プログラミングPBL

「Raspberry Piでデータロガーを作ろう！」

～第3回～

今回はRaspberryPiで作ったデータロガーを
インターネットにつないで遠隔操作してみよう！！

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！

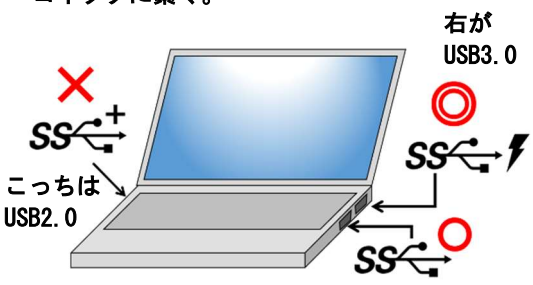
❶セットを確認しよう。

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:
Password:                    ←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. 前回の復習

前回までに学んだコマンドや、代表的なスクリプトをもう一度掲載します！

(1)基本的な Linux のコマンド

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ファイルやディレクトリの消去（ディレクトリの消去は -r のオプションを付ける）

```
$ rm （消したいファイル）　　または　　$ rm -r（消したいディレクトリ）
```

⑥自分がいるディレクトリの確認

```
$ pwd
```

⑦時刻の確認

```
$ date
```

※ネットに繋いでないと古い時刻が出ます。

⑧ディレクトリを作る

```
$ mkdir （ディレクトリ名）
```

⑨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）
```

⑩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

⑪python3 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

⑫ラズパイの再起動

```
$ reboot
```

⑬ラズパイの終了

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

（2）センサーの原型となる「main.py」

このスクリプトでは、温度・照度・加速度を測定できます。また、照度をグラフ化して表示しています。収集したデータは、csv ファイルとして保存されます。

```
$ mkdir enviro-logger
$ cd enviro-logger
$ nano main.py
```

```
#!/usr/bin/env python
# Based on Pimoroni exsample all.py.

import argparse, os, sys, time
from envirophat import light, weather, motion, analog

def write(line):
    fn = os.path.dirname(__file__)
    fn += "/%s-%s.csv" % (args.name,time.strftime("%y%m%d"))
    sys.stderr.write("File: %s¥n" % (fn))
    with open(fn, mode='a') as f:
        f.write(line+"¥n")

def draw_graph(v,max_v):
    w = 60
    i = int(v * (w / max_v))
    sys.stderr.write("¥033[92m")
    sys.stderr.write('¥r' + ('o' * i) + ' ' * (60-i))
    sys.stderr.write("¥033[0m")
    sys.stderr.write(' ' + ('%04d' % (v)))
    sys.stderr.flush()

parser = argparse.ArgumentParser(description='Envirophat interface')
parser.add_argument('-g','--graph',action='store_true',help='Show graph')
parser.add_argument('-n','--name',help='Name of CSV')
parser.add_argument('-i','--interval',type=int,default=1)
args = parser.parse_args()

try:
    while True:
        d = time.strftime("%F")
        t = time.strftime("%T")
        vs = [d,t]
        # Temp,Light,Accelerometer X,Y,Z
        temp = weather.temperature()
        ligh = light.light()
        vs += [round(temp,2),ligh]
        vs += [round(x,2) for x in motion.accelerometer()]
        if args.graph: draw_graph(ligh,5000)
        else: write(",".join(map(str,vs)))

        time.sleep(args.interval)

except KeyboardInterrupt:
    pass
```

（3）LEDを点灯させるセンサー

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…なども作れます。

（4）プログラムを自動起動させる準備（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 nano を指定して下さい。
※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.←これはメモ。
```

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 10 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 10 秒ごとにデータを取るという意味。

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。

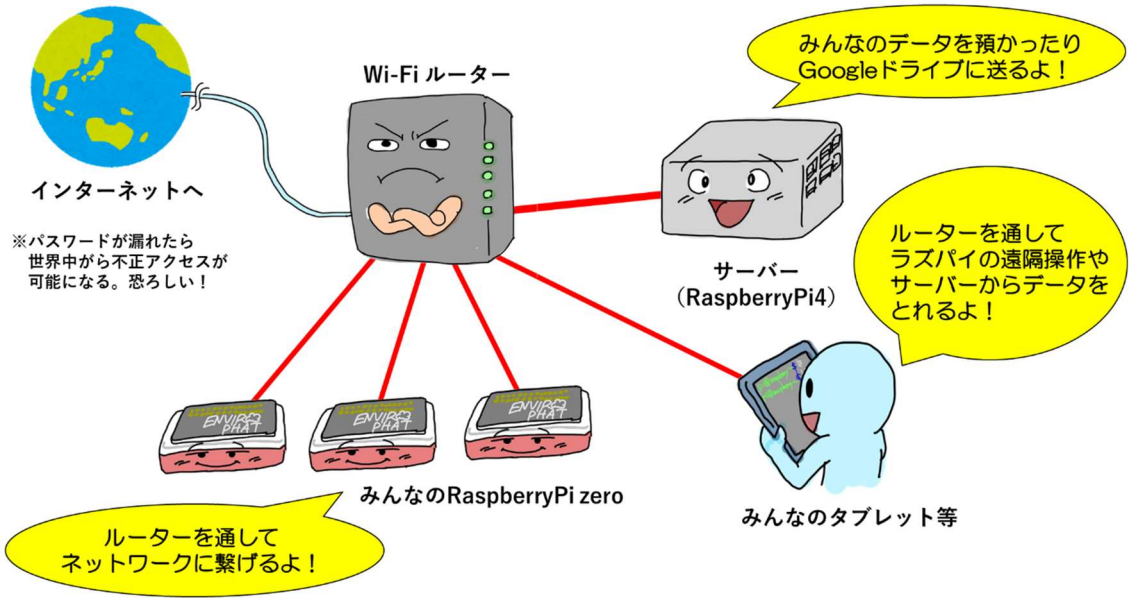
```
$ sudo reboot
```

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
```

2. ネットワークに接続する

いよいよ RaspberryPi をネットワークに繋ぎます。ネットワークは概ね以下の図のようなイメージで、Wi-Fi ルーターを中心として構築されています。



（1）Wi-Fi ルーターに接続する

まずは RaspberryPi と Wi-Fi ルーターの接続です。コマンド入力で行います。

/etc/wpa_supplicant/というディレクトリにある wpa_supplicant.conf というファイルに接続先を入力することでルーターに接続できます。

ルーターへの接続には SSID とパスワードが必要です。

今回使用する Wi-Fi ルーターの SSID とパスワードはこちら

ＳＳＩＤ：[redacted]

パスワード：[redacted]

というわけで、以下の手順で wpa_supplicant.conf を書き換えよう。

コマンド入力は「Tab キー」を活用すると素早く正確になるよ！

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

ファイルの中身を次の画像のように書き換える。

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=JP
network={
        ssid="[redacted]"
        psk="[redacted]"
        key_mgmt=WPA-PSK
}
```

この 3 行に書き換えてください！ key_mgmt とは、通信の暗号化の方法を示しています。コレが大事。

書き終わったら、Ctrl+X → Y → Enter でＯＫ。

nano は保存していないまま終了しようとする（Ctrl+X を押す）と、「保存する？」と聞かれるので、そこで Yes と答えて（Y を押して）Enter を押せば保存できます。

(2)自分のIPアドレスと、接続先のIPアドレスを入力する

IP アドレスは、インターネット上のコンピューターやスマートフォンなどの住所のようなものです。IP アドレスによって、コンピューターはインターネット上でお互いを見つけて通信することができます。例えば「192.168.15.1」のように数字で表されます。

通常は Wi-Fi ルーターから各 RaspberryPi に対して自動で IP アドレスが割り振られるのですが、今回はみなさんの RaspberryPi に対して、固定した IP アドレスを決めておきます。

以下の手順で入力してください。

```
$ sudo nano /etc/dhcpcd.conf
```

ファイルの一番下に、下記の 3 行のコメントを追加する。

```
static ip_address=192.168.[redacted].●●/24
static routers=192.168.[redacted].[redacted]
static domain_name_servers=192.168.[redacted].[redacted]
```

これが君達のIPアドレス！
●●の部分は、先生に指定された番号を入力してください！

これが今回のルーターの IP アドレス。
接続先の番号ですね。

ここまで入力できたら一度再起動をしましょう。

```
$ reboot
```

(3)インターネットに接続できているか確認しよう

ラズパイがうまくネットワークに繋がっているかどうか確認するためのコマンドを幾つか紹介します。

方法その1 ip addr **コマンドで確かめる！**

ip addr コマンドでは、現在の自分自身の IP アドレスを確認することができます。

```
$ ip addr

 :省略

2: wlan0: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc pfifo fast state UP
group default qlen 1000
    link/ether [redacted]
    inet [redacted]/24 brd 192.168.9.255 scope global noprefixroute wlan0
       valid lft forever preferred lft forever
    inet6 [redacted]
mngtmpaddr noprefixroute
       valid lft 298sec preferred lft 298sec
    inet6 [redacted]
       valid lft forever preferred lft foreverping
```

※ここに君たちが設定したラズパイの IP アドレスが表示されていればOK！！
（これは先生が使っていたラズパイのものです。）

方法その2 ping **コマンドで確かめる！**

「ping」コマンドでは、インターネット上の別のサーバーと通信が出来るかどうか、またその際の遅延速度を確認できます。試しに、今回サーバーとして使う先生のラズパイ 4 と通信してみましょう。

先生のラズパイ４の IP アドレスは [redacted] です。

```
$ ping [redacted]
PING [redacted] ([redacted]) 56(84) bytes of data.
64 bytes from [redacted]: icmp_seq=1 ttl=64 time=27.1 ms
64 bytes from [redacted]: icmp_seq=2 ttl=64 time=43.8 ms
：省略
--- [redacted] ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7012ms
rtt min/avg/max/mdev = 7.716/23.211/43.781/12.407 ms
```

コマンドを終了するには「Ctrl+C」をタイプしてください。

方法その3 traceroute **コマンドで確かめる！**

もう少し専門的なコマンド「traceroute」では、目的のホストに到達するまでにどんなルーターを経由しているかを調べることが出来ます。試しに Google.com までの道のりを調べると…

```
$ traceroute google.com
traceroute to google.com (172.217.25.174), 30 hops max, 60 byte packets
 1  aterm.me ([redacted])  2.545 ms  2.291 ms  3.070 ms
 2  153.153.246.233 (153.153.246.233)  5.198 ms  4.981 ms  4.856 ms
 3  153.153.246.69 (153.153.246.69)  5.067 ms  4.769 ms  5.175 ms
 4  118.23.43.149 (118.23.43.149)  9.237 ms  10.278 ms  11.918 ms
: (省略)
```

と、12 個以上のルーターを経由して Google にたどり着くことがわかりました。

（４）いよいよサーバーに繋ごう！サーバーってなんだ？

ネットワークの接続が確認できたら、いよいよサーバーと繋ぎます。

サーバーは、データや情報を集めて保存する「大きな箱」のようなものです。

今回はラズパイ４をサーバーにしています。

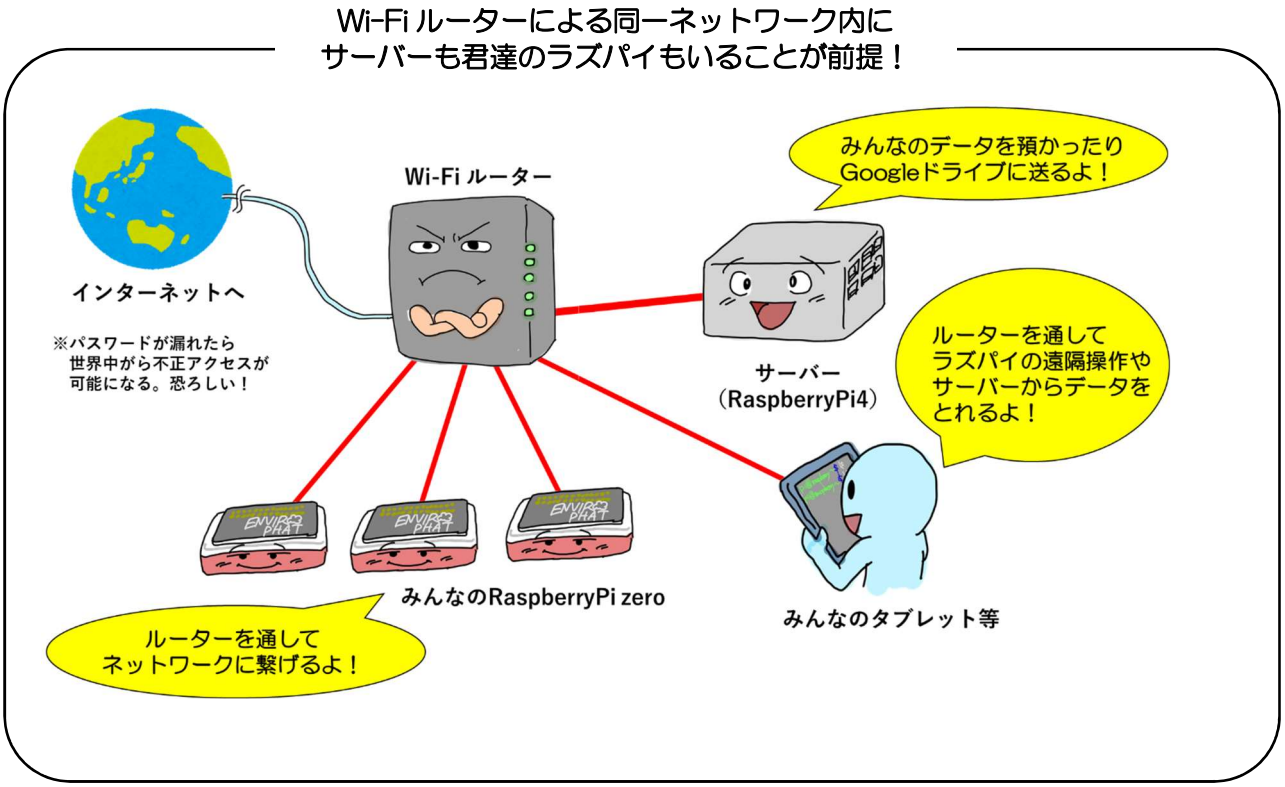
例えるなら、ラズパイ４は学校の先生のような存在で、他の小さい Raspberry Pi zero（生徒たち）からたくさんの情報を集めて、それを整理して保管します。

生徒たちが宿題やレポートを先生に提出するように、小さい Raspberry Pi zero は測定したデータをサーバーに送ります。そして、サーバーはそれらのデータをしっかりと保管し、私たちがいつでもその情報を見ることができるようにしてくれます。

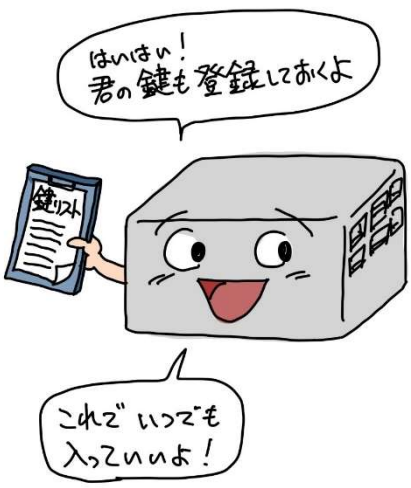
（５）SSH（Secure Shell）でサーバーにつなごう！

今回は、先生が用意した「Raspberry Pi4」をサーバーにして、皆の計測データを集めてみます。
他のラズパイにつなぐためには「SSH」を使います。
「SSH」とは他のパソコンをリモート操作するための通信手順で「Secure Shell」の略です。
君たちは SSH 校で SSH をするのです（笑）。

さて、Enviro pHAT で取得した観測データをサーバーに SSH を使って転送する準備をします。
が、ここで大切な前提があります。
先生が構築したサーバーが同一のネットワーク内で稼働していて、かつサーバーのログインパスワードが分かっていることが前提です。



まずは、君達のラズパイがサーバーにアクセスするために鍵を作ってサーバーに登録します。
ラズパイが鍵を作る → サーバーに鍵を登録する → サーバーに入れるようになる。
イメージは下の図のような感じです。



というわけで、まずは君達のラズパイに鍵を作ります。

```
$ ssh-keygen
```

何か入力を促された場合、ここはひたすら Enter を押して進んで下さい。

これで、ラズパイは鍵を作りました。鍵は、.ssh という隠しディレクトリに作成されます。

次に、この鍵をサーバーに登録します。以下のコマンドを実行します。

```
$ ssh-copy-id   -i   ~/.ssh/id_rsa.pub   enviro-host.local
```

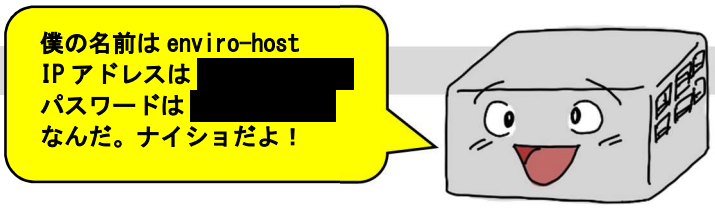
上手くいけば「接続しますか！？」というような内容を聞かれるので

```
yes
```

と入力して進みましょう。
正常に接続できるとパスワードを聞かれますので、このサーバのパスワードを入力してください。

サーバーのパスワードはこちら

```
password: [redacted]
```



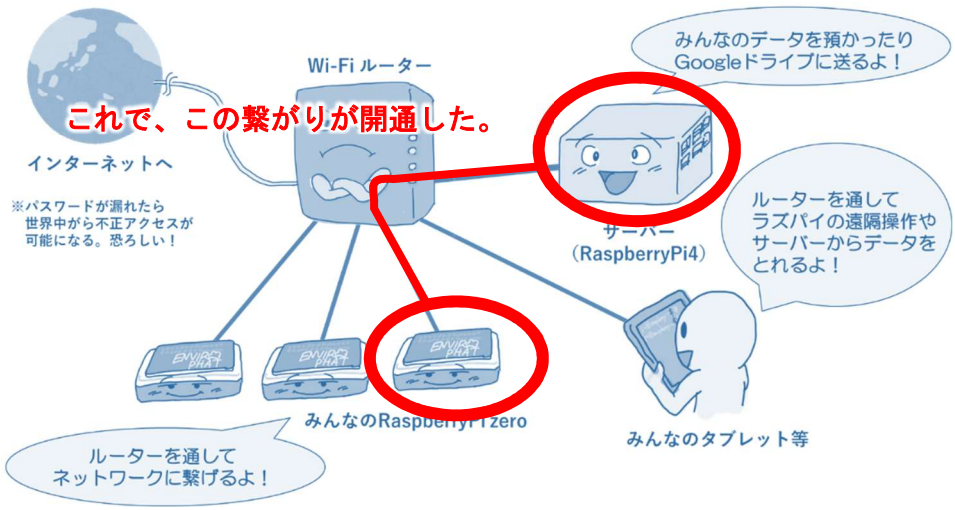
ここでうまく繋がらない場合、ネットワーク接続に問題がある場合があります。
この章の最初に紹介した ip addr や ping などのコマンドを使ってネットワークへの接続状態を確認してください。それでも繋がっていないときは、各ラズパイの固定 IP をチェックしてください。
名前解決に問題がある場合、端末を再起動することで改善する場合があります。

鍵の登録が出来たら、以下のコマンドで接続が可能かを確認します。

```
$ ssh enviro-host.local hostname
```

サーバーの名前「enviro-host」が表示されたら正常です。

サーバに接続できたら準備完了です。



（６）「rsync」でサーバーにデータを送ってみよう！！

では、早速サーバーのラズパイ４にファイルを送ってみよう。

とりあえずダミーデータとなるテキストファイルを作ってみよう。

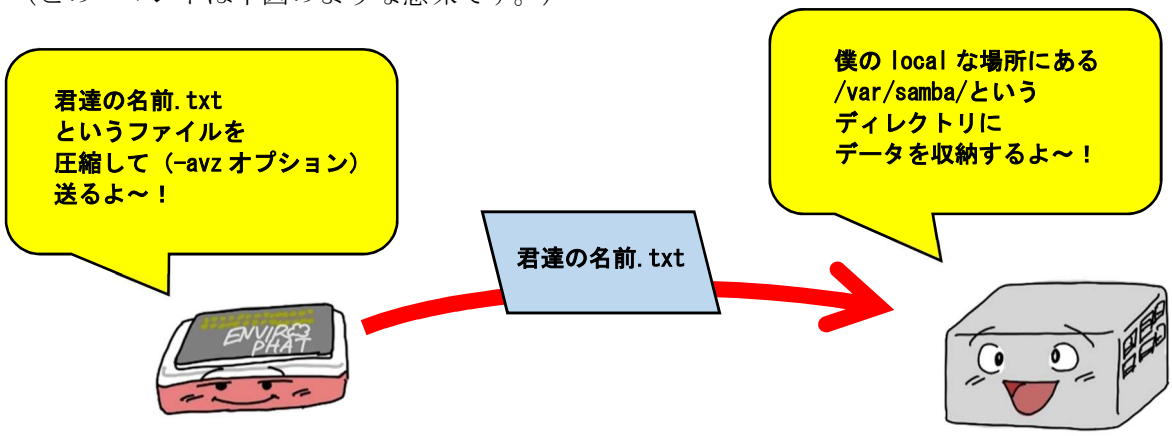
```
$nano　君達の名前.txt
```

中になにか適当な文章を書いて、Ctrl+X →Y →Enter で保存しよう。

データをサーバに転送するには「rsync」というコマンドを利用します。

```
$ rsync　-avz　君達の名前.txt　enviro-host.local:/var/samba/
```

（このコマンドは下図のような意味です。）



（７）「main.py」と「rsync」を crontab に登録すると凄いことになる！（今回はしなくてOKだけど）

これらのコマンドは crontab に登録すると便利です（P.5 参照）。

main.py（P.4 参照）が完成していたら、一緒に登録してみよう。

以下はラズパイ起動と同時に 10 分おきに測定データを転送する例です。

君達のラズパイの/enviro-logger/というディレクトリに main.py があるとします。

```
$crontab -e
:省略
（一番下に書き込もう）
@reboot sleep 60; python3 enviro-logger/main.py -n　ssh（2 桁の IP 番号）
 -i 600 2>&1 | logger -p cron.info -t "enviro"

*/10 * * * * rsync -avz　enviro-logger/*.csv　enviro-host.local:/var/samba/
```

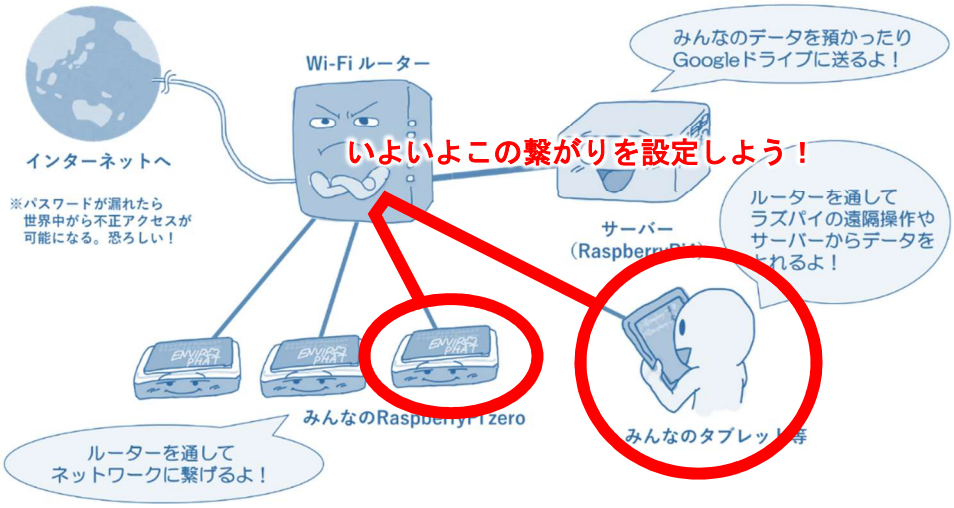
これは 1 行で書いてね

これは enviro-logger ディレクトリの全ての csv ファイルという意味です。

以上で、すべてが上手くいけば君の Raspberry　Pi は電源を入れれば自動でデータを計測し、自動でサーバにデータを送るようになった。

3. WebSSHを使って、タブレットでラズパイに入っちゃおう！！

自動起動も良いのですが、タブレットから直接ラズパイを操作できた方が
はるかに便利じゃない！？ということで･･･。



（1）君達のタブレットを、ラズパイと同じ Wi-Fi に繋ごう

まずは、君達のうち、1 人でいいのでタブレットを同じネットワークに繋ごう。
以下のネットワークを探して、接続してください。

今回使用する Wi-Fi ルーターの SSID とパスワードはこちら

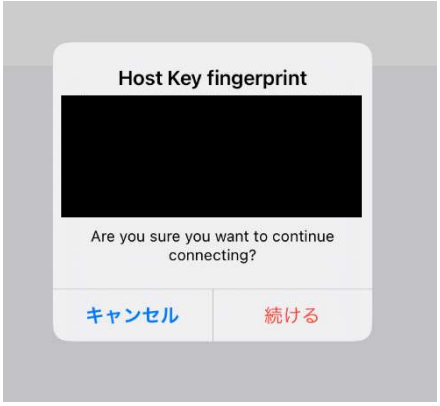
ＳＳＩＤ：[redacted]
パスワード：

（2）WebSSHを起動しよう

ラズパイの電源が入っていること、タブレットがネットワークに繋がっていることを確認したら WebSSH を起動しよう。

右上の「＋」を押して、以下の 3 箇所を入力して新しい接続を設定しよう。

下記のような警告が出ますが、内容としては「信頼できない接続だけどいいのか！？繋ぐぞ！？」といわれています。そのまま「続ける」をタップ。



あとは、下記のIPアドレスをタップして、ＳＳＨで接続しましょう。

上手くいけば繋がります。

詳しい操作方法は次ページに。

《WebSSH の大体の使い方》

・・・という訳で

ここまでが準備運動！！

今日のものづくりお題は次ページ！！

本日の課題

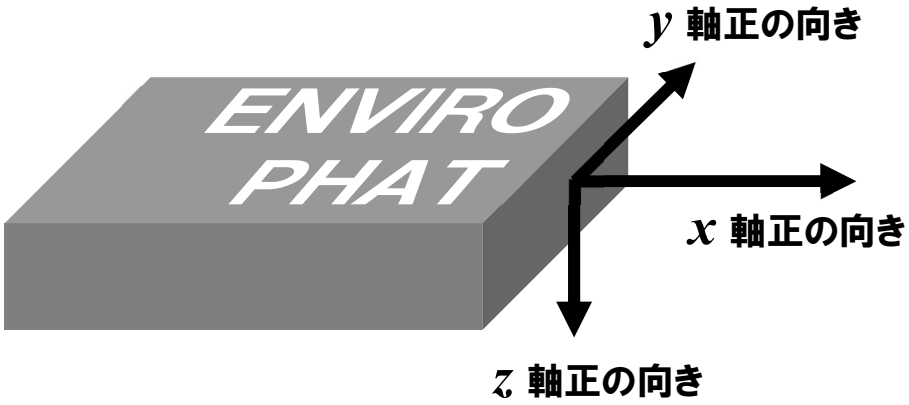
RaspberryPi で遠隔操作できる加速度計を作り、モバイルバッテリーを取り付けて力学台車に乗せ壁に衝突させた時の加速度グラフを作ってみよう！

大まかな手順

①加速度測定用のスクリプトを作ろう（次ページにサンプル掲載。）

②モバイルバッテリーに繋ぎ直し、力学台車に乗せよう。

なお、加速度計の軸の向きは以下のようになっています。台車に乗せる向きを工夫して。



③WebSSH を使ってスクリプトを起動しよう。

④ゆっくりと壁にぶつけながら、加速度の変化を計測しよう。

⑤rsync を使ってサーバーにデータを送ろう。

```
$ rsync  -avz  （加速度の csv ファイル）  enviro-host.local:/var/samba/
```

⑥サーバーに集まったデータは、先生が Google ドライブに送ります。

Google の共有ドライブ→ACT-SI1 生徒用（令和 5 年度入学生）→RaspiData にいれます。

パソコンで自分たちの測定データを回収し、エクセルでグラフを作ろう。

完成したら見せて下さい。

加速度計のスクリプト（例）

```
$ nano accel.py

import time
from envirophat import motion
import csv

# CSV ファイル名の指定
file_name = "accel_data●●.csv"    ←自分たちの IP 番号を入れてね。

# 無限ループでデータを記録
while True:
    with open(file_name, "a") as file:
        writer = csv.writer(file)
        x, y, z = motion.accelerometer()
        current_time = time.strftime("%Y-%m-%d %H:%M:%S")
        writer.writerow([current_time, x, y, z])
    # 1 秒間隔で測定
    time.sleep(1)
```

《目指す答え》

壁に６回衝突させて得られた加速度グラフです（キレイに測定できてるでしょ）。
このようなグラフを作って下さい。