作します。以下の「$」より後ろの文字を入力しよう。

●今いる場所（ディレクトリ）を確認する。
`$pwd`

●今いる場所（ディレクトリ）にあるファイルを確認する。
`$ls`

●ディレクトリを移動する
`$cd`

（例）例えば、今回用意されている「Pimoroni」というディレクトリに行きたいとき
`$cd Pimoroni/`

●一つ上のディレクトリに移動する
`$cd ../`

●ディレクトリの作成
`$mkdir ディレクトリ名`

（例）例えば、「Science」というディレクトリを作りたかったら
`$mkdir Science`

４．テキストエディタ　nano を使ってみよう

マイコンへの命令（プログラム）は、”文書作成ソフトみたいなもの”で編集します。
今回は、「nano エディタ」で編集します。
早速使ってみよう。基本的には文書作成ソフトと同じ感覚で、改行などが出来ます。でも、今回は日本語を打てないよ。
ファイル名「sample.txt」を作って編集してみよう。下記のコマンドを打ってみて。

`$nano sample.txt`

では、色々入力して試してみよう！

nano エディタの基本的なコマンド

Ctrl + O …ファイルを保存する
Ctrl + K …カーソルがある行をカットする
Ctrl + U …カットした行をペーストする
Ctrl + X …終了する

５．プログラム言語「python3」を使ってみよう

ラズパイに命令を送るために、プログラム言語「python（パイソン)」を使います。
まぁ、とにかく使ってみよう！まずは下記のコマンドを打ってみて。

```
$python3
>>>
```

左に表示されていた pi@raspberry:~$ が >>>（プロンプト）に切り替わった！python3 が起動した証です。

例えばこれは簡単な電卓としても使えるぞ！
さっそく以下の計算を入力してみよう。Python が計算してくれるぞ。

```
>>>1+3
```

とか

```
>>>5*20/4
```

を打ってみると…どうだったかな？　他にも、色々な計算を試してみよう。

また、なんと**文字の足し算、かけ算もできる**。その場合、文字を’’で囲む必要がある。
試しにやってみよう。

```
>>>’I love you’*10
```

ではここで、python に**ちょっと複雑な命令**を出してみよう。
場合分けの命令コマンド「if」を使う。

```
>>>if 条件式1:
       条件式1がその通り（True）のときに行う命令文
>>>elif 条件式2:
       条件式1が違って（False）条件式2がその通り（True）のときに行う命令文
>>>elif 条件式3:
       条件式1，2が違って（False）条件式3がその通り（True）のときに行う命令文
>>>else:
       すべての条件式が違う（False）ときに行う命令文
```

上記の説明にそって入力していこう。

```
>>>a  =  10                ←「a に 10 を代入します。」
>>>if   a  >=  100:        ←「もし、a が 100 以上ならば…」
         a  *=  2          ←「a に 2 をかけなさい。」
   elif a  >=  50:         ←「そうではなくて、a が 50 以上ならば…」
         a /=  2           ←「a を 2 で割りなさい。」
   else:                   ←「どの条件にも当てはまらないなら…」
         a  += 2           ←「a に 2 を足しなさい。」
         print(a)          ←「それでは、結果を表示(print)しなさい。」
```

さて、なんと表示されたかな?
ちなみにコマンド中の「：」は複合文といって
「まだ命令は終わってないですよ！続きがあります！」という意味。

次は、**繰り返し処理**をしてみよう。

```
>>>while  条件式：
        繰り返したい命令文
```

とりあえず、以下を 1 行ずつ入力していこう。

```
>>>a = 1
>>>while a <= 10:
        print(a)
        a + = 1
```

さぁ、なんと表示された？
a が 1 からスタートして、1 ずつ足していく命令を繰り返し、10 以下の数字まで全て表示する･･･という意味だ。
if と while を覚えたので、次に行きましょう。

Python3 を終了するときは、「Ctrl + D」**です。**

６．nano でスクリプト（プログラムが書かれたファイル）を作ろう！

Python に色んな命令ができるのはわかりましたが、**一行ずつ命令を送るのは不便ですね。**
毎回命令を出していては、変数の指定もできません。
つまり、先ほどの if や while も、a の値を変更するたびに同じ命令文を入力しないといけない。

そこで、nano で 1 行ずつのプログラムを書いたファイルを作り、それを読み込ませよう。
これが「スクリプト」だ。
先ほどの if 文の命令を、nano エディターで作ってみよう。

```
$nano sample.py      ←拡張子を「.py」にすること！
```

nano が起動したら、以下のプログラムを入力しよう。

```
a  =  10
if   a  >=  100:
     a  *=  2
elif a  >=  50:
     a /=  2
else:
     a  +=   2
     print(a)
```

なお、行頭は上記のように合わせましょう。これを「インデント」と言います。
書けたら、Ctrl + O で保存、Ctrl+X で終了。
では、実行してみよう。

```
$python3 sample.py    ←   「python3 で sample.py を実行する」というコマンド
```

さぁ！何が表示されたかな？
自分で数字を変えて、色々試してみよう。

これが、python を使ったスクリプトの概要だ。
Nano と python の使い方や関係が少しはつかめたかな？

５．おみくじスクリプトを作ってみよう！！

では早速スクリプトを作っていくよ！習うより慣れろだ！**知らない関数もあるけどガンガンいこう！**
まずは、nano エディタで次のスクリプトを組んでみよう。
なんと、おみくじが出来るプログラムだぞ！

```
$nano omikuji.py

import random

def number_input(message):
    result = input(message)
    if(result.isdigit()):
        result = int(result)
        return result
    else:
        return 0

def judgment(omikuji):
    rand = random.randint(1, 9)
    if(omikuji == rand):
        print(”You are very lucky!”)
    elif(omikuji > rand):
        print(”Well... not bad.”)
    elif(omikuji < rand):
        print(”Oh my GOD!You are Unlucky!”)

while  True  :
    omikuji = number_input(”Please choose the number you like among 1 to 9 ＼n＼n”)

    if((omikuji != 0) and (omikuji > 0) and (omikuji < 10)):

        print(”You like ” + str(omikuji) + ”  ＯＫ？＼n＼n”)
        print(”Then I will take a fortune.＼n＼n”)

        judgment(omikuji)

        break

     else:
        print(”Choose 1 to 9!!!＼n＼n”)
```

できあがったら、保存して実行してみよう！

```
$python3 omikuji.py
```

上手く動かないときは、インデントやスペースがきちんと入っていない、文字が間違っている･･･等が考えられます。
英語でエラーが表示されているので、該当する行に入力ミスがないか丁寧に調べよう。
大体エラーの正体は入力ミスです。とにかく頑張れ！

おみくじスクリプトの説明（頑張って自分が入力したスクリプトを見ながら読んでみよう！）

このスクリプトは実行後、ユーザーに１～９の数値を入力させてきます。
その数値が、スクリプト内でランダムに設定された数値とピタリ会えば"You are very lucky!"
と表示されます。そうでない時は、いろんなコメントが表示されます。
１～９以外を入力すると怒られます。そんなスクリプトになっています。
どんな仕組みなのでしょうか。
詳しく見ていきましょう。

まず最初に２つの関数を def で定義しています。
関数とはインプットされた値をもとに、決められた処理を実行してくれるプログラムの部品です。
def は自分で関数を作ることができるコマンドです。

と言うわけで、１つ目の関数は number_input(　)、２つめの関数は judgment(　)です。
（関数の名称は英数字で作らなければなりません。）

number_input(　)関数は、入力してもらった数値が「文字列」なのか「数値」なのか判別する関数です。
（Python に元から設定されている関数を組み合わせて、新しい関数を創り上げています。）
パソコン上では、数字は「文字列」か「数値」のどちらかに該当します。
同じ「１００」でも「１００という文字であって、計算できないモノ」か「100 という数値なので計算に使えるモノ」に分かれるわけです。（基本的に、文字列には“　”がついています。）

input(　)関数の**戻り値**は必ず文字列型です。**戻り値**とは、関数を実行すると出てくる数値のことです。
例え１００と入力しても “100”という文字列型の数値が返ってくるのです。
isdigit(　)関数はとっても有能で、文字列型の“100”でも、それが数字の形をしていれば、数値であると判断してくれるのです。
int 関数は、「文字列」の数字を「数値」に変換してくれます。int(“100”)とすると、その戻り値は純粋な整数としての 100 になります。
でも int(“apple”)とするとエラーが起きてプログラムが止まってしまいます。
そこで事前に、isdigit(　)で数値かどうかチェックし、数値なら int(　)で整数値に変換して返します。
もし数値でないなら、エラーとして０を返しています。

お次は judgment(　)関数です。この関数は単純明快です。
ただ単に、1 から９の乱数を発生させて、それがユーザーが入力した値と同じだった場合、ユーザーが入力した値よりも大きかった場合、ユーザーが入力した場合よりも小さかった場合に分けて、占い結果のメッセージを画面に表示するだけですね。

で、何回ユーザーが入力ミスするかは不明なので、あえて意図的に while(True)で無限ループを作ります。
そして変数 omikuji に、ユーザーから入力された数値を number_input（　）関数を利用して、受け取ります。

number_input(　)関数はエラーならば０を返しますから、まず０以外でないといけません。
またユーザーに 1 から９の数値の入力を求めているので、omikuji は０より大きく、10 より小さくなければなりません。これら３つの条件を and でつないでいるだけです。

そしてこの３つの条件を全てクリアできたら、占いをはじめる旨のメッセージを表示し、さらに judgment(　)関数に変数 omikuji を渡して、最終的な占い結果を画面に表示します。

ここまでやったらプログラムの目的は果たしたので、break で while ループを抜け、プログラムを終了します。
逆に、omikuji の値が、1～9 のいずれにも該当しない場合は else 以下でエラーメッセージを表示し、もう１回、while ループをやり直す形になります。

・・・ここでは、なんとなくわかれば大丈夫です。慣れるために、かなり難しい事をやりました。

６．いよいよセンサリングをしてみよう！！

では、いよいよ**本日の目玉！**センサリングだ！
RaspberryPi に取り付けている EnviroPHAT というセンサーを使ってみるぞ。
このセンサーを動かすのに必要なライブラリは、すでに君たちのラズパイに先生がコピーしている。
まずは、カレントディレクトリ（ホームみたいなもの）に移動して、センサリング用のスクリプトが入った場所に行ってみよう。

```
$cd
$cd Pimoroni/envirophat/examples
```

ls で中身を見てみよう。

```
$ls
```

なにやら、沢山のスクリプトがあるね。
試しに、all.py を走らせてみよう。

```
$python3 all.py
```

すげぇ！いろんなデータが見られる！！
でも、これ･･･見られるだけで、**保存や記録はされてないんだ。**なので、**保存ができるスクリプトを書いてみよう！！**
example を壊したら大変なので、カレントディレクトリに戻ろう。

```
$cd
```

さぁ、スクリプトを作っていこう！！

【MISSION！】温度、照度、3 軸の加速度を 1 秒ごとに計測して、それを csv ファイルに保存するスクリプトを作る！

```
$nano mysensor.py
```
（←ファイル名は何でもいいけど、とりあえずマイセンサーで。）

```
import sys, time
from envirophat import light, weather, motion, analog

INTERVAL = 1 #sec

def write(line):
    print(line)
    fn = “envirophat-%s.csv” % (time.strftime( “%y%m%d” ))
    with open (fn, mode = ‘a’ ) as f:
        f.write(line+” ＼n” )

try:
    while True:
      vs = [round(weather.temperature(),2),light.light()]
      vs += [round(x,2) for x in motion.accelerometer()]
      write( “,” .join(map(str,vs)))

      time.sleep(INTERVAL)


except keyboardInterrupt:
     pass
```

保存して，実行してみよう！
1 秒ごとのデータが出力されるぞ！数値の変化を観察してみよう。
→ 記録する内容を変更することは出来ないか？（加速度→照度の RGB に出来そうだ。）
次回は、さらに工夫していろいろなセンサリングを行う予定だ！

今ではネット上（特に YouTube）にラズパイなどの説明動画や実践例がたくさん存在します。
電子工作 YouTuber もたくさんいて、とても参考になります。
探究活動のヒントになるので、ちょっとプログラミングが好きになった人は、いろいろ見てみるのも良いでしょう。自己流でどんどんやってください。