

抜粋版

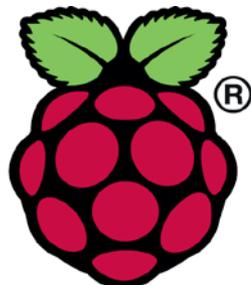
クラウド型温湿度ロガー開発キット

～AWSを使って、温湿度情報の活用により、感染予防、農業等に貢献～

実践編



AWS IoT



Raspberry Pi

スペクトラム・テクノロジー株式会社

<https://spectrum-tech.co.jp>

sales@spectrum-tech.co.jp

温湿度ロガー開発キット 目次

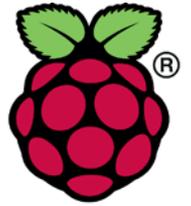
Pi運用マニュアル

- | | ページ |
|----------------------|-------------------|
| 1. RaspberryPiについて | 3 |
| 2. Linux基本コマンド | 4 |
| 3. 基本操作 | 5 |
| 4. 日常運用(ウイルススキャン、更新) | 6 |

AWS IoT開発

- | | ページ |
|-----------------|--------------------|
| ① メニュー | |
| ② AWS設定 | 8 |
| ③ AWS IoT設定 | 9 |
| ④ 温湿度ロガー | 16 |
| • 全体構成 | 29 |
| • 絶対湿度 | 31 |
| • BME280接続 | 35 |
| • 単体試験 | 36 |
| • AWS IoTルール設定 | 37 |
| • プログラム設定 | 43 |
| • IoT折り返し試験 | 45 |
| • DynamoDB確認 | 46 |
| • リアルタイム表示設定、確認 | 48 |
| • IAMユーザ追加 | 51 |
| • CSVデータ取得 | 55 |
| • Cloudwatch使い方 | 57 |

ページ 抜粋版のため、ページと本文は一致しません



RaspberryPi運用マニュアル

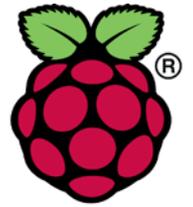
1. Raspberry Piについて

既に全世界で1000万台以上販売された手のひらサイズのコンピュータです。
LinuxベースのRasbianOSで動作しております。

2. Linux基本コマンド

① システム関係

- 起動: 電源を入れると自動で起動します。
- 再起動: `$ reboot`
又は、`menu>shutdown>reboot`; 左上のメニューから
- 終了: `$ shutdown`
又は、`menu>shutdown>shutdown`; 左上のメニューから
- ログアウト `$ exit`
又は、`menu>shutdown>logout`; 左上のメニューから
- **日本語／英語の入力切替**: キーボードの CTL と `j` を同時に押します (コントロール: 左下と `j`)



RaspberryPi運用マニュアル

2. Linux基本コマンド

② ディレクトリ操作、コピー、移動、削除

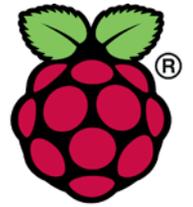
pi@raspberrypi:~\$ **cd** /home/pi/Documents ディレクトリの切り替え
 pi@raspberrypi:/home/pi/Documents\$ **ls** ファイルとディレクトリの表示(表示したら操作したいファイルを右クリックでコピーして操作します)
 pi@raspberrypi:~\$ **cp** ファイル名 ディレクトリ 配下のディレクトリのファイルを別のディレクトリへコピー
 pi@raspberrypi:~\$ **mv** ファイル名 ディレクトリ 配下のディレクトリのファイルを別のディレクトリへ移動
 pi@raspberrypi:~\$ **sudo rm** ファイル名 ファイルの削除
 便利な機能 **rm -help** コマンドのオプションが分からない場合は、ヘルプで問い合わせる。すべてのコマンド共通(マイナスを2個とhelp)

③ ユーザ権限、プロセス他

pi@raspberrypi:~\$ **su -** スーパーユーザ(root)に切り替え、パスワードを入力
 pi@raspberrypi:~\$ **ps a** 現状の動いているプロセスを表示
 pi@raspberrypi:~\$ **kill** 特定のプロセスを強制終了
 pi@raspberrypi:~\$ **sudo apt-get install pkg** パッケージのインストールなどに使用
 pi@raspberrypi:~\$ **date** 日付、時間の設定を行います。
 pi@raspberrypi:~\$ **sudo leafpad /etc/network/interfaces** インタフェースに記述している内容を変更します。Viよりも使いやすいです。

④ モジュール、usb、メモリ、HDDなどの表示

pi@raspberrypi:~\$ **lsmod** linuxのモジュールリスト表示
 pi@raspberrypi:~\$ **lsusb** usbのデバイス表示
 pi@raspberrypi:~\$ **free -mt** メモリ使用状態表示
 pi@raspberrypi:~\$ **df** HDD(マイクロSD)の使用状態表示

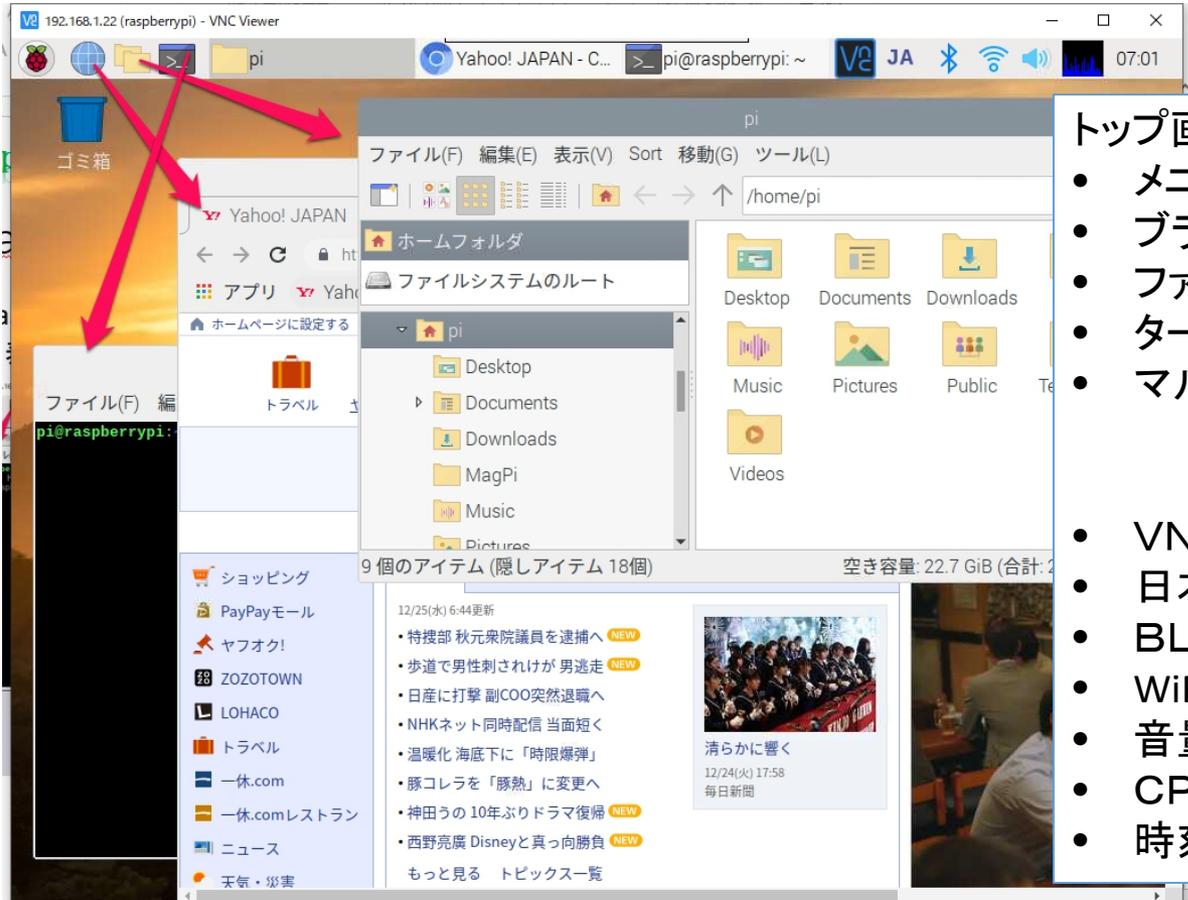


RaspberryPi運用マニュアル

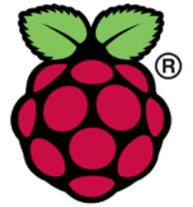
3. Raspberry Piの基本操作

① 表示画面と内容

デスクトップ上によく使うコマンド.txtがあります。コピペして使ってください



- トップ画面(上段のタスクバーで選択)
- メニュー
 - ブラウザ
 - ファイルマネージャ
 - ターミナル
 - マルチ画面選択
-
- VNC
 - 日本語入力
 - BLE
 - WiFi
 - 音量
 - CPU使用率
 - 時刻



RaspberryPi運用マニュアル

4. 日常運用

① セキュリティ対策(アンチウイルス更新、スキャン)

- アンチウイルス対策として無料のclamAVをインストールしてます。
- 手動での運用を基本としています。

パターンファイル更新

手動スキャン時に更新されます

手動でスキャン

\$ sudo clamscan --infected --remove --recursive
自動化可能ですが、バックグラウンドで重くなる可能性大。コマンド入力後約5分位かかります。

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
ERROR: /var/log/clamav/freshclam.log is locked by another  
ERROR: Problem with internal logger (UpdateLogFile = /var/  
og).  
root@raspberrypi: ~# leafpad /etc/clamav/freshclam.conf  
root@raspberrypi: ~# freshclam  
ClamAV update process started at Fri J  
main.cvd is up to date (version: 57, sigs: 4216790, f-level: 60, builder: mishh  
ammer)  
daily.cvd is up to date (version: 21862, sigs: 394456, f-level: 63, builder: neo  
)  
bytecode.cvd is up to date (version: 283, sigs: 53, f-level: 63, builder: neo)  
root@raspberrypi: ~# clamscan --infected --remove --recursive  
SCAN SUMMARY  
Known viruses: 4607906  
Engine version: 0.99.2  
Scanned directories: 264  
Scanned files: 2063  
Infected files: 0  
Data scanned: 61.31 MB  
Data read: 49.02 MB (ratio 1.25:1)  
Time: 71.844 sec (1 m 11 s)  
root@raspberrypi: ~#
```

手動でスキャン

AWS IoT開発

①. メニュー

- AWS設定

AWS IoTを実施するためのAWSの設定
AWSの詳細説明は以下のURLへ

https://aws.amazon.com/jp/cloud/?nc2=h_l2_cc

- AWS IoT設定

AWS IoTのデバイス、証明書、ポリシー、ルールなどの設定を行います。
開発者ガイドは以下のURLへ

https://docs.aws.amazon.com/ja_jp/iot/latest/developerguide/what-is-aws-iot.html

- 温湿度ロガー

BMP280センサを使いMQTTプロトコルを介して、AWS IoTに接続し、AWS DynamoDBにデータを保存し、そのデータを外部ツールでリアルタイム表示します。またその過程の、Raspberry Piでのセンサ単体試験、AWS IoT折り返し試験などのプログラムも提供。順序だてて構築して行きます。

(参考)

- Githubなど

- <https://github.com/aws/aws-iot-device-sdk-python3> (一部のプログラムはTLSエラーで動きません)
- <https://github.com/eclipse/paho.mqtt.python3>

- MQTT

- <https://www.ibm.com/developerworks/jp/iot/library/iot-mqtt-why-good-for-iot/> (IBMの資料でわかりやすい)

AWS IoT開発

②. AWS設定

- 手順1: アカウントの作成

- <https://aws.amazon.com/jp/register-flow/>
- 必要なメールアドレス、パスワードなどを入力します。



AWS アカウント作成の流れ

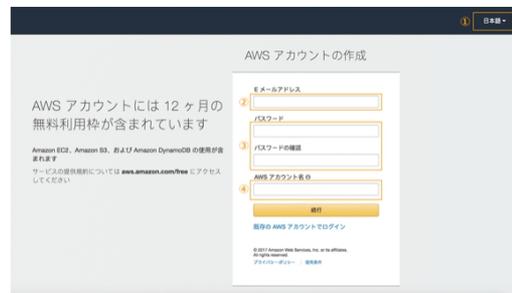
AWS アカウントを作成すると、1年間の無料利用枠はもちろん、AWS クラウドの世界中のリージョンで提供されるすべてのサービスを始めることができます。こちらでは日本のお客様に AWS アカウント作成におけるポイントをご紹介します。

今すぐクラウドを無料でお試ください

[今すぐ無料アカウント作成 >](#)

12 か月間の AWS 無料利用枠と、24 時間年中無休のカスタマーサービスやサポートフォーラムなどの AWS の基本的なサポート機能を利用できます。

ステップ 1: AWS アカウントの作成



※クリックすると大きな画像でご覧いただけます。

このページの上部タイトルおよび、末尾に設置されているオレンジ色のアカウント作成ボタンよりサインアップ画面へ移動します。

各ページ右上 ① の言語選択ボックスより、「日本語」でない場合「日本語」を選択後、こちらのサインアップ画面へお進みください。

最初に AWS アカウントとなる情報を設定します。

- ② の「E メールアドレス」には、AWS へのログインに利用したいメールアドレスを設定します。(※)
- ③ の「パスワード」および「パスワードの確認」で AWS へのログイン時に使用するパスワードを設定し、さらに確認用にもう一度同じパスワードを入力します。
- ④ の「AWS アカウント名」テキストボックスに、お客様のお名前を半角アルファベットで入力します。
- 入力後、「続行」ボタンをクリックします。

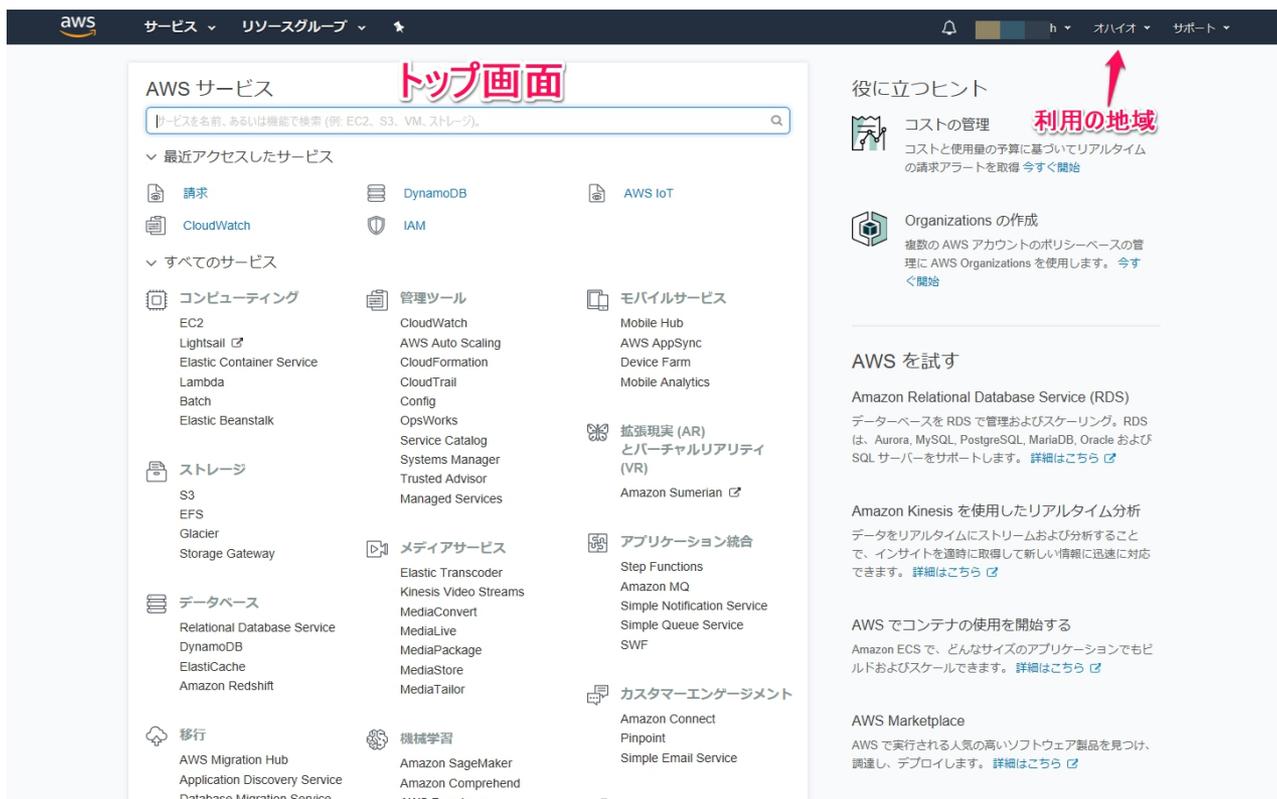
※登録いただくメールアドレスは、AWS 側からの通知等にも利用されます。複数の方へ

AWS IoT開発

②. AWS設定

- AWSトップ画面

- 利用する場合に、地域を意識して設定してください。地域毎に料金が変わったり、利用できるサービスが限定されている場合があります。



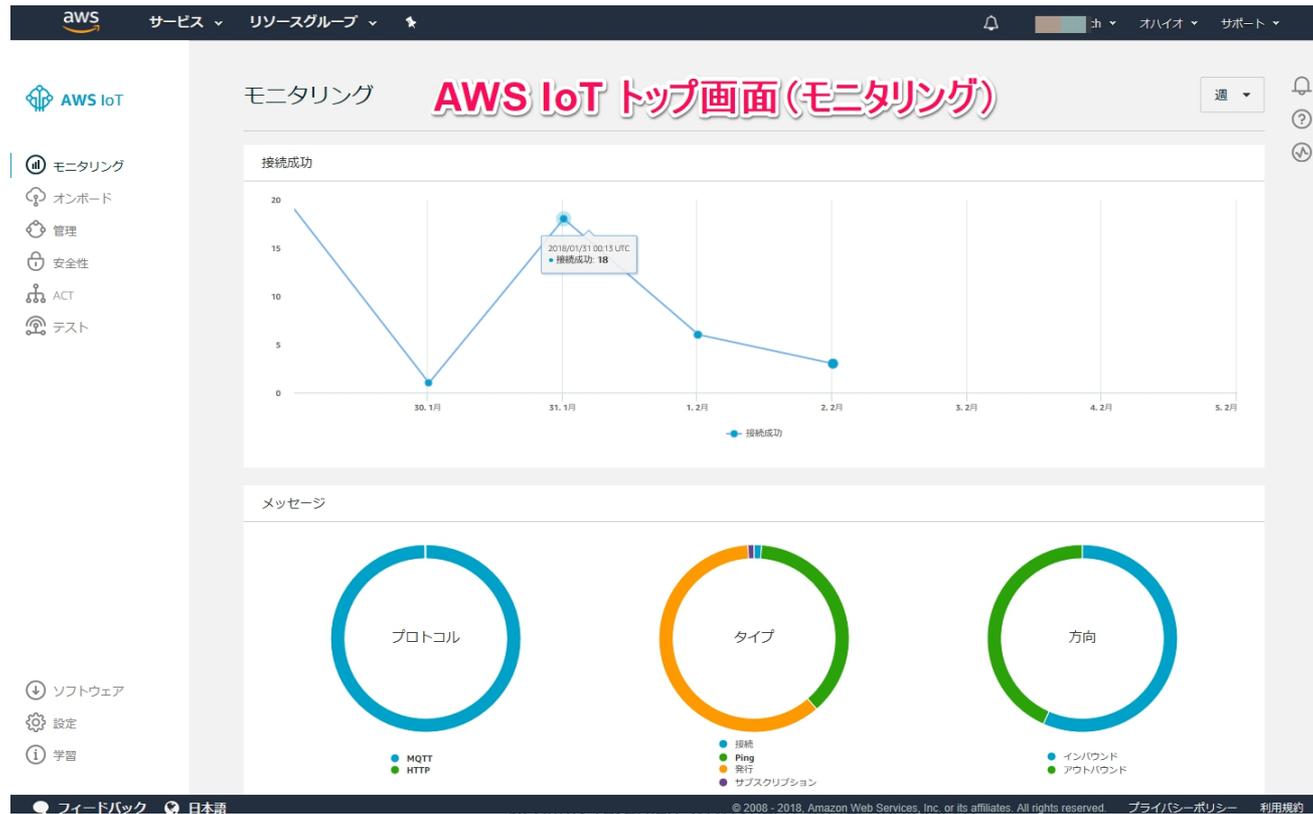


AWS IoT開発

③. AWS IoT設定

● 画面説明1

- AWSTップ画面からAWS IoTのサービスを選択します。
- モニタリング: 接続状況がひとめで分かります

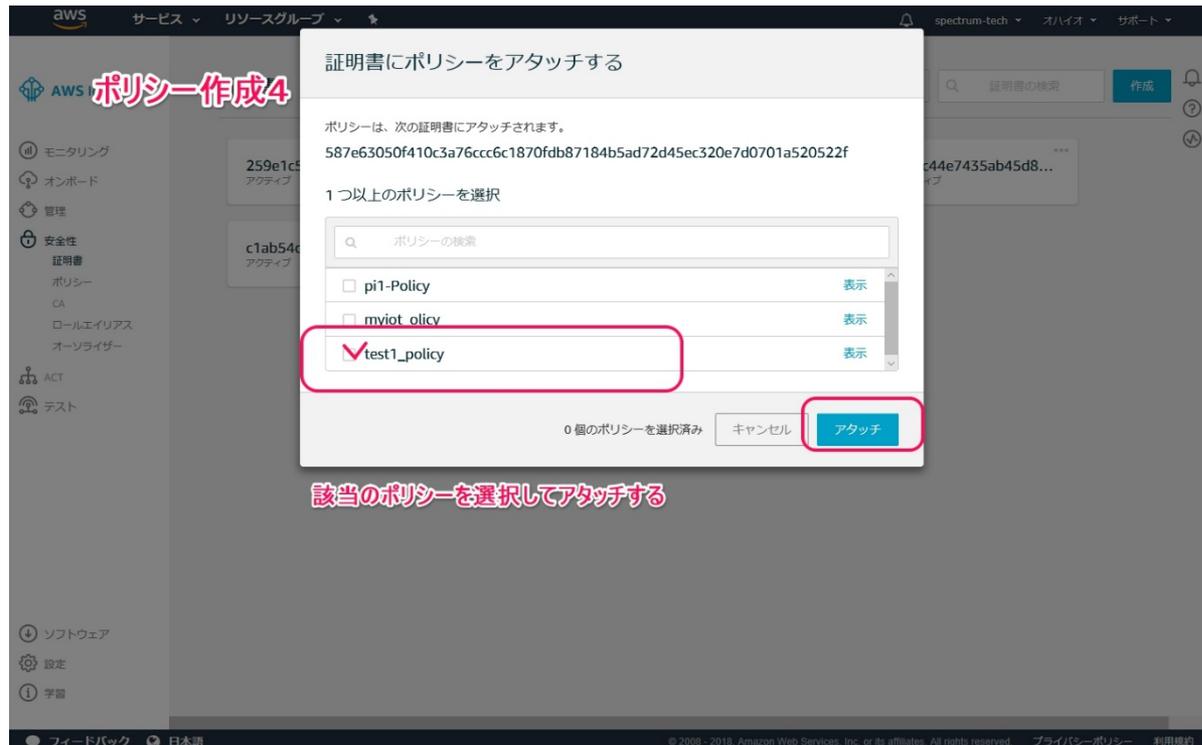


AWS IoT開発

③. AWS IoT設定

• ポリシー作成4

- ポップアップしたポリシーの一覧から該当のポリシーを選んでアタッチします。



証明書にポリシーをアタッチする

ポリシーは、次の証明書にアタッチされます。
587e63050f410c3a76ccc6c1870fdb87184b5ad72d45ec320e7d0701a520522f

1つ以上のポリシーを選択

ポリシーの検索

- pi1-Policy 表示
- mviot_policy 表示
- test1_policy 表示

0個のポリシーを選択済み キャンセル アタッチ

ポリシー作成4

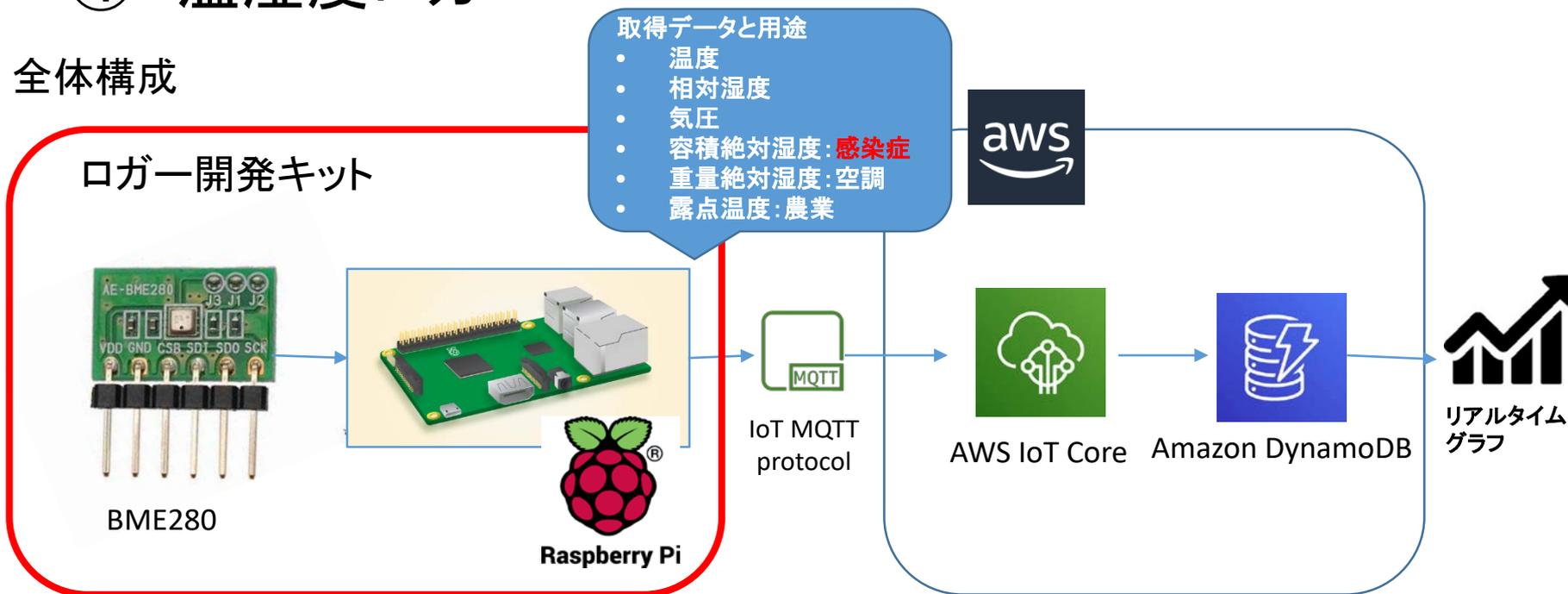
該当のポリシーを選択してアタッチする



AWS IoT開発

④ 温湿度ロガー

全体構成



BME280センサーの気温、湿度、気圧、絶対湿度等データをAWS IoTに送信し、DynamoDBに確保します。DynamoDBの値を外部からリアルタイムのグラフ表示します。またDynamoDBを温湿度ロガーとしてデータを保存します。

AWS IoT開発

④ 温湿度ロガー

取得データ一覧

BME280から取得したデータと算出したデータ一覧です。10秒毎にdynamoDBに格納します。

取得データ名	値(例)	内容	利用用途
count	18806	カウンタ	
Timestamp	2020/10/22 15:52:55	取得時刻	
deviceid	bme280	デバイス名	dynamoDBのテーブル内での装置区分に利用します。
Temp(C)	22.840765323402593	温度	
Humid(%)	48.499156482511886	相対湿度	
Pressure(hPa)	1005.34559519194	気圧	
volumetric hum(g/m3)	9.892641569438664	容積絶対湿度	計算式、医療系に利用
mix ratio(g/kg)	8.359050575661259	重量絶対湿度	計算式、空調、建築系で利用
dew point(C)	11.415924788405619	露点温度	計算式、農業で利用

AWS IoT開発

④ 温湿度ロガー

容積絶対湿度

- 容積絶対湿度(volumetric humidity)について

大気中に含まれる水蒸気の密度(容積あたりの質量)である。単位はグラム毎立方メートル(g/m^3)が用いられている。

医療系で使用される。

- 容積絶対湿度計算式

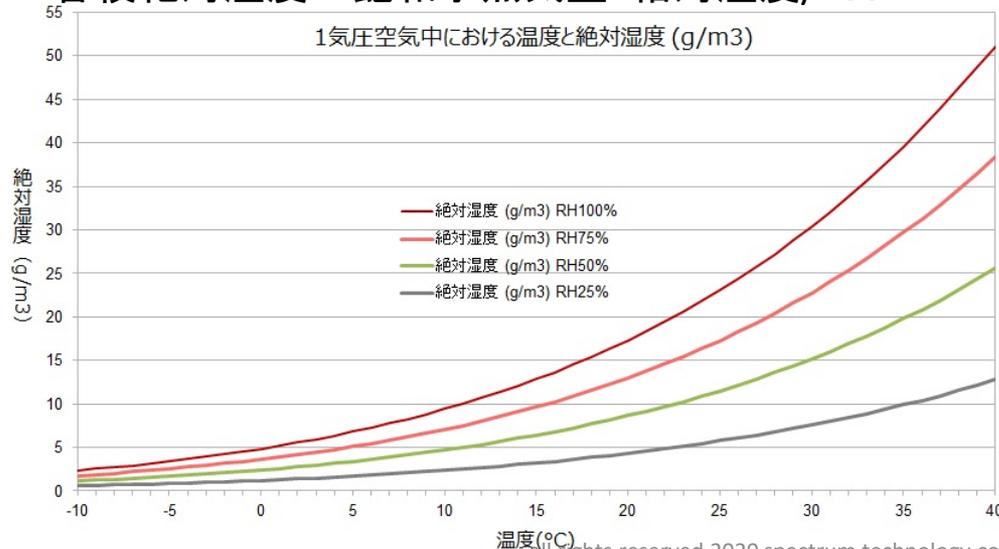
温度と相対湿度から算出します。以下の計算式サイトを参照

https://www.shinyei.co.jp/stc/service/water_converter.html

飽和水蒸気圧 = $6.1078 \cdot 10^{\{7.5 \cdot \text{温度} / (\text{温度} + 237.3)\}}$

飽和水蒸気量 = $217 \cdot \text{飽和水蒸気圧} / (\text{温度} + 273.15)$

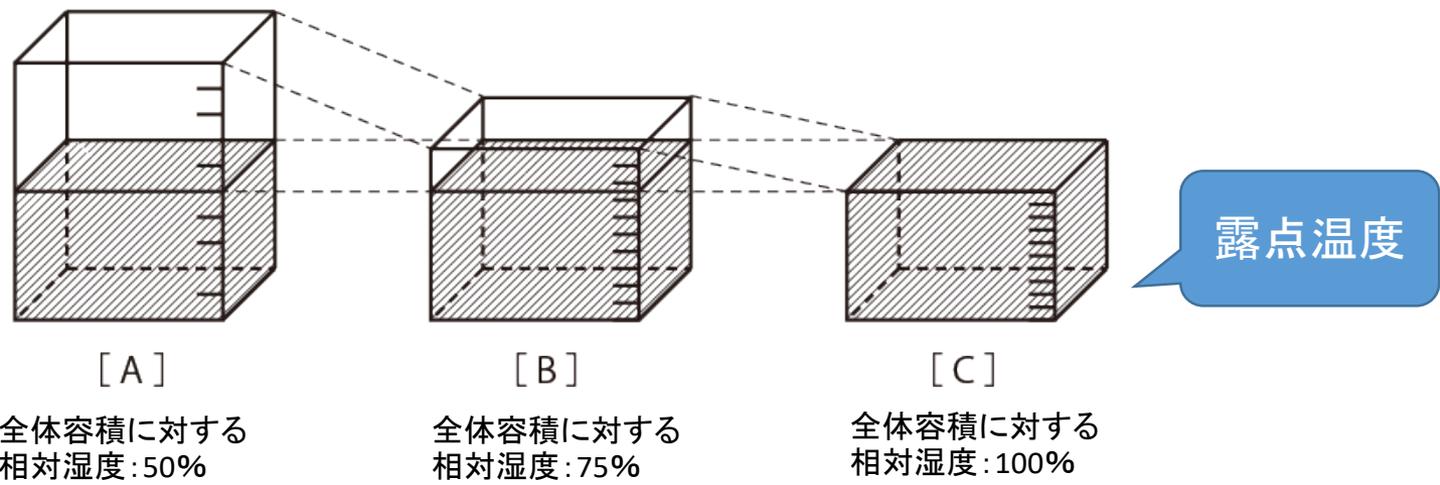
容積絶対湿度 = $\text{飽和水蒸気量} \cdot \text{相対湿度} / 100$



AWS IoT開発

④ 温湿度ロガー 露点温度

- 露点温度(dew point)について
 - 露点温度は、水蒸気を含む空気を冷却した時に凝結し始める温度です。つまり空気中の水蒸気分圧が飽和水蒸気圧となる時の温度です。
 - 露点温度が高いと空気が湿っている。低いと乾燥している状態。特に農業では重要。
- 露点温度計算式
 - 温度と相対湿度から算出します。
 - 飽和水蒸気圧 = $6.1078 \cdot 10^{\{7.5 \cdot \text{温度} / (\text{温度} + 237.3)\}}$
 - 水蒸気圧 = 飽和水蒸気圧 * 相対湿度 / 100
 - 露点温度 = $237.3 \cdot \log_{10}(6.1078 / \text{水蒸気圧}) / (\log_{10}(\text{水蒸気圧} / 6.1078) - 7.5)$



AWS IoT開発

④ 温湿度ロガー 絶対湿度と感染症の関係

- 絶対湿度について

大気に含まれる水蒸気の量を質量で表すものである。単位あたりの水蒸気量が質量(gあるいはkg)で表される。1m³の空気の中に水蒸気が質量として何gあるかを実際の量として表現するものである。

なお、一般的に使われている相対湿度は、その気温の空気が水分を保有できる最大量に対してどれだけの水分があるかどうかを割合で表現したものである。

<https://komoriss.com/relative-humidity-volumetric-humidity/>

- 本キットでは、以下の計算式で温湿度から絶対湿度を算出しています。

絶対湿度

$$=217*(6.1078*10^{(7.5*t/(t+237.3))})/(t+273.15)*RH/100$$

t:気温、RH:相対湿度

NW ウェザーニュース

絶対湿度(g/m³)早見表

危険 注意		気温															
		10°C	11°C	12°C	13°C	14°C	15°C	16°C	17°C	18°C	19°C	20°C	21°C	22°C	23°C	24°C	25°C
相 対 湿 度	100%	9.4	10.0	10.7	11.7	12.1	12.8	13.6	14.5	15.4	16.3	17.3	18.3	19.4	20.6	21.8	23.0
	95%	8.9	9.5	10.2	11.1	11.5	12.2	12.9	13.8	14.6	15.5	16.4	17.4	18.4	19.6	20.7	21.9
	90%	8.5	9.0	9.6	10.5	10.9	11.5	12.2	13.1	13.9	14.7	15.6	16.5	17.5	18.5	19.6	20.7
	85%	8.0	8.5	9.1	9.9	10.3	10.9	11.6	12.3	13.1	13.9	14.7	15.6	16.5	17.5	18.5	19.6
	80%	7.5	8.0	8.6	9.4	9.7	10.2	10.9	11.6	12.3	13.0	13.8	14.6	15.5	16.5	17.4	18.4
	75%	7.1	7.5	8.0	8.8	9.1	9.6	10.2	10.9	11.6	12.2	13.0	13.7	14.6	15.5	16.4	17.3
	70%	6.6	7.0	7.5	8.2	8.5	9.0	9.5	10.2	10.8	11.4	12.1	12.8	13.6	14.4	15.3	16.1
	65%	6.1	6.5	7.0	7.6	7.9	8.3	8.8	9.4	10.0	10.6	11.2	11.9	12.6	13.4	14.2	15.0
	60%	5.6	6.0	6.4	7.0	7.3	7.7	8.2	8.7	9.2	9.8	10.4	11.0	11.6	12.4	13.1	13.8
	55%	5.2	5.5	5.9	6.4	6.7	7.0	7.5	8.0	8.5	9.0	9.5	10.1	10.7	11.3	12.0	12.7
	50%	4.7	5.0	5.4	5.9	6.1	6.4	6.8	7.3	7.7	8.2	8.7	9.2	9.7	10.3	10.9	11.5
	45%	4.2	4.5	4.8	5.3	5.4	5.8	6.1	6.5	6.9	7.3	7.8	8.2	8.7	9.3	9.8	10.4
40%	3.8	4.0	4.3	4.7	4.8	5.1	5.4	5.8	6.2	6.5	6.9	7.3	7.8	8.2	8.7	9.2	
35%	3.3	3.5	3.7	4.1	4.2	4.5	4.8	5.1	5.4	5.7	6.1	6.4	6.8	7.2	7.6	8.1	
30%	2.8	3.0	3.2	3.5	3.6	3.8	4.1	4.4	4.6	4.9	5.2	5.5	5.8	6.2	6.5	6.9	
25%	2.4	2.5	2.7	2.9	3.0	3.2	3.4	3.6	3.9	4.1	4.3	4.6	4.9	5.2	5.5	5.8	
20%	1.9	2.0	2.1	2.3	2.4	2.6	2.7	2.9	3.1	3.3	3.5	3.7	3.9	4.1	4.4	4.6	

宮城県医師会より

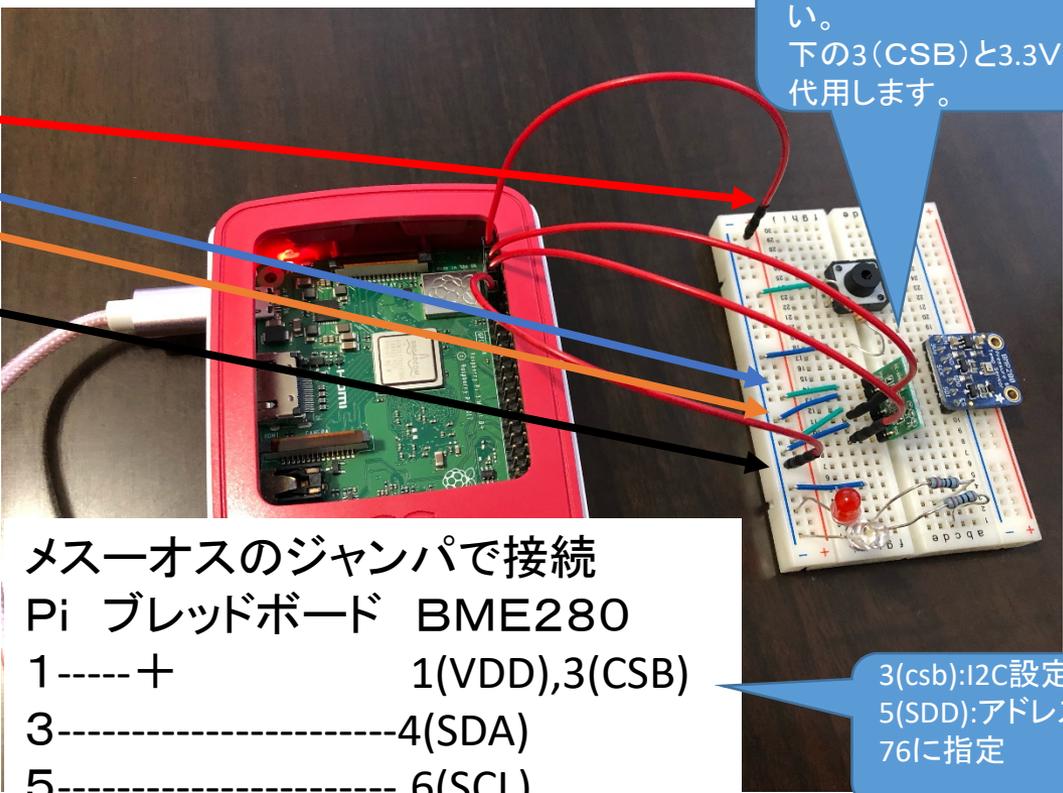
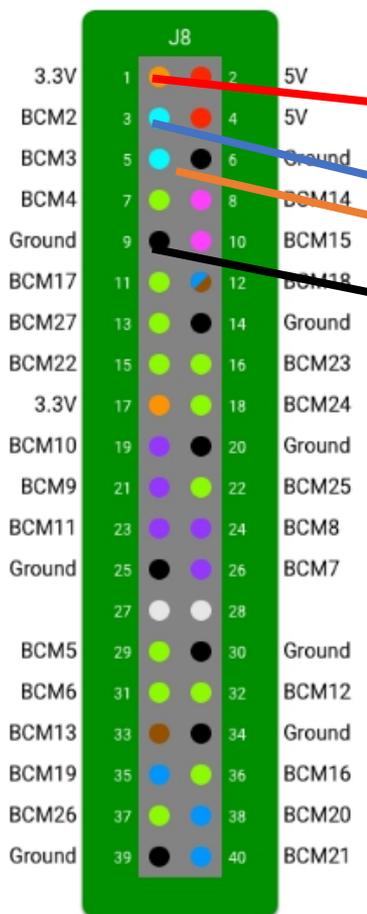
絶対湿度とインフルエンザの関係
 ~7g/m³ : より起こりやすい: 危険
 7~11g/m³ : 流行しやすい: 注意
 11~g/m³ : 流行しにくい: 快適



AWS IoT開発

④ 温湿度ロガー

Raspberry PiとBME280の接続



BMEを接続するためにハンダでPinを付けます。J3のハンダは実施しないでください。下の3(CSB)と3.3V接続で代用します。

メスオスのジャンプで接続
 Pi ブレッドボード BME280
 1-----+ 1(VDD),3(CSB)
 3-----4(SDA)
 5-----6(SCL)
 9-----2(GND), 5(SDD)

3(csb):I2C設定
 5(SDD):アドレスを0x76に指定



AWS IoT開発

④ 温湿度ロガー

```
$ cd /home/pi/Documents/paho2  
$ python3 bme280_sample.py
```

BME280単体試験

- Piコンソールでbme280_sample.pyを動作させます。
- 気温、気圧、湿度が出力されます。
- 出力されない場合は、I2Cの設定を確認。0x76に値が表示されれば正常。

\$ i2cdetect -y 1

```
pi@raspberrypi: ~/Documents/paho2  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
pi@raspberrypi:~/Documents/paho2/BME280/Python27 $ python bme280_sample.py  
temp : 21.27 °C  
pressure : 994.02 hPa  
hum : 48.75 %  
pi@raspberrypi:~/Documents/paho2/BME280/Python27 $ cd /home/pi/Documents/paho2  
pi@raspberrypi:~/Documents/paho2 $ python bme280_sample.py  
temp : 21.13 °C  
pressure : 994.07 hPa  
hum : 49.31 %  
pi@raspberrypi:~/Documents/paho2 $ i2cdetect -y 1  
0 1 2 3 4 5 6 7 8 9 a b c d e f  
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
70: -- -- -- -- -- 76 -- -- -- -- -- -- -- -- -- --  
pi@raspberrypi:~/Documents/paho2 $ python bme280_sample.py  
temp : 20.93 °C  
pressure : 993.94 hPa  
hum : 49.84 %  
pi@raspberrypi:~/Documents/paho2 $
```

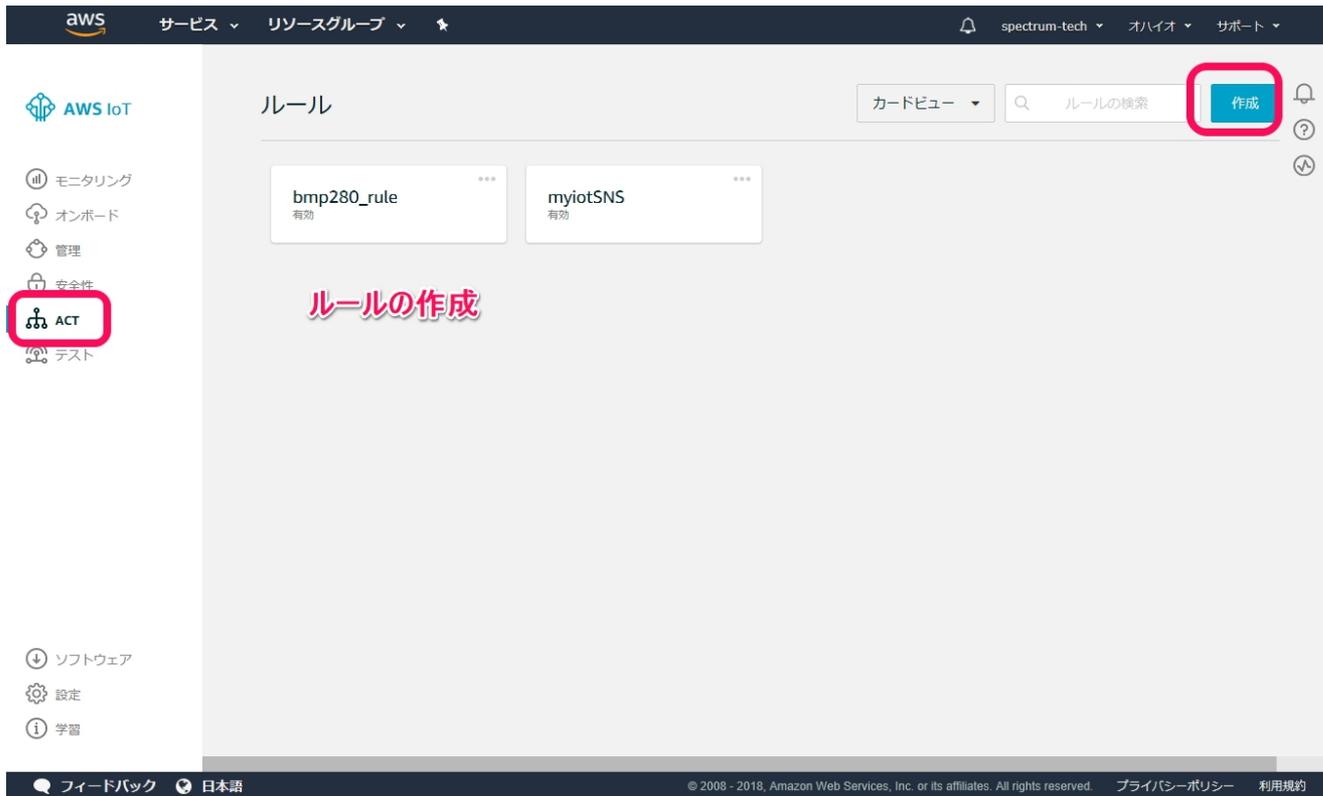


AWS IoT開発

④ 温湿度ロガー

AWS IoTのルール設定

- AWS IoTのコンソールからACT>ルール
- ルール作成を押します。(メール送信と同様)





AWS IoT開発

④ 温湿度ロガー

mqtt_subscribe_json.pyのプログラム設定

- Windows PCにさくらエディタなどをインストール
- Windowsネットワークからrasberry piをクリックしてpi>documents>paho2
 - mqtt_subscribe_json.pyをwindowsのローカルにコピーします。
 - エディタでローカルのmqtt_subscribe_json.pyを開きます。
 - 電子証明書などの情報を設定します。5-14行

MQTT_TOPIC = "pi3" aws iotルールの中のトピック

MQTT_HOST = "xxx.amazonaws.com"エンドポイントの値を入れます

CA_ROOT_CERT_FILE = "root-CA.crt"

THING_CERT_FILE = "xxx-certificate.pem.crt"電子証明書番号

THING_PRIVATE_KEY = "xxx-private.pem.key"電子証明書番号

```

1 # Import package
2 import paho.mqtt.client as mqtt
3 import ssl
4
5 # Define Variables
6 MQTT_PORT = 8883
7 MQTT_KEEPALIVE_INTERVAL = 45
8 MQTT_TOPIC = "pi1"
9 MQTT_MSG = "hello MQTT"
10
11 MQTT_HOST = "a[redacted].amazonaws.com"
12 CA_ROOT_CERT_FILE = "root-CA.crt"
13 THING_CERT_FILE = ".cert.pem"
14 THING_PRIVATE_KEY = ".private.key"
15
16 # Define on connect event function
17 # We shall subscribe to our Topic in this function
18 def on_connect(self, mosq, obj, rc):
19     mqttc.subscribe(MQTT_TOPIC, 0)
20
21 # Define on message event function
22 # This function will be invoked every time,
23 # a new message arrives for the subscribed topic
24 def on_message(mosq, obj, msg):
25     print "Topic: " + str(msg.topic)
26     print "QoS: " + str(msg.qos)
27     print "Payload: " + str(msg.payload)
28
29 def on_subscribe(mosq, obj, mid, granted_qos):
30     print("Subscribed to Topic: " +
31           MQTT_TOPIC + " with QoS: " + str(granted_qos))
32
33 # Initiate MQTT Client
34 mqttc = mqtt.Client()
35
36 # Assign event callbacks
37 mqttc.on_message = on_message
38 mqttc.on_connect = on_connect
39 mqttc.on_subscribe = on_subscribe
40
41 # Configure TLS Set

```

秘密鍵のペアである公開鍵は使用しません

- 証明書をフォルダにアップロード
 - 最初にダウンロードした証明書のうち
 - root-CA.crt, xxx-certification.pem.crt,
 - Xxx-private.pem.keyの3個を
 - pi>documents>pahoのフォルダにアップロード
- 書き換えたmqtt_subscribe_json.pyをアップロード



AWS IoT開発

④ 温湿度ロガー

AWS DynamoDBでの確認

- テーブル > 設定したテーブル > 項目
- やりとりのデータが入っていればOK

The screenshot shows the AWS IoT console interface. On the left, the 'DynamoDB' section is expanded, and 'テーブル' (Tables) is selected. Under 'テーブル', the 'pi2_db' table is highlighted. The main area shows the '項目' (Items) tab for the 'pi2_db' table. The table has columns for 'deviceid', 'timestamp', 'Humid(%)', and 'Pressure(hPa)'. The data rows show multiple entries for 'bme280' with timestamps ranging from 2019/11/12 09:25:02 to 2019/11/12 09:26:43. The humidity values are consistently 34.74073020266086, and the pressure values are consistently 1000.315714241029.

deviceid	timestamp	Humid(%)	Pressure(hPa)
bme280	2019/11/12 09:25:02	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:25:12	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:25:22	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:25:32	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:25:42	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:25:52	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:26:02	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:26:13	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:26:23	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:26:33	34.74073020266086	1000.315714241029
bme280	2019/11/12 09:26:43	34.74073020266086	1000.315714241029



AWS IoT開発

④ 温湿度ロガー

リアルタイム表示

- IoT. Kyoto様の利用>グラフ
- Deviceid:bme280 (python3で書き込み済)
- 表示対象をチェックすると自動で表示されます。





AWS IoT開発

④ 温湿度ロガー

DynamoDBのCSVエクスポート

- コマンドを入力し、抽出します。便利です。
- -t の後ろはテーブル名
- >の後ろは、出力ファイル名

```
$ cd /home/pi/Documents/DynamoDBtoCSV
$ node dynamoDBtoCSV.js -t pi2_db > output.csv
```

