プログラミングPBL

「Raspberry Piでデータロガーを作ろう！」

～第2回～

今回は Python の基礎をより詳しく学んだ後に
いろいろな「自分だけのデータロガー」を作ってみよう！

RaspberryPi 起動の復習…すべての機器を接続して、最後に電源ケーブルをつなぐこと！

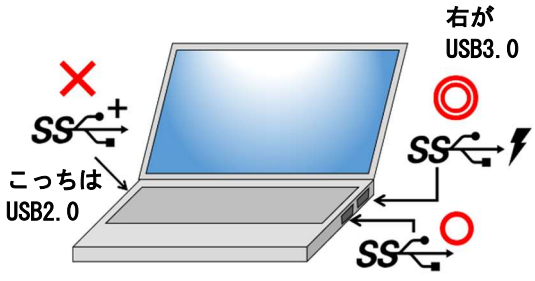
❶セットを確認しよう。

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、ラズパイが起動するので、下記の ID とパスワードを入れよう。

==========

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！

==========

１．スクリプトを書きまくった今だからこそ、各項目を詳しく学ぼう！

前回は、ただスクリプトを写してもらう作業をしました。それだけでも、ラズパイがどんなモノか感覚的に理解できたと思います。今回は、前回書いてもらったスクリプトの詳しい部分を学んだ上で、いよいよマニュファクチャリングに挑戦！自分で考えて、オリジナルのデータロガーを作ってもらいます！沢山悩んで、試行錯誤してくれ！

（１）基本的なLinuxのコマンド（新しいコマンドも紹介！）

①ディレクトリの中にどんなファイルやフォルダがあるか見る

```
$ ls
```

②ディレクトリを移動する。

```
$ cd （移動先）
```

※cdだけを実行すると、ホームに戻ってくる。便利なので使おう。 ../で１つ上に行く。

③ファイルのコピー

```
$ cp （コピーしたいファイル） （コピーしたいディレクトリ）
```

（例）$ cp main.py ../

main.pyというファイルを、一つ上のディレクトリにコピーする。

ちなみに「../」は一つ上のディレクトリ、「./」は自分がいるディレクトリを表します。

④ファイルの移動

```
$ mv （移動したいファイル） （移動先のディレクトリ）
```

⑤自分がいるディレクトリの確認

```
$ pwd
```

⑥時刻の確認

```
$ date
```

※ネットに繋いでないと古い時刻が出ます。

⑦ディレクトリを作る

```
$ mkdir （ディレクトリ名）
```

⑧ファイルの中身の最後の方を確認する

```
$ tail （確認したいcsvファイルなど）
```

⑨ファイルを編集する（nanoを使う）

```
$ nano （ファイル名）
```

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

⑩python3でプログラムを走らせる

```
$ python3 （～.pyというファイル）
```

プログラムの停止は

Ctrl + C

⑪ラズパイの再起動

```
$ reboot
```

⑫ラズパイの終了

```
$ sudo shutdown -h now
```

（２）EnviroPHAT をつかったセンサーのスクリプト

それでは、データロガーを作るためのスクリプトで一体何をしていたのか勉強していこう。
なお、プログラムは基本的に

```
◇◇ ＝ ●●●●●●●●●●
○○ ＝ ■■■■■■■■■■■■■■
△△ ＝ ○○ ＊ ◇◇
```

という感じで、前の行で後で使う情報を定義していきます。ここに関数などが入ってくる。
意外に英語でそのまんま文章を書けばプログラムになることも多いのです。
なお、# マークはコメントアウトといいます。# 以降の文字は読み込みません。
メモにも使えるし、ちょっと別のコマンドを試したいときに、わざわざ行を消さずに# を付けておけば、復帰もラクですね。
↓こんな感じで使うんじゃ。

```
◇◇ ＝ ●●●●●●●●●●    #This is Days
○○ ＝ ■■■■■■■■■■■■■■ #This is Times
#□□ ＝ ○○ ＋ ◇◇    ←この行は無視されるようになるよ。
△△ ＝ ○○ ＊ ◇◇
```

さて、前回間に合わなかった人は、ここから完成させよう！

```
$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
```

２．それぞれの説明＋α（何をやっていたのか！？＆他に何が出来るのか？）

では、センサーのプログラムを詳しく見ていきながら練習もしてみよう。

（１）import って何してたの？

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

ここでは、プログラムを動かす準備として、「便利な道具」を読み込んでいました。
「便利な道具」を**モジュール**と言います。このモジュールが集まったものが**ライブラリ**です。
ライブラリには初めから python に組み込まれている**「標準ライブラリ」**と、外部からダウンロードする**「外部ライブラリ」**があります。”組み込まれている”といっても、最初は眠っている状態のようなもので import で読み込まないと使えません。import で読みこんでおけば、プログラムの中で使うことができます。

（１）－１　標準ライブラリについて

まず､標準ライブラリの **sys モジュール**はプログラムを実行するために必要なモジュールです。
また､**time モジュール**は時刻の取得､変換､プログラムの一時停止をするためのモジュールです。
プログラムの中に

```
time.strftime () →日時を読み込むためのモジュール
```

とか

```
time.sleep () →ちょっとインターバルを取るためのモジュール
```

が出てきます。これは、time モジュールを読み込んでいるから使えるのです。
ちなみに、time.strftime モジュールでは、以下のような書式文字列を使うことができます。

文字	意味	例
%Y	西暦 4 ケタ	2018
%m	月 2 ケタ	11
%d	日 2 ケタ	22
%A	曜日	Sunday

time.strftime () モジュールはその時の日時を読み込んでくれます。

```
   fn = “envirophat-%s.csv” % (time.strftime( “%y%m%d”)) #8 行目のスクリプト
```

つまりこれはファイルネームを envirophat-20231217s.csv にしてくれるのです。
まぁ･･･ラズパイをネットに繋いでいないと、正確な時刻になってくれないんだけどね。

（１）－２　外部ライブラリについて

次に、外部ライブラリの envirophat です。これは enviroPHAT センサーを使用するためのライブラリです。事前に河野先生が皆のラズパイに徹夜でダウンロードしておきました。

他にもラズパイをネットに繋げば、色々なライブラリをダウンロードすることができます。その全貌を紹介しましょう。**今日のマニュファクチャリングで好きなモノを使って下さい。**

envirophat ライブラリに用意されているモジュール

【weather モジュール】

```
weather.altitude()      現在の高度　（どうやって計測しているのだろう…？不思議。）
weather.temperature()      気温　（ただし、ラズパイ本体の熱が影響するようだ。）
weather.pressure(unit=○○)気圧　○○の部分でヘクトパスカル(hPa)かパスカル(Pa)を選べる。
```

【light モジュール】

```
light.light() 照度
light.rgb() RGB 照度（赤、緑、青の「光の三原色」成分）
```

【motion モジュール】

```
motion.accelerometer()　加速度計
motion.magnetometer()　磁力計
motion.heading() 方位計
```

【analog モジュール】

```
analog.read_all() センサーの左端にある「ANALOG IN」の 0,1,2,3 のピンに電流が流れると、
                  その値を拾う。
                  今回は使わないけど今後のモジュール開発においては役に立つかも…！
```

【leds モジュール】

```
leds.on()     LED が光る！
Leds.off()    LED が消える！
```

ちなみに
/Pimoroni/envirophat/examples/の中にある all.py を実行すると、このセンサーの持つ全ての機能でデータを取ります。（ただし、記録はしないしデータを表示するだけ。）
これを nano で開くと、それぞれのモジュールの使い方がわかります。

（２）def って何してたの？

Python には様々な組み込み関数がありますが、自分だけのオリジナル関数を作るコトができます。それが def 文です。

作ってみよう！←このマークが出たら、新しく nano を立ち上げて（ファイル名）.py を作ってみよう。

```
$nano hello.py
```

```
def helloworld():
    name = input('input your name: ')
    message = 'Hello ' + name + ' !'
    print(message)

helloworld()                    #4 行目までで創った関数を呼び出すためにこの行がある。
```

書けたら

```
$python3 hello.py
```

で出力してみよう。

（３）while って何してたの？

while は繰り返し文です。データを取り続けるには、この while で測定を繰り返すのです。
簡単な while 文を作ってみましょう。

作ってみよう！

```
$nano sheep.py
```

```
count = 0
while count < 10:
    count += 1
    print( '{}sheep' .format(count))
print( 'Good night...' )
```

ひつじが 1 匹…ひつじが 2 匹…と 10 匹数えるまで繰り返したわけですね。
ちなみに、これで print 関数もなんとなくわかってきたと思います。
これは、()内の引数を表示してくれる関数です。

（４）if 文って何してたの？

if 文は分岐を表します。

作ってみよう！

```
$nano test.py
```

```
score = int(input( 'Enter your Test score >>' ))
if score < 0 or score > 100:
   print( 'It is an error' )
   print( 'Please re-enter' )

elif score >= 60:
   print( 'Good!' )
   print( 'You are excellence!' )

else:
   print( 'Oh my God!' )
   print( 'You are no good...' )
```

テストの点を入力してね。
０～１００以外を入れると、「エラーだよ。点数を入れ直してね！」
６０以上を入れると、「合格だよ！優秀だね！」
それ以外では、「なんてこった！君はだめだ・・・」
と表示しているわけです。場合分けの方法は他にも色んな入力方法があります。

（５）with open 関数って何してたの？

作ってみよう！

```
$nano diary.py
```

```
text = input( ‘What are you doing? >>’ )
with open( ‘diary.txt’ , ‘a’ ) as file:
    file.write(text + ‘＼n’ )
```

これは日記なのです。最初に、「君何した？」と聞かれてるので、何か書いて下さい。
すると、diary.txt というファイルが作られ、その中にどんどん書き込んだ内容が追加されていきます。
何回も diary.py を起動して、日記を書いてみて下さい。
diary.txt の中に日記が溜まっていきます。

open(‘diary.txt’ , ‘a’)の ’ a’ は追記モードを指定しています。
同じファイルにどんどん上書きしていくという意味ですね。
with 文で書いた命令が終了したら、開いていた diary.txt も一緒に閉じる…という設定です。
まぁ、やってみて！

ちなみに、‘＼n’はエスケープシーケンスと言って、改行させるコマンドです。
これが無いと、連続した行に入力し続けます。よかったら、’＼n’を抜いて試してみてね。見にくいよ～！

（６）try-except 文とは？

try は、**「プログラム実行中にエラーが発生してもいちいち止めずに表示を続ける！」**という感じです。
あまりに変な構文エラーで無い限り、try 以下の命令を続けるのです。
それを止めるのが、except の後のコマンド。
KeyboardInterrupt とは、Ctrl+ C でプログラムを終了するよ！という意味。

というわけで、今回データロガーに使う構文を一通り説明してみました。
まだまだ謎は多いと思うけど、書き換えながら理解していってね！

４．制御文字でグラフ化してみよう！

ここまで触ってわかるように、RaspberryPi は CUI 形式なのでグラフ表示などのグラフィカルな表現ができません･･･。

と、思いきや！

Python は GUI でなくても、グラフィカルな表現を使うことができます。それが制御文字です。

例として、ランダムな数値を表示するプログラムを示します。

これを実行してみれば意味がわかります。

作ってみよう！

```
$nano graph.py
```

```
import random, sys, time
color = 91 # 90:gray, 91:red, 92:green, 93:yellow, 94:purple

while True:
    w = 60                ←画面の幅（各人の画面設定によって違う。変化させてみるべし。）
    i = int(random.random() * w)
    sys.stderr.write( “\033[%dm” % (color))  ←\033[%dm 装飾開始のコマンド。なぜか 033。
    sys.stderr.write( ‘\r’ + ( ‘o’ * i) + ‘ ’ * (w-i))  ← ‘o’ の部分を他のものに変えてみると…？
    sys.stderr.write( “\033[0m” )             ←\033[0m 装飾終了のコマンド。
    sys.stderr.write( ‘ ’ + ( ‘%02d’ % (i)))    ←実行したらわかるけど右端に表示される数値。
    sys.stderr.flush()
    time.sleep(0.3)
```

終了は Ctrl + C です。

５．本日の仕上げ課題に挑戦しよう！

課　題１”照度”を制御文字でグラフ化してみよう！

下記が基本となるスクリプトです。完成したら、色を変えたり色々な改造をしてみましょう。

```
$nano lightsensor.py
```

```
import random, sys, time
from envirophat import light

color = 91 # 90:gray, 91:red, 92:green, 93:yellow, 94:purple

while True:
    max = 3000      表示するセンサーの最大値（本当は 65535 まで行く。）
    w = 60          画面の幅（各人の画面設定によって違う。変化させてみるべし。）
    v = light.light()
    i = int(v * ( w / max))    良い感じに割り算して画面からはみ出さないようにしている。
    sys.stderr.write( “\033[%dm” % (color))     ←\033[%dm 装飾開始のコマンド。
    sys.stderr.write( ‘\r’ + ( ‘o’ * i) + ‘ ’ * (w-i))
    sys.stderr.write( “\033[0m” )               ←\033[0m 装飾終了のコマンド。
    sys.stderr.write( ‘ ’ + ( ‘%02d’ % (i)))
    sys.stderr.flush()
    time.sleep(1)    １秒間隔で。
```

課　題２

制御文字を表示しながら、特定の条件を満たすと LED が点灯するオリジナルセンサーを作ろう！

下記が「ある値より照度が暗くなったら LED が点灯するセンサー」です。

```
$nano lightsensor.py
```

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

color = 92
INTERVAL = 1
DRAW_GRAPH = True

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

def draw_graph(v)
    max = 10000
    s = 90
    w = 80
    v = light.light()
    i = int(led * ( w / max))
    sys.stderr.write( “＼033[%dm”  % (color))
    sys.stderr.write( ‘＼r’ + ( ‘o’  * i) +  ‘ ’ * (s-i))
    sys.stderr.write( “＼033[0m” )
    sys.stderr.write( ‘ ’ ) + ( ‘%02d’  % (led)))
    sys.stderr.flush()

try:
    while True:
      led = light.light()
      max = 3000
      w = 50
      i = int(led * w / max )
      vs = [led]
      if DRAW_GRAPH:
         draw_graph(i)
         write( “,” .join(map(str,vs)))
      else:
         pass

      if i > 25:
         leds.off()
      else:
         leds.on()
      time.sleep(INTERVAL)

except KeyboardInterrupt:
     pass
```

これを改造して、特定の方向を向けると LED が点灯するセンサー…なども創ってみよう。

６．プログラムを自動起動する（crontab）

Linux には crontab というプログラムを決まった時間に起動するための仕組みがあります。 以下のコマンドで起動されるエディタを使って、設定を行います。 初回起動時のみ利用するエディタを聞かれますので、好みのものを選択してください。 ここまで、このテキストを読み進めて来ている方は、nano がオススメです。　※これは Linux のコマンドなので、python プログラムでは無いのよ。

以下の青い部分は、まだ読むだけにしてね！

```
$ crontab -e
```

```
：省略

# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

※ここの一番下に自動で起動したいプログラムを書く。
```

たとえば毎日 15 時に hello.py という Python のプログラムを起動するには以下のように書きます。

```
0 15 * * * python3 hello.py
```

前半の 5 つの数字 (またはアスタリスク) がそれぞれ「分 時 日 月 曜日」となっていて、「*」が指定された場合は、「全て」という意味になります。以下のように書くと、毎月 1 日の 14:30 に起動する、という意味になります。また、「#」で始めた行はコメントとして扱われ処理内容に影響しません。メモを残す場合などに利用します。

```
30 14 1 * * python3 hello.py
# Run only on the first day of each month.
```

他にも特殊な構文があります。

@reboot に続けて書かれたコマンドは、システムが起動した直後に 1 度だけ呼ばれます。電源を入れたらいつも動かしたいプログラムがある場合は、ここに書いておくと便利です。

```
@reboot python3 hello.py
```

実際には、ただ起動するだけだと何が起こっているのか分からなくなることが多いため、以下のようにリダイレクトとパイプを使って（詳細は割愛）、マシンの上に記録（ログ）が残るように運用すると便利です。

```
@reboot python3 hello.py 2>&1 | logger -p cron.info -t "hello"
```

ログは/var/log/syslog というファイルに書き出されています。上記のように hello というタグをつけた場合、以下のように確認することが出来ます。

```
$ sudo grep hello /var/log/syslog
```

現在の設定を確認するには、「-e」の代わりに「-l」オプションを使用します。

```
$ crontab -l
```

「-r」オプションを指定すると、設定が全て消去されます！

```
$ crontab -r  # danger!!
```

では、やってみよう！

まず、下準備で第１回目で創った「mysensor.py」がある場所を確認しておきましょう。
その場所で作業を行います。

```
$ crontab -e
```

```
：省略

# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command


```

いろいろ説明が表示される。一番下にこれをそのまま書き込む。改行しなくていいよ。

```
@reboot sleep 60; python3 ○○○/○○○/mysensor.py -i 60 2>&1 | logger -p
  cron.info -t "enviro"
```

※ここは mysensor.py が置いてあるディレクトリの場所を指定しましょう。
mysensor.py が置いてある場所で$pwd を実行すると場所がわかります。

※ここの数字は 60 秒（１分ごと）にデータを取るという意味。
すぐに確認したい場合は、10 秒などに設定して様子を見ましょう。

書けたら Ctrl＋O（オー）で保存して、Ctrl+X で外に出る。

マシンを再起動します。

```
$ sudo reboot
```

しばらく待って、○○○/○○○/と指定したディレクトリに.csv ファイルが出来ているか確認します。logger コマンドにリダイレクトされた出力は以下のコマンドで確認することもできます。

```
sudo grep enviro /var/log/syslog
```

どう？

コレが上手くいっていれば、画面もキーボードも無く、電源を入れるだけで測定を開始するセンサーが組めてしまいます。

ここまでで一通りの工作は終了！
ACT-SI で使おうと思ったら、ここまでの話で十分にものづくりは可能です！

次回はネット接続の方法を勉強していきます。

参考文献・引用元：Raspberry Pi 入門（https://lmlab.net/books/2102_raspi/index.html）
著者：伊藤陽生（伊藤製作所・株式会社ランバーミル）