

高柔軟度 IoT の環境構築

[a2ty.dev@gmail.com]

A2TY

<https://github.com/A2TY/module-IoT>

2016年2月25日

概要

今日では、IoT の開発が進み、多くの IoT 製品を見かけるようになった。一例を上げれば、ソニーと Wil がスマートロック事業として合弁会社を興した「Qrio」のスマートキー やソニーの開発した様々なセンサを搭載し、インターネットに接続することにより赤外線リモコンとして動作することができるシーリングライトなどがある。確かに、これらの製品は便利であり生活を豊かにするものであるが、これらの製品に共通して言えることは、初期設定後もスマートフォンや PC 等による人の操作が必要になるてんである。また、様々なセンサを搭載し高性能であるにも関わらず、他の機器との連携が不十分であり、ものがインターネットに繋がることを十分に活かせず、ユーザはそれを利用するしかないという現状となっている。そこで、本プロジェクトでは、ネットワークに繋がるセンサおよび出力部分を個々に分けそれぞれを好みの場所に設置できるようにした。これらをユーザによって「レシピ」というかたちでセンサによる反応に応じて出力部分の機能を自由に設定できる様にした。

1 IoT の現状

IoT とは、Internet of Things つまりモノのインターネットであり、「もの」がインターネット / クラウドに接続され情報交換を行うことにより相互に制御する仕組みのことである。この言葉は、1999 年にケビン・アシュトンというイギリス人技術者によって提唱された。IoT という言葉は、近年よく耳にする様になり、スマートフォンの普及やウェアラブル端末の流行から、多くの IoT 製品が世の中に浸透してきたことが伺える。また、図 1 に示した様に、世界の検索キーワードの傾向を調べることができる Google トレンドにおける「IoT」の傾向検索結果からも近年急激に IoT が話題になっていることが分かる。

こうした IoT において、データの収集窓口となるセンサは最も重要な要素である。2014 年には、Intel によってモーションセンサモジュール「Intel RealSense 3D カメラ」を発表するなど今まで把握することが困難であった指先のジェスチャー や顔認証が可能となり、より人間の動作や感情の把握ができるようになった。こうしたことから、収集可能なデータの種類や量を爆発的に増やすことができ、ビッグデータ活用の実用性や実現性が高まることが期待できる。

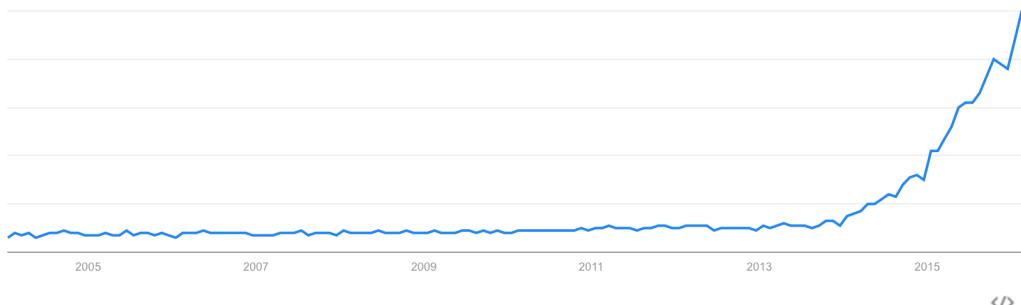


図 1 Google トレンドによる「IoT」の検索結果

ここで身近に出回っている IoT 製品について考える。IoT 製品の一例を上げると、ソニーの開発した様々なセンサを搭載し、インターネットに接続し、TV リモコンや音楽スピーカの機能を備えた多機能 LED シーリングライトやインターネットに繋がり健康管理を行える体重計である「FitBit Aria」などがある。こうした製品は、生活に溶け込み便利な製品であるが、初期設定以外にスマートフォンや PC を使うことが多く、「もの」が情報を収集し提供しているが最終的に「人間」がスマートフォンや PC を介して情報を取りに行くといった現状となっている。また、情報収集を行うセンサが一箇所に固定・集中してしまい提供できるサービスが限られるとともに、ユーザのニーズが限定されてしまっていると言える。

こうした、IoT 製品の現状を改善するため、本プロジェクトを企画した。以下、システム構成および実際の使用例を述べる。

2 環境構築するにあたって

本プロジェクトにおける柔軟度の高い IoT 環境を様々な通知方法を持ち、情報を様々な場所から収集することができ、ユーザによって機能を自由に設定できる環境とする。

2.1 I/O のあり方

柔軟度の高い IoT 環境を構築するためには、「インプット」いわゆる情報収集を行うセンサと、「アウトプット」いわゆる光ったり音を鳴らしたりなどの動作部分が個々に別れることが必要である。また、個々に分けることにより 1 つ当たりの電流消費量が小さくなるためモバイルバッテリなどの持ち運びができる小さな電源でも運用が可能となる。このように電源に制限を受けること無く設置ができることから更なる活用の幅が広がると言える。

2.2 デバイスプロトコル

デバイスプロトコルは大きく分けて有線と無線に分かれる。有線プロトコルは利便性よりも信頼性が求まられる場合に使用され、無線プロトコルは信頼性よりも利便性が求められる場合に使用される。今回は、利便性を優先すべきであるため、無線プロトコルを用いることにする。しかし、無線プロトコルと言っても数多く存在する。そこで、いくつか無線プロトコルを紹介するとともに使用するプロトコルを決定する。

■ BLE (Bluetooth Low Energy)

BLE (Bluetooth Low Energy) とは、Bluetooth の仕様における、バージョン 4.0 の呼称である。バージョン 3.0 の Bluetooth と比較して、省電力かつ省コストで行うことができる。現在では、多くの PC およびスマートフォンに対応し、広く普及している。

今回の場合で考えると、省電力かつ広く普及していることから非常に IoT の環境に適していると言える。しかし、BLE は、無線 LAN ルータの様なインターネットとの間を仲介してくれる機器が PC やスマートフォンなどしか無いため設置場所が限定されてしまう。

■ Dust Networks / IEEE 802.15.4

Dust Networks は、2.4GHz 帯を使う無線メッシュネットワークである。電池駆動でメッシュネットワークを簡単に組むことができ、低消費電力で「切れない無線」として注目されている。日本国内ではまだ実績が無いが、海外では徐々に普及している。

今回の場合で考えると、電池で駆動できるほどの低消費電力かつ非常に切れにくいことから信頼性も高いと言える。しかし、日本国内での実績が無いことからこのプロトコルを用いるのは難しいと言える。

■ ZigBee / IEEE 802.15.4

ZigBee は、2.4GHz 帯を用いた無線メッシュネットワークが可能である。ZigBee のスリープ時の待機電力は非常に小さく、その上、復帰からデータ送信までに要する時間も数十 ms 程度と非常に短い。しかし、省電力性を活かすためには 0 た送信間隔が十分に空いていることが条件となる。また、スリープ状態にある相手デバイスを起こすことが出来ないため、スリープ状態にあるデバイスへデータを送ることが出来ない。

今回の場合で考えると、非同期的に情報を送り合うためセンサデータを扱う場面では向かないと考えられる。

■ Wi-Fi / IEEE 802.11 b/g/n a/n/ac

Wi-Fi は、2.4GHz 帯を用いる b/g/n と 5GHz 帯を用いる a/n/ac に分けられる。世界的に普及し、多くの家庭や公共機関で利用されている。高速で大容量通信を行うことができるが、先に説明したプロトコルを比べると消費電力は大きい。

今回の場合で考えると、消費電力はそれほど低くないが、広く普及したデバイスプロトコルであるため設置場所が限定されることが少ないと見える。また、インターネットへの接続が容易であることからシステムの構築がし易いと言える。

こうしたことから、広く普及しインターネットへの接続が容易である Wi-Fi を柔軟度の高い IoT 環境のデバイスプロトコルとして採用することとする。

2.3 ゲートウェイプロトコル

次に、使用するゲートウェイプロトコルについて幾つか例を上げて決めていく。

■ HTTP/REST

HTTP (Hypertext Transfer Protocol) は、Web ブラウザと Web サーバの間で HTML などのコンテンツを送受信するのに用いられる通信プロトコルである。クラウド側の通常の Web 技術が使用できるため構築が容易に行える。しかし、センサデータの送信は容易だがデータの受信が難しいため双方向通信には向かない。

今回の場合で考えると、デバイスは、送信だけでなく受信も行うため向かないと言える。また、HTTP 通信によって送られて来る情報には、ユーザエージェント名やリファラなどが記述された HTTP ヘッダが 50byte 以上含まれる。このように、送られてくる情報が多いとそれだけ電力を消費してしまいモバイルバッテリなどでセンサを稼働させる場合大きな障害となる。

■ CoAP (Constrained Application Protocol)

CoAP (Constrained Application Protocol) は簡易 HTTP を目指したプロトコルである。ヘッダが 4byte と小さく、UDP ベースであるため信頼性は低くなるが再送制御が無いためレスポンスがシンプルとなる。

今回の場合で考えると、ヘッダが小さく消費電力を抑えることができるが、HTTP 通信と同様に外部から受信する場合に少し工夫が必要になるといえる。また、無線通信を用いることで信頼性が低くなっていることを考えると UDP を用いることでさらに信頼性が低くなるのは避けたい。

■ MQTT (Message Queue Telemetry Transport)

MQTT (Message Queue Telemetry Transport) は、アプリケーション層で使用される、TCP/IP による Pub/Sub 型の軽量なメッセージキュー プロトコルである。非力なデバイスやネットワークが不安定な場所でも動作しやすい様にメッセージが軽量に設計されている。メッセージ ローカルを設置する必要がある。ローカルは、メッセージのトピックに基いて、それを必要としているクライアントにメッセージを配信する。クライアント側のグローバル IP を必要としないため受信も容易に行うことができる。また、接続を確立し続けるので、通信がほぼリアルタイムに行うことができる。

今回の場合で考えると、通信が軽量であることから消費電力を抑えることができるといえる。また、ローカルを用いることにより、クライアントの送受信が容易に行うことができ環境構築がし易いと言える。また、リアルタイムに通信を行うことができるためセンサで収集した情報を有効に活用できると言える。

こうしたことから、デバイスの消費電力を抑えられセンサデータをリアルタイムに活用することのできる MQTT をゲートウェイ プロトコルとして採用することとする。

3 実際に構築した環境

今まで述べてきたデバイスプロトコルおよびゲートウェイプロトコルを用いて図2の様なIoT環境を構築した。サーバとクライアントである個々に別れたセンサとアウトプットデバイスはMQTTのPub/Sub型の通信によってMQTTブローカと通信を行っており、このような個々に別れたデバイスの通信に機能を持たせるのがユーザが自由に設定することができるレシピである。レシピには、どのセンサが反応した時にどのアウトプットデバイスをどのように動作させるかを設定することができる。そして、設定されたレシピをレシピ実行管理システムによってMQTTブローカに送られてくるデータを処理し、設定されたアウトプットデバイスが動作するようにMQTTブローカにメッセージをPubする。つまり、このIoT環境はユーザが自由に機能を持たせることができ、さらにユーザ自らが情報処理に関与することなく目的の機能を実現する事が出来るのである。このIoT環境について、詳しく述べていく。

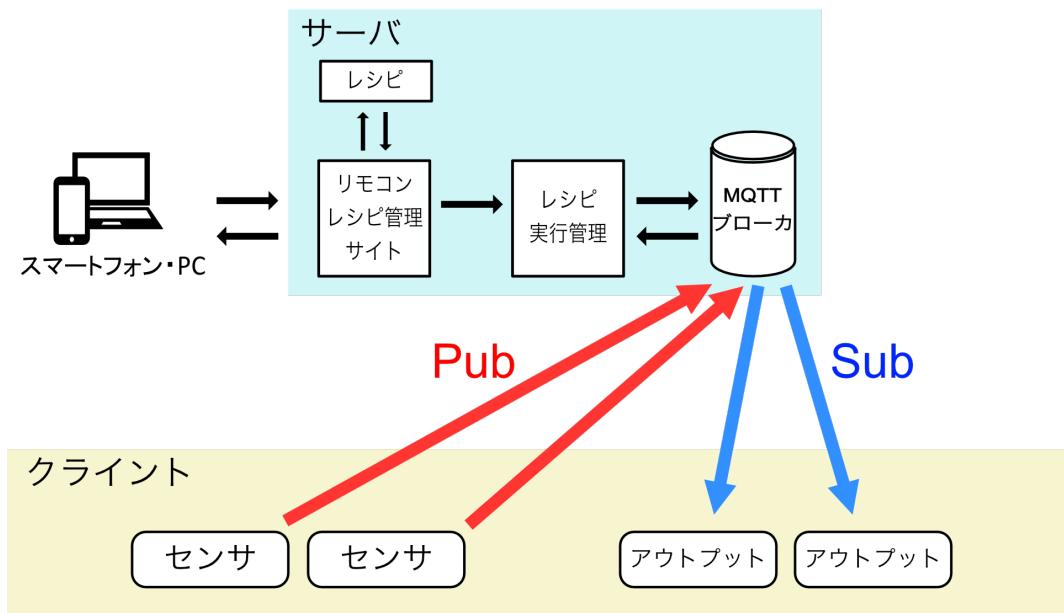


図2 システム全体図

3.1 レシピ

レシピは、個々に別れたセンサとアウトプットデバイスを繋ぎ合わせ機能を持たせる命令書である。レシピには、どのセンサがどういったあたいを送ってきたときに、どのアウトプットデバイスをどのように動作させることかを記述することが出来る。これらは、センサの条件を表す「If データ」と、その時どのような動作をする「Then データ」として図 3 の様な json データとして保存される。json データとして保存されるため、プログラムだけでなく、人間である管理者にも理解しやすいデータとなっている。

```
1 {
2   "actionDataIf": "センサの反応",
3   "actionDataThen": "アウトプットデバイスの指定",
4   "recipeId": レシピID,
5   "recipeName": "レシピ名",
6   "valueIf": [
7     [
8       "センサの条件値"
9     ]
10   ],
11   "valueThen": [
12     [
13       "アウトプットデバイスへ送信する値"
14     ]
15   ]
16 }
```

図 3 json データを利用したレシピ

3.2 リモコン・レシピ管理サイト

リモコン・レシピ管理サイトは、2つの機能を持っている。1つ目は、テレビのリモコンとしての機能に即座にアクセスできるようにしたものである。このリモコンのボタンを押すと赤外線信号を送信する事が出来るアウトプットデバイスに MQTT を通して命令することができる。2つ目の機能は、レシピの管理サイトである。この管理サイトは、レシピの作成、編集、削除ができるだけでなく、レシピによる命令を強制的に実行させる機能を備えている。このサイトとスマートフォン・PCとの間は、WebSocket というゲートウェイプロトコルを用いることにより、よりリアルタイムな通信を行える様にした。WebSocket は、HTTP 通信とは異なり、コネクションを貼り続けることができ、TCP 上で双方向通信を容易に行うことが出来る。また、このサイトは、Bootstrap と Web サイトや Web アプリケーションを作成するフリーソフトウェアツール集を利用している。タイポグラフィ、フォーム、ボタン、ナビゲーション、その他構成要素や JavaScript 用拡張などが HTML 及び CSS ベースのデザインテンプレートとして用意されている。グリッドレイアウトレスポンシブデザインが特徴であり、画面幅に応じて最適なレイアウトに設定することが出来る。これにより図 4 の様にレイアウトを自動整形することが出来る。

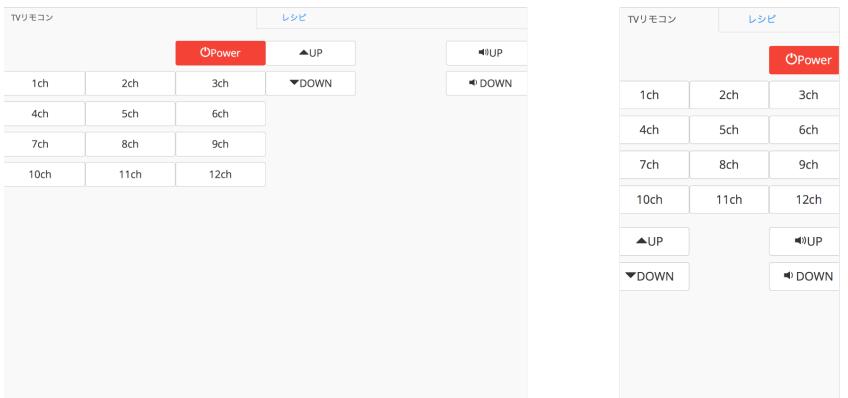


図 4 Bootstrap によるリモコン・レシピ管理サイトのレイアウト自動整形

3.3 レシピ実行管理システム

レシピ実行管理システムは、クライアントから Pub されるメッセージを全て Sub し、全てのセンサデータを解釈している。これらのセンサデータとレシピを照らし合わせ、Sub したセンサデータがレシピに書かれた条件を満たしていた場合、MQTT ブローカにアウトプットデバイスへの値をメッセージとして Pub する。レシピ実行管理システムでは、初めに外部ファイルのレシピデータを読み込み、図 5 の様にレシピの If データと Then データを MQTT のトピックに変換するなどし、処理しやすい形式にしてから処理を行う。

```
19 #レシピデータのIfとThenを日本語からMQTTのtopic名に変換
20 def transformTopicName(recipeDataJson):
21     global recipeDataList
22     topicList = [['人感センサが反応', '温度センサが反応', '湿度センサが反応', '加速度センサが反応', 'スイッチが押される', 'TV
23     に赤外線送信', 'LEDを点灯', 'ブザーを鳴らす'], ['PIRsensor', 'temperature', 'humidity', 'acceleration', 'switch', 'tvIr',
24     'onLED', 'ringSpeaker']]
25
26     for i in range(len(recipeDataJson)):
27         for j in range(len(topicList[0])):
28             if (recipeDataJson[i]["actionDataIf"] == topicList[0][j]):
29                 recipeDataJson[i]["actionDataIf"] = topicList[1][j]
30             if (recipeDataJson[i]["actionDataThen"] == topicList[0][j]):
31                 recipeDataJson[i]["actionDataThen"] = topicList[1][j]
32
33     print (json.dumps(recipeDataJson, sort_keys=True, indent=2, ensure_ascii=False))
34     recipeDataList = recipeDataJson.copy()
35
36
37     return recipeDataList
```

図 5 レシピデータをプログラムで処理しやすい形式に変換

3.4 MQTT トピック

MQTT による Pub および Sub される情報は図 6 の様に MQTT ブローカの IP アドレスとトピック、メッセージによって構成される。トピックは、階層構造をとっている。また、「#」や「+」という記号を使うことによって、複数のトピックを指定することが出来る。例えば、「/client/sensor/#」とした場合、先頭が「/client/sensor/」に該当するトピック全てを指定することができる。また、「/client/+server/」とした場合は、先頭が「/client/」でトピックの最後が「/server/」に該当するトピックを指定することができる。今回構築した環境では、図 7 の様にトピックを構成した。トピックを server と client で分けたのは、発信元がセンサかサーバかを判断するためである。

```
1 mosquitto_pub -h __ADDRESS__ -t "__TOPIC__" -m "__MESSAGE__"
2
3 mosquitto_sub -h __ADDRESS__ -t "__TOPIC__" -v
```

図 6 MQTT による Pub/Sub 通信の例

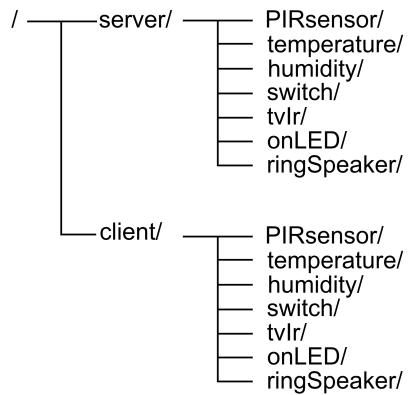


図 7 MQTT トピックの構成図

3.5 センサとアウトプット

クライアントであるセンサとアウトプットデバイスについて説明する。冒頭でも述べたが、柔軟度の高い IoT 環境を作るためにセンサとアウトプットデバイスを個々に分けることにした。これらのデバイスは、ESP-WROOM-02 という Wi-Fi モジュール内蔵の Arduino 互換機を用いる。今回用いたセンサは、人感センサと温度センサ、湿度センサの 3 つである。それに加えて今回は、入力機器としてタクトスイッチを用意した。また、アウトプットデバイスでは、赤外線送信部とフルカラー LED、ブザーの 3 つである。これらについて、1 つずつ詳しく説明していく。

表 1 クライアントデバイスに使用した機器

名称	用途	URL
ESP-WROOM-02 開発ボード	クライアントデバイス	https://www.switch-science.com/catalog/2500/
焦電型赤外線（人感）センサモジュール SB412A	人感センサデバイス	http://akizukidenshi.com/catalog/g/gM-09002/
HDC1000 使用 湿度センサモジュール	温湿度センサデバイス	http://akizukidenshi.com/catalog/g/gM-08775/
5mm 赤外線 LED OSIS5FU5111C-40	赤外線送信デバイス	http://akizukidenshi.com/catalog/g/gI-03261/
RGB フルカラー LED OSTA5131A	フルカラー LED デバイス	http://akizukidenshi.com/catalog/g/gI-03037/
圧電スピーカー（圧電サウンド）SPT08	ブザーデバイス	http://akizukidenshi.com/catalog/g/gP-01251/
FRISK	クライアントデバイスケース	

■人感センサ

人感センサを搭載したセンサデバイスは、人感センサに反応があった場合、MQTT プロトコルにトピック「/client/PIRsensor」でメッセージを Pub する。人感センサデバイスの回路図は図 8 のようになる。このセンサは、1 度反応すると 2 秒程反応した状態が続いてしまう。そこで、図 10 のように大きな値を人感センサが返した時、つまり、人感センサが反応した時に 5 秒間待つことにより解決した。これにより、正しく近くで人が動作したことを検知できるようになった。

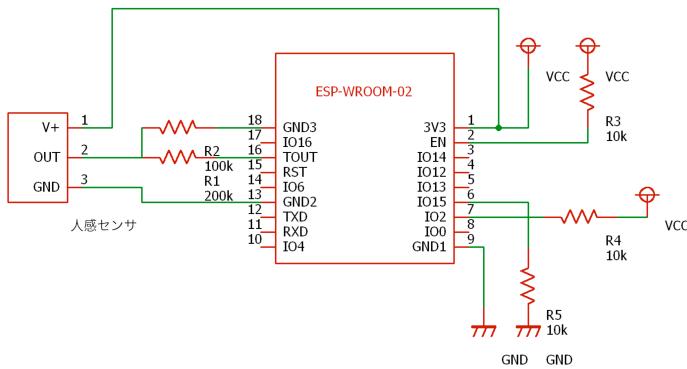


図 8 人感センサデバイスの回路図

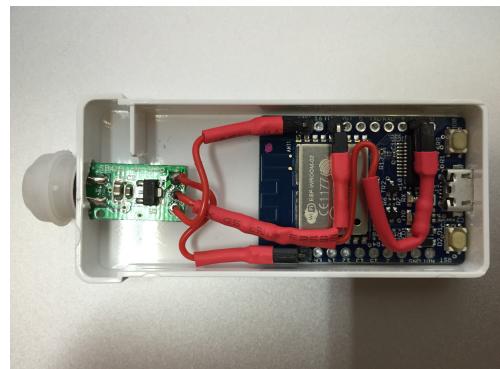


図 9 実際の人感センサデバイス

```

86 //人感センサが大きい値を返したらPubする
87 if (ADC_Value > 512) {
88   PIRsensorPublisher.publish("on");
89   delay(5000); //短時間に何度もPubしないようにする
90 }
91 else {
92   delay(1000);
93 }

```

図 10 人感センサをより正確にするためのプログラム

■温湿度センサ

温湿度センサを搭載したセンサデバイスは、1分間に1度温度と湿度を計測し、その計測した値をMQTTプローラにメッセージPubする。回路図は、図11の様になっている。MQTTトピックは、それぞれ温度センサは「/client/temperature/」、湿度センサは「/client/humidity/」を用いている。このセンサは、I2C(Integrated Circuit)でセンサデータを送ってくるため、アナログ入力ピンを必要としないため、取り扱いが非常に容易である。温度センサデバイスは、1分に1度センサから値を読み取り、MQTTプローラに読み取った温度をトピックでpubする。

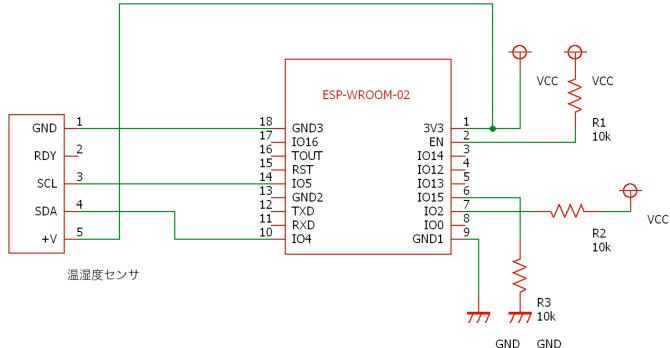


図11 温湿度センサデバイスの回路図

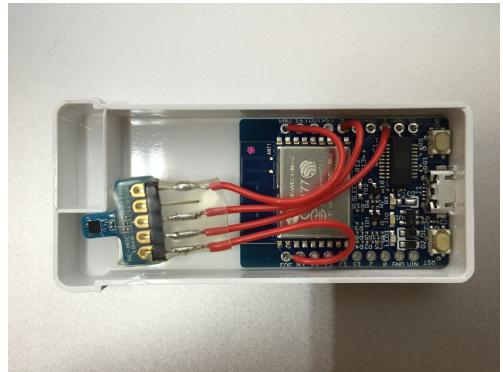


図12 実際の温湿度センサデバイス

■タクトスイッチ

タクトスイッチを搭載したデバイスは、ユーザの好きなタイミングでレシピを実行出来るボタンをスマートフォン・PCの様なソフトウェアからだけでなく、ハードウェアからも行える様にするものである。これは、センサによって決まった動作を読み取って行うことが出来ない場面で用いるのに有効であると言える。スイッチが押されるとMQTTプローラに、トピック「/client/switch/」でメッセージをPubする。回路図は、図13の様になっている。

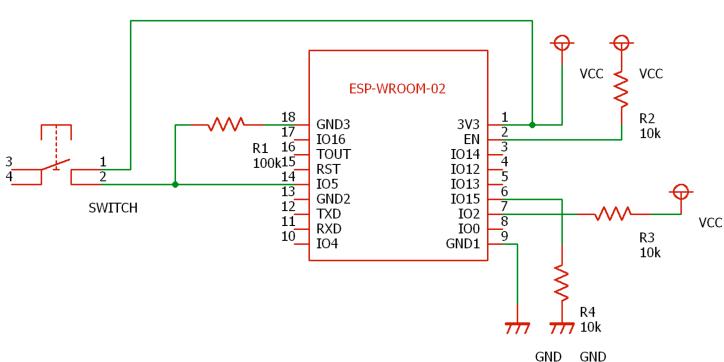


図13 タクトスイッチデバイスの回路図

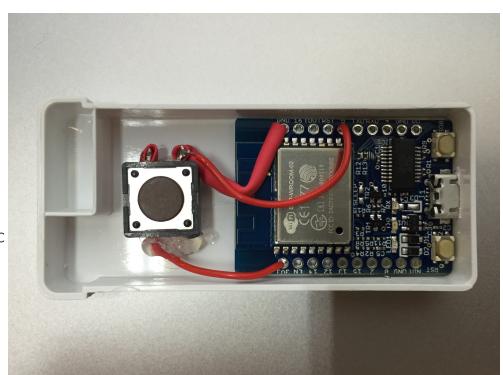


図14 実際のタクトスイッチデバイス

■赤外線送信

赤外線 LED を用いて赤外線通信を行うことが出来る。回路図は、図 15 の通りである。このデバイスは、TV に赤外線を送信することによりリモコンとして機能することが出来る。MQTT トピック「/server/tvIr/」で Sub したメッセージに応じて送信する赤外線情報を決定している。MQTT の Sub はコネクションを貼り続けることが出来るため、リアルタイムな応答が可能であり、通常の TV リモコンを用いたのと全く変わらない速さである。

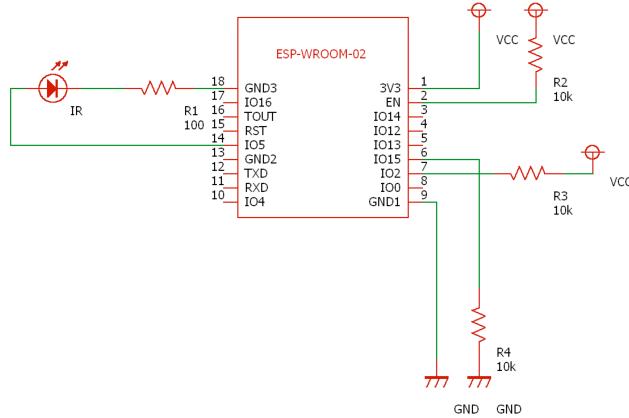


図 15 赤外線送信デバイス

■フルカラー LED

フルカラー LED を搭載したデバイスは、色と点灯時間、点滅間隔を自由に設定することが出来る。回路図は、図 16 の通りである。MQTT トピック「/server/onLED/」で Sub したメッセージの情報を元に点灯させる色および点灯時間、点滅間隔を決定する。

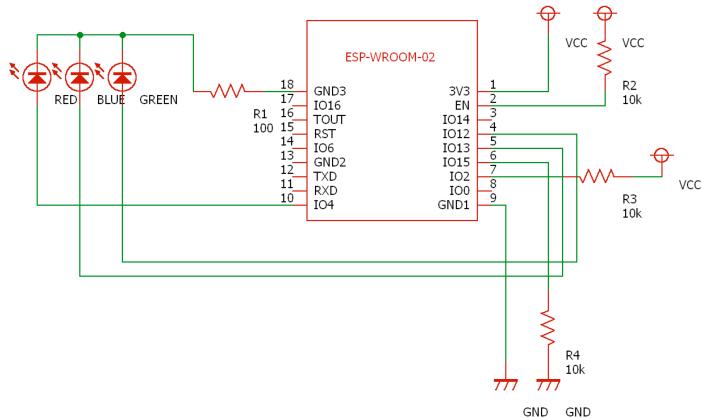


図 16 フルカラー LED デバイス

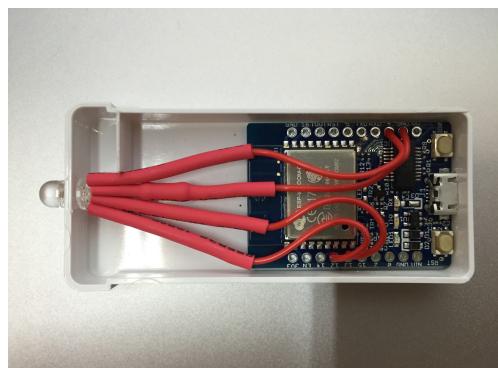


図 17 実際のフルカラー LED デバイス

■ブザー

ブザーを搭載したデバイスは、圧電ブザーによってユーザに通知を行えるデバイスである。回路図は、図 18 の通りである。MQTT トピック「/server/ringSpeaker」で Sub したメッセージに記述された何秒間鳴らしているかを受け取ってブザーを鳴らしている。

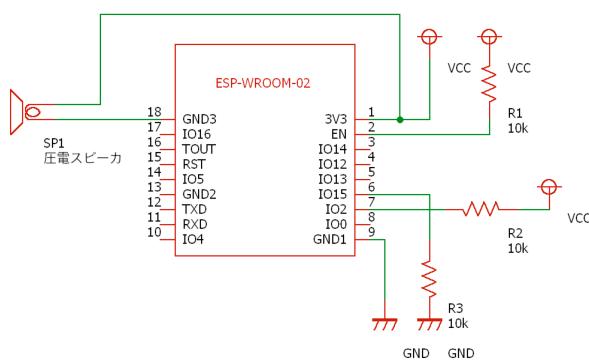


図 18 ブザーデバイス

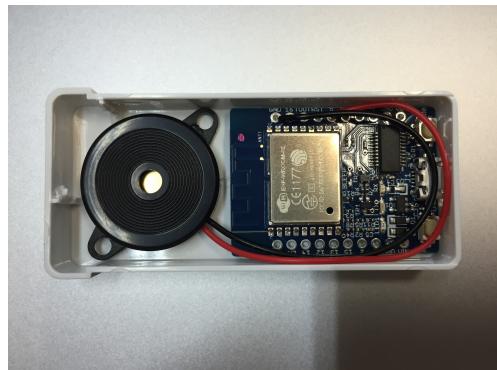


図 19 実際のブザーデバイス

4 これまでの IoT 製品との違い

今回構築した IoT 環境と現行の IoT 製品を比較する。比較する IoT 製品の一例として、フィリップスが開発した「Hue」というスマート LED 照明がある。Hue は、センサとフルカラー LED を内蔵した電球とブリッジ、アプリで構成される環境により、照明および光を利用した通知が行えるものである。この IoT 製品と比較し、3 つの観点から利点を述べる。

4.1 センサとアウトプット部分の分離

今回構築した環境は、Hue の様にセンサとアウトプット部分が 1ヶ所に集合されたかたちではなく、全て個々に分離されているため、情報の収集と通知を受け取りたい場所を任意に設定することが出来る。

また、センサを複数設置する際のコストを抑えることが出来る。例えば、センサを複数の場所に設置したい場合、Hue の場合では、センサとアウトプット部分がセットになっているため、1つセンサを増やすだけでも多くのコストがかかってしまう。これに対し、構築した環境は、センサのみを増やすことが出来るので、コストを低く抑えることが出来る。

4.2 ユーザによる機能の決定

Hue に限らず多くの製品において言えることだが、製品には事前に機能が設定されているため、ユーザがその機能が実行される条件に合わせて製品の設置や行動を行う必要があった。しかし、今回構築した環境では、レシピという形でユーザに自由に機能を設定出来る様にした。これにより、機能が実行される条件にユーザが合わせるのではなく、ユーザの行動に環境を合わせられるようになり、よりユーザが意識すること無く、設定した機能を実行することが出来る様になった。

4.3 設置場所の自由度

センサおよびアウトプット部分の設置場所の自由度を上げるには、サイズが小さいこと、電源の確保が容易であること、省電力であることである。

サイズは、FRISK のケースに収めることにより、小さくまとめることが出来た。また、厚さが 1cm 程度であるため目につきにくい狭い場所にも設置することが可能である。

電力の供給は、5V の USB を使用している。そのため、図 20 の様に PC から電源を確保できたり、図 21 の様にコンセントから容易に電源を確保することが出来る。

また、各デバイスは省電力であるため図 22 の様に、モバイルバッテリからも電源が確保でき、さらに設置の場を広げることが出来る。モバイルバッテリによる稼働時間は、容量が 5000mAh のモバイルバッテリで 3.5 日である。



図 20 PC から電源を確保



図 21 コンセントから電源を確保



図 22 モバイルバッテリから電源を確保

付録 A サーバで稼働するプログラム

公式ドキュメントを見ながらコーディングを行ったため、個人のコードを流用した箇所は無い。

A.1 server.py (Web サイトを稼働させる)

```
1 import os
2 import time
3 import json
4 import tornado.ioloop
5 import tornado.web
6 import tornado.websocket
7 from tornado.ioloop import PeriodicCallback
8 from tornado.options import define, options, parse_command_line
9
10 define("port", default = __PORT__, help = "run on the given port", type = int)
11
12 class RecipeModalHandler(tornado.web.RequestHandler):
13
14     def get(self):
15         self.render('index.html')
16
17 class IndexHandler(tornado.web.RequestHandler):
18     @tornado.web.asynchronous
19     def get(self):
20         f = open("recipeData.json", "r", encoding='utf-8')
21         recipeDataJson = json.loads(f.read())
22         f.close()
23
24         recipeNameList = []
25         for i in range(len(recipeDataJson)):
26             recipeNameList.append((recipeDataJson[i]["recipeName"]))
27         titles = recipeNameList
28         self.render("index.html", titles=titles)
29
30 class SendWebSocket(tornado.websocket.WebSocketHandler):
31     #on_message -> receive data
32     #write_message -> send data
33
34     def open(self):
35         self.i = 0
36         self.callback = PeriodicCallback(self._send_message, 400) #遅延用コールバック
37         self.callback.start()
38         print ("WebSocket opened")
39
40     #クライアントからメッセージが送られてくると呼び出す
41     def on_message(self, message):
42         print (message)
43
44         #リクエストを受けたレシピデータを送信
45         if ('requestRecipeData' in str(message)):
46             recipeName = str(message).replace('requestRecipeData', '')
47             f = open("recipeData.json", "r", encoding='utf-8')
48             recipeDataJson = json.loads(f.read())
49             f.close()
50             for i in range(len(recipeDataJson)):
51                 if (recipeDataJson[i]["recipeName"] == recipeName):
52                     break
53             print (json.dumps(recipeDataJson[i], sort_keys=True, indent=2, ensure_ascii=False))
54             self.write_message(recipeDataJson[i])
55
56         #レシピ追加のボタンが押された時に空のレシピデータを送信
57         elif ('addNewRecipe' in str(message)):
58             newRecipeDataJson = {
59                 "recipeName": "",
60                 "recipeId": 0,
61                 "actionDataIf": "-",
62                 "actionDataThen": "-",
63                 "valueIf": [""],
64                 "valueThen": [""]
65             }
66             self.write_message(newRecipeDataJson)
67
68         #レシピ実行ボタンが押された時
69         elif ('actionRecipe' in str(message)):
70             actionRecipeId = str(message).replace('actionRecipe', '')
71             cmd = "mosquitto_pub -h __ADDRESS__ -t \"/client/actionRecipe\" -m \" + actionRecipeId + \""
72             print (cmd)
73             os.system(cmd)
74
75         #レシピが変更・追加された時にレシピデータを外部ファイルrecipeData.jsonに出力
76         elif ('f' in str(message)):
77             newRecipeData = json.loads(message)
78             print (json.dumps(newRecipeData, sort_keys=True, indent=2, ensure_ascii=False))
```

```

79     f = open("recipeData.json", "r", encoding='utf-8')
80     recipeDataJson = json.loads(f.read())
81     f.close()
82
83     #新規レシピをrecipeData.jsonに出力
84     if (newRecipeData["recipeId"] == 0) :
85         print(0)
86         try:
87             newRecipeData["recipeId"] = recipeDataJson[len(recipeDataJson)-1]["recipeId"] + 1
88         except: #レシピが1つも登録されていなかった時の処理
89             newRecipeData["recipeId"] = 1
90     newRecipeDataList = recipeDataJson
91     newRecipeDataList.append(newRecipeData)
92     print (json.dumps(newRecipeData, sort_keys=True, indent=2, ensure_ascii=False))
93     with open('recipeData.json', 'w') as f:
94         json.dump(newRecipeDataList, f, sort_keys=True, indent=2, ensure_ascii=False)
95
96     #編集したレシピをrecipeData.jsonに出力
97     elif (newRecipeData["recipeId"] <= int(recipeDataJson[len(recipeDataJson)-1]["recipeId"])) :
98         print(1)
99         newRecipeDataList = []
100        for i in range(len(recipeDataJson)) :
101            if (newRecipeData["recipeId"] == recipeDataJson[i]["recipeId"]) :
102                newRecipeDataList.append(newRecipeData)
103            else :
104                newRecipeDataList.append(recipeDataJson[i])
105        with open('recipeData.json', 'w') as f :
106            json.dump(newRecipeDataList, f, sort_keys=True, indent=2, ensure_ascii=False)
107
108    #レシピに変更・追加があったことをPubしてレシピ管理システムactionRecipe.pyに伝える
109    cmd = "mosquitto_pub -h ___ADDRESS___ -t \"/client/recipeData\" -m \"change\""
110    print (cmd)
111    os.system(cmd)
112
113    #レシピの削除
114    elif ('deleteRecipe' in str(message)) :
115        deleteRecipeId = int(str(message).replace('deleteRecipe', ''))
116        newRecipeDataList = []
117        f = open("recipeData.json", "r", encoding='utf-8')
118        recipeDataJson = json.loads(f.read())
119        f.close()
120        for i in range(len(recipeDataJson)) :
121            if (deleteRecipeId != recipeDataJson[i]["recipeId"]) :
122                newRecipeDataList.append(recipeDataJson[i])
123        with open('recipeData.json', 'w') as f:
124            json.dump(newRecipeDataList, f, sort_keys=True, indent=2, ensure_ascii=False)
125    #レシピが削除されたことをPubしてレシピ管理システムactionRecipe.pyに伝える
126    cmd = "mosquitto_pub -h ___ADDRESS___ -t \"/client/recipeData\" -m \"change\""
127    print (cmd)
128    os.system(cmd)
129
130    #ネットワークリモコンのリクエストをMQTTプロトコルにPubを送信
131    else :
132        cmd = "mosquitto_pub -h ___ADDRESS___ -t \"/server/tvIr\" -m \"\" + message + \"\""
133        print (cmd)
134        os.system(cmd)
135
136    #コールバックスタートで呼び出し開始
137    def _send_message(self):
138        self.i += 1
139        #self.write_message(str(self.i))
140
141    #ページが閉じ、コネクションが切断された場合に呼び出す
142    def on_close(self):
143        self.callback.stop()
144        print ("WebSocket closed")
145
146 app = tornado.web.Application([
147     (r"/", IndexHandler),
148     (r"/ws", SendWebSocket),
149     ],
150     static_path=os.path.join(os.getcwd(), "bootstrap"),
151 )
152
153 if __name__ == "__main__":
154     parse_command_line()
155     app.listen(options.port)
156     tornado.ioloop.IOLoop.instance().start()

```

A.2 index.html (リモコン・レシピ管理サイト)

```

1 <!DOCTYPE html>
2 <html lang="ja">
3 <title>リモコン</title>
4 <script language="javascript">
5 var ws = new WebSocket("ws://__ADDRESS__/ws");
6
7 //サーバへレシピデータをJSONデータで送信
8 function mergeSelectData(request) {
9     if (request == "newRecipe") {
10         var message = {
11             recipeName : $("#recipeName").val(),
12             recipeId : parseInt($("#recipeId").val()),
13             actionDataIf : $("#selectActionIf option:selected").text(),
14             actionDataThen : $("#selectActionThen option:selected").text(),
15             valueIf : [$( "#valueIf" ).val().split(",")],
16             valueThen : [$( "#valueThen" ).val().split(",")]
17         };
18         ws.send(JSON.stringify(message));
19     };
20     location.reload();
21 }
22
23 //サーバへリクエストを送信
24 function requestAction(request) {
25     if (request == "deleteRecipe") {
26         ws.send("deleteRecipe" + $("#recipeId").val())
27         location.reload();
28     }
29     else if (request == "actionRecipe") {
30         ws.send("actionRecipe" + $("#recipeId").val())
31     }
32     else {
33         document.getElementById('valueIf').placeholder=request;
34         ws.send(request);
35     }
36 }
37
38 //サーバからレシピデータをJSONで受け取る
39 ws.onmessage = function (evt) {
40     //var data = evt.data
41     var data = JSON.parse(evt.data);
42     document.getElementById('recipeName').value=data.recipeName;
43     document.getElementById('recipeId').value=data.recipeId;
44     document.getElementById('selectActionIf').value=data.actionDataIf;
45     document.getElementById('selectActionThen').value=data.actionDataThen;
46     document.getElementById('valueIf').value=data.valueIf;
47     document.getElementById('valueIf').placeholder="値を入力";
48     document.getElementById('valueThen').value=data.valueThen;
49     document.getElementById('valueThen').placeholder="値を入力";
50 };
51
52 function valueLabelIf (selectActionIf) {
53     if (selectActionIf == "人感センサが反応") {
54         document.getElementById('valueIf').placeholder="入力の必要はありません";
55     }
56     else if (selectActionIf == "温度センサが反応") {
57         document.getElementById('valueIf').placeholder="反応温度範囲（上限, 下限）";
58     }
59     else if (selectActionIf == "湿度センサが反応") {
60         document.getElementById('valueIf').placeholder="反応湿度範囲（上限, 下限）";
61     }
62     else if (selectActionIf == "加速度センサが反応") {
63         document.getElementById('valueIf').placeholder="入力の必要はありません";
64     }
65     else if (selectActionIf == "スイッチが押される") {
66         document.getElementById('valueIf').placeholder="入力の必要はありません";
67     }
68 }
69
70 function valueLabelThen (selectActionThen) {
71     if (selectActionThen == "TVに赤外線送信") {
72         document.getElementById('valueThen').placeholder="送信順序に入力, …, …";
73     }
74     else if (selectActionThen == "LEDを点灯") {
75         document.getElementById('valueThen').placeholder="色, 点灯時間[秒], 点灯間隔[秒]";
76     }
77     else if (selectActionThen == "ブザーを鳴らす") {
78         document.getElementById('valueThen').placeholder="鳴らす時間[秒]";
79     }
80 }
81 </script>
82 <head>
83     <meta charset="utf-8">
84     <meta http-equiv="X-UA-Compatible" content="IE=edge">
85     <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">

```

```

86  <!-- The above 3 meta tags *must* come first in the head; any other head content must come *after* these
87  tags -->
88
89  <!-- Bootstrap -->
90  <link rel="stylesheet" href="{{ static_url("css/bootstrap.min.css") }}>
91
92  <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
93  <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
94  <!--[if lt IE 9]>
95      <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
96      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
97  <![endif]-->
98
99  </head>
100 <body>
101     <div class="row">
102         <ul class="nav nav-tabs">
103             <li class="active col-xs-6"><a href="#tab1" data-toggle="tab">TVリモコン </a></li>
104             <li class="col-xs-6"><a href="#tab2" data-toggle="tab">レシピ</a></li>
105         </ul>
106     </div>
107     <div class="tab-content">
108         <!--TVリモコンタブ-->
109         <div class="tab-pane active" id="tab1">
110             <div class="row">
111                 <div class="col-sm-6">
112                     <p>
113                         <div class="row">
114                             <div class="col-xs-12"></div>
115                         </div>
116                     </p>
117                     <div class="row">
118                         <div class="col-xs-8"></div>
119                         <button type="button" onclick="requestAction('p')" class="btn btn-danger btn-lg col-xs-4">
120                             <span class="glyphicon glyphicon-off" aria-hidden="true"></span>Power
121                         </button>
122                     </div>
123                     <p>
124                         <div class="row">
125                             <button type="button" onclick="requestAction('1')" class="btn btn-default btn-lg col-xs-4">1
126                             ch</button>
127                             <button type="button" onclick="requestAction('2')" class="btn btn-default btn-lg col-xs-4">2
128                             ch</button>
129                             <button type="button" onclick="requestAction('3')" class="btn btn-default btn-lg col-xs-4">3
130                             ch</button>
131                         </div>
132                     </p>
133                     <p>
134                         <div class="row">
135                             <button type="button" onclick="requestAction('4')" class="btn btn-default btn-lg col-xs-4">4
136                             ch</button>
137                             <button type="button" onclick="requestAction('5')" class="btn btn-default btn-lg col-xs-4">5
138                             ch</button>
139                             <button type="button" onclick="requestAction('6')" class="btn btn-default btn-lg col-xs-4">6
140                             ch</button>
141                         </div>
142                     </p>
143                     <p>
144                         <div class="row">
145                             <button type="button" onclick="requestAction('7')" class="btn btn-default btn-lg col-xs-4">7
146                             ch</button>
147                             <button type="button" onclick="requestAction('8')" class="btn btn-default btn-lg col-xs-4">8
148                             ch</button>
149                             <button type="button" onclick="requestAction('9')" class="btn btn-default btn-lg col-xs-4">9
150                             ch</button>
151                         </div>
152                     </p>
153                     <div class="col-sm-6">
154                         <p>
155                             <div class="row">
156                                 <div class="col-xs-12"></div>
157                             </div>
158                         <p>
159                             <div class="row">
160                                 <button type="button" onclick="requestAction('C')" class="btn btn-default btn-lg col-xs-4">
161                                     <span class="glyphicon glyphicon-triangle-top" aria-hidden="true"></span>UP
162                                     </button>
163                                 <button type="button" onclick="requestAction('V')" class="btn btn-default btn-lg col-xs-4">
164                                     col-xs-offset-4</button>
165                                     <span class="glyphicon glyphicon-volume-up" aria-hidden="true"></span>UP

```

```

163         </button>
164     </div>
165   </p>
166   <div class="row">
167     <button type="button" onclick="requestAction('c')" class="btn btn-default btn-lg col-xs-4">
168       <span class="glyphicon glyphicon-triangle-bottom" aria-hidden="true"></span>DOWN
169     </button>
170     <button type="button" onclick="requestAction('v')" class="btn btn-default btn-lg col-xs-4
171       col-xs-offset-4">
172       <span class="glyphicon glyphicon-volume-down" aria-hidden="true"></span>DOWN
173     </button>
174   </div>
175   </p>
176 </div>
177 </div>
178 <!--レシピタブ-->
179 <div class="tab-pane" id="tab2">
180   <div class="row">
181     <div class="col-sm-4"></div>
182     <div class="col-sm-4">
183       <div class="row">
184         <p>
185           <a href="#" data-toggle="modal" data-target="#recipeModal" data-whatever="レシピ追加">
186             <button type="button" class="btn btn-default col-xs-10 col-xs-offset-1" onclick="
187               requestAction('addNewRecipe'))">
188               <span class="glyphicon glyphicon-plus" aria-hidden="true"></span>レシピ追加
189             </button>
190           </a>
191         </p>
192       </div>
193       <!--レシピ一覧を作成-->
194       {% for title in titles %}
195         <div class="row">
196           <p>
197             <div class="dropdown col-xs-12">
198               <button class="btn btn-success dropdown-toggle center-block col-xs-12" type="button" id=
199                 "dropdownMenu" data-toggle="dropdown" aria-haspopup="true" aria-expanded="true"
200                 onclick="requestAction('requestRecipeData{{ escape(title) }}'))">
201                 {{ escape(title) }}
202                 <span class="caret"></span>
203               </button>
204               <ul class="dropdown-menu col-xs-12" aria-labelledby="dropdownMenu">
205                 <li><a href="#" data-toggle="modal" data-target="#recipeModal" data-whatever="レシピ編
206                   集">編集</a></li>
207                 <li><a href="#" onclick="requestAction('actionRecipe')">実行</a></li>
208                 <li role="separator" class="divider"></li>
209                 <li><a href="#" data-toggle="modal" data-target="#deleteRecipeModal">削除</a></li>
210               </ul>
211             </div>
212           <% end %>
213         </div>
214       <div class="col-sm-4"></div>
215       <div class="modal fade" id="recipeModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
216         <div class="modal-dialog">
217           <div class="modal-content">
218             <div class="modal-header">
219               <button type="button" class="close" data-dismiss="modal">
220                 <span aria-hidden="true">×</span><span class="sr-only">閉じる</span>
221               </button>
222               <h4 class="modal-title" id="exampleModalLabel">レシピ追加</h4>
223             </div><!-- /modal-header -->
224             <div class="modal-body">
225               <p>
226                 <input type="text" class="form-control" id="recipename" placeholder="レシピ名">
227               </p>
228               <input type="hidden" class="form-control" id="recipetid" >
229               <div class="row">
230                 <div class="col-sm-6">
231                   <h5 class="col-sm-2">If</h5>
232                   <div class="col-sm-10">
233                     <select class="form-control" id="selectActionIf" value="--" onclick="valueLabelIf(
234                       value)">
235                       <option>--</option>
236                       <option>人感センサが反応</option>
237                       <option>温度センサが反応</option>
238                       <option>湿度センサが反応</option>
239                       <option>加速度センサが反応</option>
240                       <option>スイッチが押される</option>
241                     </select>
242                     <p>
243                       <input type="text" class="form-control" id="valueIf" placeholder="値を入力">
244                     </p>
245                   </div>
246                 <div class="col-sm-6">
247                   <h5 class="col-sm-2">then</h5>

```

```

247         <div class="col-sm-10">
248             <select class="form-control" id="selectActionThen" value="-" onclick="valueLabelThen
249                 (value)">
250                 <option>-</option>
251                 <option>TVに赤外線送信</option>
252                 <option>LEDを点灯</option>
253                 <option>ブザーを鳴らす</option>
254             </select>
255             <p>
256                 <input type="text" class="form-control" id="valueThen" placeholder="値を入力">
257             </p>
258         </div>
259     </div>
260     <div class="modal-footer">
261         <button type="button" class="btn btn-default" data-dismiss="modal">キャンセル</button>
262         <button type="button" class="btn btn-primary" onclick="mergeSelectData('newRecipe')" data-
263             dismiss="modal">保存
264         </button>
265     </div>
266 </div>
267 </div>
268 <div class="modal fade" id="deleteRecipeModal" tabindex="-1" role="dialog" aria-labelledby="
269     exampleModalLabel" aria-hidden="true">
270     <div class="modal-dialog">
271         <div class="modal-content">
272             <div class="modal-header">
273                 <button type="button" class="close" data-dismiss="modal">
274                     <span aria-hidden="true">&#215;</span><span class="sr-only">閉じる</span>
275                 </button>
276                 <h4 class="modal-title" id="exampleModalLabel">レシピの削除</h4>
277             </div><!-- /modal-header -->
278             <div class="modal-body">
279                 <div class="alert alert-danger" role="alert">
280                     <span class="glyphicon glyphicon-exclamation-sign" aria-hidden="true"></span>
281                     <span class="sr-only">Error:</span>
282                     この動作は取り消せません。本当に削除しますか？
283                 </div>
284                 <input type="hidden" class="form-control" id="recipeId">
285             </div>
286             <div class="modal-footer">
287                 <button type="button" class="btn btn-default" data-dismiss="modal">キャンセル</button>
288                 <button type="button" class="btn btn-danger" onclick="requestAction('deleteRecipe')" data-
289                     dismiss="modal">削除
290                 </button>
291             </div>
292         </div>
293     </div>
294 </div>
295
296 <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
297 <script src="{{ "https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js" }}></script>
298 <!-- Include all compiled plugins (below), or include individual files as needed -->
299 <script src="{{ static_url("js/bootstrap.min.js") }}></script>
300 <script type="text/javascript">
301     $('#recipeModal').on('show.bs.modal', function (event) {
302         var button = $(event.relatedTarget) //モーダルを呼び出すときに使われたボタンを取得
303         var recipient = button.data('whatever') //data-whatever の値を取得
304         //Ajaxの処理はここに
305         var modal = $(this) //モーダルを取得
306         modal.find('.modal-title').text(recipient) //モーダルのタイトルに値を表示
307         modal.find('.modal-body input#recipient-name').val(recipient) //inputタグにも表示
308     })
309 </script>
310 </body>
311 </html>

```

A.3 actionRecipe.py (レシピ実行管理システム)

```

1 # -*- coding: utf-8 -*-
2 import re
3 import json
4 import paho.mqtt.client as mqtt
5
6 host = '___ADDRESS___'
7 port = ___PORT___
8
9 #レシピデータを格納する空のjsonデータ
10 recipeDataList = {
11     "recipeName" : "",
12     "recipeId" : 0,
13     "actionDataIf" : "-",
14     "actionDataThen" : "-",
15     "valueIf" : [""],
16     "valueThen" : []
17 }
18
19 #レシピデータのIfとThenを日本語からMQTTのtopic名に変換
20 def transformTopicName(recipeDataJson):
21     global recipeDataList
22     topicList = [[',人感センサが反応', ',温度センサが反応', ',湿度センサが反応', ',加速度センサが反応', ',スイッチが押される', ',TVに赤外線送信', ',LEDを点灯', ',ブザーを鳴らす'], ['PIRsensor', 'temperature', 'humidity', 'acceleration', 'switch', 'tvIr', 'onLED', 'ringSpeaker']]
23
24     for i in range(len(recipeDataJson)):
25         for j in range(len(topicList[0])):
26             if (recipeDataJson[i]["actionDataIf"] == topicList[0][j]):
27                 recipeDataJson[i]["actionDataIf"] = topicList[1][j]
28             if (recipeDataJson[i]["actionDataThen"] == topicList[0][j]):
29                 recipeDataJson[i]["actionDataThen"] = topicList[1][j]
30     print (json.dumps(recipeDataJson, sort_keys=True, indent=2, ensure_ascii=False))
31     recipeDataList = recipeDataJson.copy()
32
33     return recipeDataList
34
35 #外部ファイルrecipeData.jsonからレシピデータを取得
36 def openRecipeDataFile():
37     f = open("recipeData.json", "r", encoding='utf-8')
38     recipeDataJson = json.loads(f.read())
39     f.close()
40
41     return recipeDataJson
42
43 #Subscribeするtopicを設定
44 def on_connect(client, userdata, flags, respons_code):
45     print('status {0}'.format(respons_code))
46
47     client.subscribe('/client/#')    #クライアントからPubされたメッセージを全てSubする
48
49 #メッセージをSubした時に実行する
50 def on_message(client, userdata, msg):
51     global recipeDataList
52     deletePattern = re.compile('[\[\]\\\\ ]')
53
54     #レシピデータが更新された時に外部ファイルrecipeData.jsonからレシピデータを取得し直す
55     if ('recipeData' in str(msg.topic)):
56         print ('recipeData')
57         recipeDataList = transfo^^cc^81rmTopicName(openRecipeDataFile())
58
59     #レシピ管理サイトでレシピ実行ボタンが押された時に登録されたactionDataThenをPubする
60     elif ('actionRecipe' in str(msg.topic)):
61         print ('actionRecipe')
62         for i in range(len(recipeDataList)):
63             if (recipeDataList[i]["recipeId"] == int(msg.payload)):
64                 if (recipeDataList[i]["actionDataThen"] == "onLED"):
65                     client.publish('/server/onLED/color', deletePattern.sub('', str(recipeDataList[i]["valueThen"])[0][0])))
66                     client.publish('/server/onLED/time', deletePattern.sub('', str(recipeDataList[i]["valueThen"])[0][1])))
67                     client.publish('/server/onLED/interval', deletePattern.sub('', str(recipeDataList[i]["valueThen"])[0][2])))
68                 else :
69                     client.publish('/server/' + recipeDataList[i]["actionDataThen"], deletePattern.sub('', str(recipeDataList[i]["valueThen"])[0])))
70                     break
71
72     #Subした内容に応じて登録されたactionDataThenをPubする
73     else :
74         for i in range(len(recipeDataList)):
75             if (recipeDataList[i]["actionDataIf"] in str(msg.topic)):
76                 if (recipeDataList[i]["actionDataIf"] == 'temperature' or recipeDataList[i]["actionDataIf"] == 'humidity'):
77                     if (int(recipeDataList[i]["valueIf"])[0][0]) >= int(msg.payload) and int(recipeDataList[i]["valueIf"])[0][1] <= int(msg.payload)):
78                         if (recipeDataList[i]["actionDataThen"] == "onLED"):

```

```

79         client.publish('/server/onLED/color', deletePattern.sub('', str(recipeDownList[i]
80                         ]["valueThen"][0][0])))
81         client.publish('/server/onLED/time', deletePattern.sub('', str(recipeDownList[i]
82                         ]["valueThen"][0][1])))
83         client.publish('/server/onLED/interval', deletePattern.sub('', str(
84                         recipeDownList[i]["valueThen"][0][2])))
85     else :
86         client.publish('/server/' + recipeDownList[i]["actionDataThen"], deletePattern.
87                         sub('', str(recipeDownList[i]["valueThen"][0])))
88     else :
89         client.publish('/server/' + recipeDownList[i]["actionDataThen"], deletePattern.sub('',
90                         str(recipeDownList[i]["valueThen"][0])))
91     print ('other')
92
93 if __name__ == '__main__':
94     # 外部ファイルrecipeData.jsonからレシピデータを取得
95     transformTopicName(openRecipeDataFile())
96
97     print (recipeDownList)
98
99     # Publisherと同様に v3.1.1を利用
100    client = mqtt.Client(protocol=mqtt.MQTTv311)
101
102    client.on_connect = on_connect
103    client.on_message = on_message
104
105    client.connect(host, port=port, keepalive=60)
106
107    # 待ち受け状態にする
108    client.loop_forever()

```

A.4 recipeData.json (レシピデータ)

```
1  {
2      "actionDataIf": "センサの反応",
3      "actionDataThen": "アウトプットデバイスの指定",
4      "recipeId": レシピID,
5      "recipeName": "レシピ名",
6      "valueIf": [
7          [
8              "センサの条件値"
9          ]
10     ],
11     "valueThen": [
12         [
13             "アウトプットデバイスへ送信する値"
14         ]
15     ]
16 }
```

A.5 passIFTTT.py (外部サービス IFTTT との連携)

```
1 # -*- coding: utf-8 -*-
2 import re
3 import json
4 import requests
5 import paho.mqtt.client as mqtt
6
7 host = '___ADDRESS___'
8 port = ___PORT___
9
10
11 #Subscribeするtopicを設定
12 def on_connect(client, userdata, flags, respons_code):
13     print('status {0}'.format(respons_code))
14
15     client.subscribe('/server/IFTTT/#')    #クライアントからPubされたメッセージを全てSubする
16
17 #メッセージをSubしたらIFTTTに送信
18 def on_message(client, userdata, msg):
19     global recipeDataList
20     deletePattern = re.compile('[\\[\\]\\]\\ ]')
21
22     messageList = str(msg.payload).lstrip("b'").rstrip(">").split(",")
23     print(messageList)
24     requests.post("https://maker.ifttt.com/trigger/" + str(messageList[0]) + "/with/key/" + str(messageList
25     [1]))
26
27 if __name__ == '__main__':
28     # Publisherと同様に v3.1.1 を利用
29     client = mqtt.Client(protocol=mqtt.MQTTv311)
30
31     client.on_connect = on_connect
32     client.on_message = on_message
33
34     client.connect(host, port=port, keepalive=60)
35
36     # 待ち受け状態にする
37     client.loop_forever()
```

付録 B クライアントと各デバイスで稼働するプログラム

公式ドキュメントを見ながらコーディングを行ったため、個人のコードを流用した箇所は無い。

B.1 PIRsensor_pub.ino（人感センサデバイス）

```
1 //*****  
2 // 人感センサ SB412Aの反応をMQTTで送信するMQTTClient  
3 //  
4 //ESP-WROOM-02のアナログ入力は1Vに分圧してTOUTに入力  
5 //  
6 // http://akizukidenki.com/catalog/g/gM-09002/  
7 // https://github.com/adafruit/Adafruit_MQTT_Library  
8 //*****  
9  
10 #include <ESP8266WiFi.h>  
11 #include <Adafruit_MQTT.h>  
12 #include <Adafruit_MQTT_Client.h>  
13 extern "C" {  
14 #include "user_interface.h"  
15 }  
16  
17 //***** WiFi Access Point *****  
18  
19 #define WLAN_SSID      "___SSID___"  
20 #define WLAN_PASS      "___PASS___"  
21  
22 //***** Your Setup *****  
23  
24 #define YOUR_SERVER      "___ADDRESS___"  
25 #define YOUR_SERVERPORT    ___PORTS___  
26 #define YOUR_USERNAME     "...your username..."  
27 #define YOUR_PASSWORD     "...your password..."  
28  
29 //***** Global State *****  
30  
31 // MQTT serverに接続するESP8266 WiFiClientクラスを作成  
32 WiFiClient client;  
33  
34 // MQTT server, クライアントID, ユーザ名, およびパスワードを保管  
35 const char MQTT_SERVER[] PROGMEM = YOUR_SERVER;  
36 const char MQTT_CLIENTID[] PROGMEM = ___TIME___ "_test_client_id";  
37 const char MQTT_USERNAME[] PROGMEM = YOUR_USERNAME;  
38 const char MQTT_PASSWORD[] PROGMEM = YOUR_PASSWORD;  
39  
40 // Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.  
41 Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, YOUR_SERVERPORT, MQTT_CLIENTID, MQTT_USERNAME, MQTT_PASSWORD  
    );  
42  
43 //***** Topic *****  
44  
45 // publisher & subscriberの設定  
46 const char TOPIC_PIRSENSOR[] PROGMEM = "/client/PIRsensor";  
47  
48 Adafruit_MQTT_Publish PIRsensorPublisher = Adafruit_MQTT_Publish(&mqtt, TOPIC_PIRSENSOR);  
49  
50 //***** Sketch Code *****  
51  
52  
53 void setup() {  
54     Serial.begin(115200);  
55  
56     // Wi-Fiに接続  
57     Serial.println(); Serial.println();  
58     Serial.print("Connecting to ");  
59     Serial.println(WLAN_SSID);  
60  
61     WiFi.begin(WLAN_SSID, WLAN_PASS);  
62     // WiFi.config(IPAddress(), WiFi.gatewayIP(), WiFi.subnetMask());  
63     while (WiFi.status() != WL_CONNECTED) {  
64         delay(500);  
65         Serial.print(".");  
66     }  
67     Serial.println();  
68  
69     Serial.println("WiFi connected");  
70     Serial.println("IP address: "); Serial.println(WiFi.localIP());  
71 }  
72  
73 int onCount = 0;  
74  
75 void loop() {  
76     MQTT_connect();  
77 }
```

```

78 //unsigned int 型の入れ物を用意
79 uint ADC_Value = 0;
80
81 //AD変換実行
82 ADC_Value = system_adc_read();
83
84 //計測結果をシリアルに書き出す。0-1024の値
85 Serial.println("ANALOG DATA " + String(ADC_Value));
86
87 //4回大きな値をセンサが返したらMQTTでMQTTプローラにPublish
88 if (ADC_Value > 512) {
89     onCount++;
90 } else {
91     onCount = 0;
92 }
93 if (onCount == 4) {
94     PIRsensorPublisher.publish("on");
95     onCount = 0;
96 }
97
98 // MQTTプローラにpingを送る
99 if(! mqtt.ping()) {
100     mqtt.disconnect();
101 }
102
103 delay(1000);
104 }
105
106 // MQTT serverの接続状況を調べ切断されていれば再接続
107 void MQTT_connect() {
108     int8_t ret;
109
110     // MQTT serverの接続状況を調べる
111     if (mqtt.connected()) {
112         return;
113     }
114
115     Serial.print("Connecting to MQTT... ");
116
117     while ((ret = mqtt.connect()) != 0) { // 接続された場合は0を返す 接続されない場合は5秒おきに接続を繰り
118         // 戻す
119         Serial.println(mqtt.connectErrorString(ret));
120         Serial.println(ret);
121         Serial.println("Retrying MQTT connection in 5 seconds... ");
122         mqtt.disconnect();
123         delay(5000); // 5秒間待機
124     }
125     Serial.println("MQTT Connected!");
126 }

```

B.2 temp-humSensor_pub.ino (温湿度センサデバイス)

```
1 //*****温湿度センサHDC1000を用いたMQTT_Client*****
2 // I2C通信を用いる
3 // Vin to 3-5VDC
4 // GND to ground
5 // SCL to I2C clock pin (5 on ESP-WR000M-02)
6 // SDA to I2C data pin (4 on ESP-WR000M-02)
7
8 // https://github.com/adafruit/Adafruit_MQTT_Library
9 // http://akizukidenki.com/catalog/g/gM-08775/
10 //***** *****
11
12 #include <ESP8266WiFi.h>
13 #include <Adafruit_MQTT.h>
14 #include <Adafruit_MQTT_Client.h>
15 #include <Wire.h>
16 #include "Adafruit_HDC1000.h"
17
18 //***** WiFi Access Point *****/
19
20 #define WLAN_SSID      "____SSID____"
21 #define WLAN_PASS      "____PASS____"
22
23 //***** Your Setup *****/
24
25 #define YOUR_SERVER    "____ADDRESS____"
26 #define YOUR_SERVERPORT  ____PORTS_____
27 #define YOUR_USERNAME   "...your username...""
28 #define YOUR_PASSWORD   "...your password...""
29
30 //***** Global State *****/
31
32 // MQTT serverに接続するESP8266 WiFiClientクラスを作成
33 WiFiClient client;
34
35 // MQTT server, クライアントID, ユーザ名, およびパスワードを保管
36 const char MQTT_SERVER[] PROGMEM = YOUR_SERVER;
37 const char MQTT_CLIENTID[] PROGMEM = __TIME__ " _test_client_id";
38 const char MQTT_USERNAME[] PROGMEM = YOUR_USERNAME;
39 const char MQTT_PASSWORD[] PROGMEM = YOUR_PASSWORD;
40
41 // Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
42 Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, YOUR_SERVERPORT, MQTT_CLIENTID, MQTT_USERNAME, MQTT_PASSWORD
43 );
44
45 //***** Topic *****/
46
47 // publisher & subscriberの設定
48 const char TOPIC_TEMPERATURE[] PROGMEM = "/client/temperature";
49 const char TOPIC_HUMIDITY[] PROGMEM = "/client/humidity";
50
51 Adafruit_MQTT_Publish temperaturePublisher = Adafruit_MQTT_Publish(&mqtt, TOPIC_TEMPERATURE);
52 Adafruit_MQTT_Publish humidityPublisher = Adafruit_MQTT_Publish(&mqtt, TOPIC_HUMIDITY);
53
54 //***** Sketch Code *****/
55
56 Adafruit_HDC1000 hdc = Adafruit_HDC1000();
57
58 void setup() {
59     Serial.begin(115200);
60
61     //温湿度センサが接続されていることを確認
62     if (!hdc.begin()) {
63         Serial.println("Couldn't find sensor!");
64         while (1);
65     }
66
67     // Wi-Fiに接続
68     Serial.println(); Serial.println();
69     Serial.print("Connecting to ");
70     Serial.println(WLAN_SSID);
71
72     WiFi.begin(WLAN_SSID, WLAN_PASS);
73     // WiFi.config(IPAddress(), WiFi.gatewayIP(), WiFi.subnetMask());
74     while (WiFi.status() != WL_CONNECTED) {
75         delay(500);
76         Serial.print(".");
77     }
78     Serial.println();
79
80     Serial.println("WiFi connected");
81     Serial.println("IP address: "); Serial.println(WiFi.localIP());
82 }
83
84 void loop() {
```

```

86 MQTT_connect();
87 double tempValue = 0; //温度センサから読み取ったデータを格納
88 double humValue = 0; //湿度センサから読み取ったデータを格納
89
90 //温度および湿度をセンサから読み取る
91 tempValue = hdc.readTemperature();
92 humValue = hdc.readHumidity();
93
94 //温湿度センサから得られた値をMQTTでMQTTプローラにPublish
95 temperaturePublisher.publish(tempValue);
96 humidityPublisher.publish(humValue);
97
98 Serial.print("Temp: "); Serial.print(tempValue);
99 Serial.print("\t\thum: "); Serial.println(humValue);
100 delay(1000*60); //1分に1回センサから値を読み取る
101
102 // MQTT プローラに ping を送る
103 if(! mqtt.ping()) {
104   mqtt.disconnect();
105 }
106 }
107
108 // MQTT server の接続状況を調べ切断されていれば再接続
109 void MQTT_connect() {
110   int8_t ret;
111
112   // MQTT server の接続状況を調べる
113   if (mqtt.connected()) {
114     return;
115   }
116
117   Serial.print("Connecting to MQTT... ");
118
119   while ((ret = mqtt.connect()) != 0) { // 接続された場合は0を返す 接続されない場合は5秒おきに接続を繰り
120     // 返す
121     Serial.println(mqtt.connectErrorString(ret));
122     Serial.println(ret);
123     Serial.println("Retrying MQTT connection in 5 seconds...");
124     mqtt.disconnect();
125     delay(5000); // 5秒間待機
126   }
127   Serial.println("MQTT Connected!");
}

```

B.3 switch_pub.ino (タクトスイッチデバイス)

```
1 //*****  
2 // タクトスイッチが押されるとPubをするMQTTClient  
3 //  
4 // https://github.com/adafruit/Adafruit_MQTT_Library  
5 //*****  
6  
7 #include <ESP8266WiFi.h>  
8 #include <Adafruit_MQTT.h>  
9 #include <Adafruit_MQTT_Client.h>  
10  
11 //***** WiFi Access Point *****  
12  
13 #define WLAN_SSID      "___SSID___"  
14 #define WLAN_PASS     "___PASS___"  
15  
16 //***** Your Setup *****  
17  
18 #define YOUR_SERVER      "___ADDRESS___"  
19 #define YOUR_SERVERPORT    ___PORTS___  
20 #define YOUR_USERNAME     "...your username..."  
21 #define YOUR_PASSWORD      "...your password..."  
22  
23 //***** Global State *****  
24  
25 // MQTT serverに接続するESP8266 WiFiClientクラスを作成  
26 WiFiClient client;  
27  
28 // MQTT server, クライアントID, ユーザ名, およびパスワードを保管  
29 const char MQTT_SERVER[] PROGMEM = YOUR_SERVER;  
30 const char MQTT_CLIENTID[] PROGMEM = ___TIME___ "_test_client_id";  
31 const char MQTT_USERNAME[] PROGMEM = YOUR_USERNAME;  
32 const char MQTT_PASSWORD[] PROGMEM = YOUR_PASSWORD;  
33  
34 // Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.  
35 Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, YOUR_SERVERPORT, MQTT_CLIENTID, MQTT_USERNAME, MQTT_PASSWORD  
    );  
36  
37 //***** Topic *****  
38  
39 // publisher & subscriberの設定  
40 const char TOPIC_SWITCH[] PROGMEM = "/client/switch";  
41  
42 Adafruit_MQTT_Publish temperaturePublisher = Adafruit_MQTT_Publish(&mqtt, TOPIC_SWITCH);  
43  
44 //***** Sketch Code *****  
45  
46  
47 void setup() {  
48     Serial.begin(115200);  
49     pinMode(16, INPUT);  
50  
51     // Wi-Fiに接続  
52     Serial.println(); Serial.println();  
53     Serial.print("Connecting to ");  
54     Serial.println(WLAN_SSID);  
55  
56     WiFi.begin(WLAN_SSID, WLAN_PASS);  
57     // WiFi.config(IPAddress(), WiFi.gatewayIP(), WiFi.subnetMask());  
58     while (WiFi.status() != WL_CONNECTED) {  
59         delay(500);  
60         Serial.print(".");  
61     }  
62     Serial.println();  
63  
64     Serial.println("WiFi connected");  
65     Serial.println("IP address: "); Serial.println(WiFi.localIP());  
66 }  
67  
68 void loop() {  
69     MQTT_connect();  
70  
71     //タクトスイッチが押されたらPub  
72     if (digitalRead(16) == HIGH) {  
73         Serial.println("on");  
74         temperaturePublisher.publish("on");  
75         delay(500);  
76     }  
77  
78     // MQTTプローラにpingを送る  
79     if(! mqtt.ping()) {  
80         mqtt.disconnect();  
81     }  
82 }  
83  
84 // MQTT serverの接続状況を調べ切断されていれば再接続  
85 void MQTT_connect() {
```

```
86     int8_t ret;
87
88     // MQTT serverの接続状況を調べる
89     if (mqtt.connected()) {
90         return;
91     }
92
93     Serial.print("Connecting to MQTT... ");
94
95     while ((ret = mqtt.connect()) != 0) { // 接続された場合は0を返す　接続されない場合は5秒おきに接続を繰り
96         // 返す
97         Serial.println(mqtt.connectErrorString(ret));
98         Serial.println(ret);
99         Serial.println("Retrying MQTT connection in 5 seconds... ");
100        mqtt.disconnect();
101        delay(5000); // 5秒間待機
102    }
103    Serial.println("MQTT Connected!");
```

B.4 tvIr_sub.ino (赤外線送信デバイス)

```
1 //***** MQTT topic "tvIr" を Subした時にTVに赤外線送信を行う *****
2 // 赤外線送信はNECフォーマットを使用
3
4 // https://github.com/adafruit/Adafruit_MQTT_Library
5 // https://github.com/markszabo/IRremoteESP8266
6 //***** *****
7
8 #include <ESP8266WiFi.h>
9 #include <Adafruit_MQTT.h>
10 #include <Adafruit_MQTT_Client.h>
11 #include <IRremoteESP8266.h>
12 #include <String.h>
13
14 //***** WiFi Access Point *****
15
16 #define WLAN_SSID      "___SSID___"
17 #define WLAN_PASS      "___PASS___"
18
19 //***** Your Setup *****
20
21 #define YOUR_SERVER      "___ADDRESS___"
22 #define YOUR_SERVERPORT  ___PORTS___
23 #define YOUR_USERNAME    "...your username...""
24 #define YOUR_PASSWORD    "...your password...""
25
26 //***** Global State *****
27
28 // MQTT serverに接続するESP8266 WiFiClientクラスを作成
29 WiFiClient client;
30
31 // MQTT server, クライアントID, ユーザ名, およびパスワードを保管
32 const char MQTT_SERVER[] PROGMEM = YOUR_SERVER;
33 const char MQTT_CLIENTID[] PROGMEM = ___TIME___ "_test_client_id";
34 const char MQTT_USERNAME[] PROGMEM = YOUR_USERNAME;
35 const char MQTT_PASSWORD[] PROGMEM = YOUR_PASSWORD;
36
37 // Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
38 Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, YOUR_SERVERPORT, MQTT_CLIENTID, MQTT_USERNAME, MQTT_PASSWORD
39 );
40
41 //***** Topic *****
42
43 // publisher & subscriberの設定
44 const char TOPIC_TVIR[] PROGMEM = "/server/tvIr";
45
46 Adafruit_MQTT_Subscribe tvIrSubscriber = Adafruit_MQTT_Subscribe(&mqtt, TOPIC_TVIR);
47
48 //***** Sketch Code *****
49
50
51 IRsend irsend(5); // GPIO 5を赤外線送信に使用
52
53 void setup() {
54     Serial.begin(115200);
55     irsend.begin();
56
57     // Wi-Fiに接続
58     Serial.println(); Serial.println();
59     Serial.print("Connecting to ");
60     Serial.println(WLAN_SSID);
61
62     WiFi.begin(WLAN_SSID, WLAN_PASS);
63     // WiFi.config(IPAddress(), WiFi.gatewayIP(), WiFi.subnetMask());
64     while (WiFi.status() != WL_CONNECTED) {
65         delay(500);
66         Serial.print(".");
67     }
68     Serial.println();
69
70     Serial.println("WiFi connected");
71     Serial.println("IP address: "); Serial.println(WiFi.localIP());
72
73     // MQTT subscriptionを設定
74     mqtt.subscribe(&tvIrSubscriber);
75 }
76
77
78 // TVに赤外線送信
79 void sendirData(String data) {
80     for (int i = 0; i <= data.length(); i++) {
81         String dataS = data.substring(i, i+1);
82         char dataC[32];
83         dataS.toCharArray(dataC, 32); // char型に変換
84         switch (dataC[0]) {
85             case 'p':
```

```

86     Serial.println("p");
87     irsend.sendNEC(0x2FD48B7, 32);
88     break;
89   case '1':
90     Serial.println("1");
91     irsend.sendNEC(0x2FD807F, 32);
92     break;
93   case '2':
94     Serial.println("2");
95     irsend.sendNEC(0x2FD40BF, 32);
96     break;
97   case '3':
98     Serial.println("3");
99     irsend.sendNEC(0x2FDC03F, 32);
100    break;
101   case '4':
102     Serial.println("4");
103     irsend.sendNEC(0x2FD20DF, 32);
104     break;
105   case '5':
106     Serial.println("5");
107     irsend.sendNEC(0x2FDA05F, 32);
108     break;
109   case '6':
110     Serial.println("6");
111     irsend.sendNEC(0x2FD609F, 32);
112     break;
113   case '7':
114     Serial.println("7");
115     irsend.sendNEC(0x2FDE01F, 32);
116     break;
117   case '8':
118     Serial.println("8");
119     irsend.sendNEC(0x2FD10EF, 32);
120     break;
121   case '9':
122     Serial.println("9");
123     irsend.sendNEC(0x2FD906F, 32);
124     break;
125   case 'x':
126     Serial.println("x");
127     irsend.sendNEC(0x2FD50AF, 32);
128     break;
129   case 'y':
130     Serial.println("y");
131     irsend.sendNEC(0x2FDD02F, 32);
132     break;
133   case 'z':
134     Serial.println("z");
135     irsend.sendNEC(0x2FD30CF, 32);
136     break;
137   case 'C':
138     Serial.println("C");
139     irsend.sendNEC(0x2FDD827, 32);
140     break;
141   case 'c':
142     Serial.println("c");
143     irsend.sendNEC(0x2FDF807, 32);
144     break;
145   case 'V':
146     Serial.println("V");
147     irsend.sendNEC(0x2FD58A7, 32);
148     break;
149   case 'v':
150     Serial.println("v");
151     irsend.sendNEC(0x2FD7887, 32);
152     break;
153   }
154   delay(500);
155 }
156 }
157
158 void loop() {
159   MQTT_connect();
160
161   String recevIrData;
162
163   // topicを監視しsubscriptionのmessageを受け取る
164   Adafruit_MQTT_Subscribe *subscription;
165   while ((subscription = mqtt.readSubscription(1000))) {
166     if (subscription == &tvIrSubscriber) {
167       Serial.print(F("Got: "));
168       Serial.println((char *)tvIrSubscriber.lastread);
169       recevIrData = String((char *)tvIrSubscriber.lastread); // subscriptionのmessageを受け取る
170       sendIrData(recevIrData);
171     }
172   }
173
174   // MQTTプロードにpingを送る
175   if(! mqtt.ping()) {
176     mqtt.disconnect();

```

```
177     }
178     delay(1000);
180 }
181
182 // MQTT serverの接続状況を調べ切断されていれば再接続
183 void MQTT_connect() {
184     int8_t ret;
185
186     // MQTT serverの接続状況を調べる
187     if (mqtt.connected()) {
188         return;
189     }
190
191     Serial.print("Connecting to MQTT... ");
192
193     while ((ret = mqtt.connect()) != 0) { // 接続された場合は0を返す 接続されない場合は5秒おきに接続を繰り
194         // 返す
195         Serial.println(mqtt.connectErrorString(ret));
196         Serial.println(ret);
197         Serial.println("Retrying MQTT connection in 5 seconds...");
198         mqtt.disconnect();
199         delay(5000); // 5秒間待機
200     }
201     Serial.println("MQTT Connected!");
}
```

B.5 onLED_sub.ino (フルカラー LED デバイス)

```

1  /*******************************************************************************
2   * MQTT topic "onLED" を Sub した時にLEDを点灯させる
3   *
4   * https://github.com/adafruit/Adafruit_MQTT_Library
5   *******************************************************************************/
6
7 #include <ESP8266WiFi.h>
8 #include <Adafruit_MQTT.h>
9 #include <Adafruit_MQTT_Client.h>
10
11 //***** WiFi Access Point *****/
12
13 #define WLAN_SSID      "___SSID___"
14 #define WLAN_PASS     "___PASS___"
15
16 //***** Your Setup *****/
17
18 #define YOUR_SERVER      "___ADDRESS___"
19 #define YOUR_SERVERPORT    ___PORTS___
20 #define YOUR_USERNAME     "...your username...""
21 #define YOUR_PASSWORD     "...your password...""
22
23 //***** Global State *****/
24
25 // MQTT serverに接続するESP8266 WiFiClientクラスを作成
26 WiFiClient client;
27
28 // MQTT server, クライアントID, ユーザ名, およびパスワードを保管
29 const char MQTT_SERVER[] PROGMEM = YOUR_SERVER;
30 const char MQTT_CLIENTID[] PROGMEM = ___TIME___ "_test_client_id";
31 const char MQTT_USERNAME[] PROGMEM = YOUR_USERNAME;
32 const char MQTT_PASSWORD[] PROGMEM = YOUR_PASSWORD;
33
34 // Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
35 Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, YOUR_SERVERPORT, MQTT_CLIENTID, MQTT_USERNAME, MQTT_PASSWORD
36 );
37
38 //***** Topic *****/
39
40 // publisher & subscriberの設定
41 const char TOPIC_ONLED_COLOR[] PROGMEM = "/server/onLED/color";
42 const char TOPIC_ONLED_TIME[] PROGMEM = "/server/onLED/time";
43 const char TOPIC_ONLED_INTERVAL[] PROGMEM = "/server/onLED/interval";
44
45 Adafruit_MQTT_Subscribe onLEDColorSubscriber = Adafruit_MQTT_Subscribe(&mqtt, TOPIC_ONLED_COLOR);
46 Adafruit_MQTT_Subscribe onLEDTIMESubscriber = Adafruit_MQTT_Subscribe(&mqtt, TOPIC_ONLED_TIME);
47 Adafruit_MQTT_Subscribe onLEDIntervalSubscriber = Adafruit_MQTT_Subscribe(&mqtt, TOPIC_ONLED_INTERVAL);
48
49 //***** Sketch Code *****/
50
51 void setup() {
52   Serial.begin(115200);
53   pinMode(4, OUTPUT);
54   pinMode(5, OUTPUT);
55   pinMode(16, OUTPUT);
56
57   // Wi-Fiに接続
58   Serial.println(); Serial.println();
59   Serial.print("Connecting to ");
60   Serial.println(WLAN_SSID);
61
62   WiFi.begin(WLAN_SSID, WLAN_PASS);
63   // WiFi.config(IPAddress(), WiFi.gatewayIP(), WiFi.subnetMask());
64   while (WiFi.status() != WL_CONNECTED) {
65     delay(500);
66     Serial.print(".");
67   }
68   Serial.println();
69
70   Serial.println("WiFi connected");
71   Serial.println("IP address: "); Serial.println(WiFi.localIP());
72
73   // MQTT subscriptionを設定
74   mqtt.subscribe(&onLEDColorSubscriber);
75   mqtt.subscribe(&onLEDTIMESubscriber);
76   mqtt.subscribe(&onLEDIntervalSubscriber);
77 }
78
79 int timeDataInt = 0;
80 int intervalDataInt = 0;
81
82 void setupLED(String colorData, int timeData, int intervalData) {
83   Serial.println("setupLED");
84   Serial.println(timeData/(2*intervalData));
85   timeDataInt = 0;

```

```

86     intervalDataInt = 0;
87     if (colorData.indexOf("赤") != -1 || colorData.indexOf('r') != -1) {
88         Serial.println("赤");
89         for (int i = 0; i < timeData/(2*intervalData); i++) {
90             digitalWrite(4, HIGH);
91             delay(intervalData);
92             digitalWrite(4, LOW);
93             delay(intervalData);
94         }
95     } else if (colorData.indexOf("緑") != -1 || colorData.indexOf('g') != -1) {
96         Serial.println("緑");
97         for (int i = 0; i < timeData/(2*intervalData); i++) {
98             digitalWrite(5, HIGH);
99             delay(intervalData);
100            digitalWrite(5, LOW);
101            delay(intervalData);
102        }
103    } else if (colorData.indexOf("青") != -1 || colorData.indexOf('b') != -1) {
104        Serial.println("青");
105        for (int i = 0; i < timeData/(2*intervalData); i++) {
106            digitalWrite(16, HIGH);
107            delay(intervalData);
108            digitalWrite(16, LOW);
109            delay(intervalData);
110        }
111    }
112 }
113
114 void loop() {
115     MQTT_connect();
116     String colorData;
117     String timeData;
118     String intervalData;
119
120     //topicをPubし、"onLED"をPubした時にLEDを点灯させる
121     Adafruit_MQTT_Subscribe *subscription;
122     while ((subscription = mqtt.readSubscription(1000))) {
123         if (subscription == &onLEDColorSubscriber) {
124             Serial.print(F("Got: "));
125             Serial.println((char *)onLEDColorSubscriber.lastread);
126             colorData = String((char *)onLEDColorSubscriber.lastread);
127         }
128         if (subscription == &onLEDTIMESubscriber) {
129             Serial.print(F("Got: "));
130             Serial.println((char *)onLEDTIMESubscriber.lastread);
131             timeData = String((char *)onLEDTIMESubscriber.lastread);
132         }
133         if (subscription == &onLEDIntervalSubscriber) {
134             Serial.print(F("Got: "));
135             Serial.println((char *)onLEDIntervalSubscriber.lastread);
136             intervalData = String((char *)onLEDIntervalSubscriber.lastread);
137         }
138         char timeDataChar[32];
139         timeData.toCharArray(timeDataChar, 32);
140         timeDataInt = atoi(timeDataChar);
141         Serial.println(timeDataInt);
142         char intervalDataChar[32];
143         intervalData.toCharArray(intervalDataChar, 32);
144         intervalDataInt = atoi(intervalDataChar);
145         Serial.println(intervalDataInt);
146         if (timeDataInt != 0 && intervalDataInt != 0) {
147             setupLED(colorData, timeDataInt, intervalDataInt);
148         }
149     }
150
151     // MQTTプロトコルにpingを送る
152     if(! mqtt.ping()) {
153         mqtt.disconnect();
154     }
155
156     delay(1000);
157 }
158
159 // MQTT serverの接続状況を調べ切断されていれば再接続
160 void MQTT_connect() {
161     int8_t ret;
162
163     // MQTT serverの接続状況を調べる
164     if (mqtt.connected()) {
165         return;
166     }
167
168     Serial.print("Connecting to MQTT... ");
169
170     while ((ret = mqtt.connect()) != 0) { // 接続された場合は0を返す 接続されない場合は5秒おきに接続を繰り
171         // 戻す
172         Serial.println(mqtt.connectErrorString(ret));
173         Serial.println(ret);
174         Serial.println("Retrying MQTT connection in 5 seconds... ");
175         mqtt.disconnect();
176         delay(5000); // 5秒間待機

```

```
176     }
177     Serial.println("MQTT Connected!");
178 }
```

B.6 ringSpeaker_sub.ino (ブザーデバイス)

```

1  **** MQTT topic "ringSpeaker" を Sub した時にブザーを鳴らす ****
2  MQTT topic "ringSpeaker" を Sub した時にブザーを鳴らす
3
4  https://github.com/adafruit/Adafruit_MQTT_Library
5  ****
6
7 #include <ESP8266WiFi.h>
8 #include <Adafruit_MQTT.h>
9 #include <Adafruit_MQTT_Client.h>
10
11 **** WiFi Access Point ****
12
13 #define WLAN_SSID      "___SSID___"
14 #define WLAN_PASS     "___PASS___"
15
16 **** Your Setup ****
17
18 #define YOUR_SERVER      "___ADDRESS___"
19 #define YOUR_SERVERPORT   ___PORTS___
20 #define YOUR_USERNAME    "...your username..."
21 #define YOUR_PASSWORD    "...your password..."
22
23 **** Global State ****
24
25 // MQTT serverに接続するESP8266 WiFiClientクラスを作成
26 WiFiClient client;
27
28 // MQTT server, クライアントID, ユーザ名, およびパスワードを保管
29 const char MQTT_SERVER[] PROGMEM = YOUR_SERVER;
30 const char MQTT_CLIENTID[] PROGMEM = ___TIME___ "_test_client_id";
31 const char MQTT_USERNAME[] PROGMEM = YOUR_USERNAME;
32 const char MQTT_PASSWORD[] PROGMEM = YOUR_PASSWORD;
33
34 // Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
35 Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, YOUR_SERVERPORT, MQTT_CLIENTID, MQTT_USERNAME, MQTT_PASSWORD
36 );
37
38 // **** Topic ****
39 // pubulisher & subscriberの設定
40 const char TOPIC_RINGSPEAKER[] PROGMEM = "/server/ringSpeaker";
41
42 Adafruit_MQTT_Subscribe ringSpeakerSubscriber = Adafruit_MQTT_Subscribe(&mqtt, TOPIC_RINGSPEAKER);
43
44 // **** Sketch Code ****
45
46 static const int speakerPin = 16;
47
48 void setup() {
49   Serial.begin(115200);
50   pinMode(speakerPin, OUTPUT);
51
52   // Wi-Fiに接続
53   Serial.println(); Serial.println();
54   Serial.print("Connecting to ");
55   Serial.println(WLAN_SSID);
56
57   WiFi.begin(WLAN_SSID, WLAN_PASS);
58   // WiFi.config(IPAddress(), WiFi.gatewayIP(), WiFi.subnetMask());
59   while (WiFi.status() != WL_CONNECTED) {
60     delay(500);
61     Serial.print(".");
62   }
63   Serial.println();
64
65   Serial.println("WiFi connected");
66   Serial.println("IP address: "); Serial.println(WiFi.localIP());
67
68   // MQTT subscriptionを設定
69   mqtt.subscribe(&ringSpeakerSubscriber);
70 }
71
72 void loop() {
73   MQTT_connect();
74
75   //topicをPubし, "ringSpeaker" を Pub した時にブザーを鳴らす
76   Adafruit_MQTT_Subscribe *subscription;
77   while ((subscription = mqtt.readSubscription(1000))) {
78     if (subscription == &ringSpeakerSubscriber) {
79       for (int i = 0; i < 30; i++) {
80         analogWrite(speakerPin, 500);
81         delay(50);
82         analogWrite(speakerPin, 0);
83

```

```

84         delay(50);
85     }
86 }
87 }
88 // MQTT ブローカに ping を送る
89 if(! mqtt.ping()) {
90     mqtt.disconnect();
91 }
92 }
93 delay(1000);
94 }
95 }
96 // MQTT server の接続状況を調べ切断されていれば再接続
97 void MQTT_connect() {
98     int8_t ret;
99
100    // MQTT server の接続状況を調べる
101    if (mqtt.connected()) {
102        return;
103    }
104
105    Serial.print("Connecting to MQTT... ");
106
107    while ((ret = mqtt.connect()) != 0) { // 接続された場合は0を返す 接続されない場合は5秒おきに接続を繰り
108        // 返す
109        Serial.println(mqtt.connectErrorString(ret));
110        Serial.println(ret);
111        Serial.println("Retrying MQTT connection in 5 seconds... ");
112        mqtt.disconnect();
113        delay(5000); // 5秒間待機
114    }
115    Serial.println("MQTT Connected!");
116 }

```

参考文献

- [1] 柴田 淳, 『みんなの Python 第 3 版』, 2012.
- [2] 池内孝啓 他, 『Python ライブライアリ厳選レシピ』, 2015.
- [3] ami_GS's diary (2014) 「Tornado で WebSocket やってみた」, <<http://ami-gs.hatenablog.com/entry/2014/02/14/121145>> (参照 2015-11-7).
- [4] Siguniang's Blog (2015) 「tornado で WebSocket サーバを動かしてみる」, <<https://siguniang.wordpress.com/2015/09/23/websocket-server-using-tornado/>> (参照 2015-11-7).
- [5] Tornado (2015) 「Tornado Web Server」, <<http://www.tornadoweb.org>> (参照 2015-11-7).
- [6] azusa9 (2015) 「ちょっと裏技っぽいけど、ESP8266 で AD 変換やる方法」, <<http://qiita.com/azusa9/items/26e74e4e0d5773ce9c41>> (参照 2015-11-7).
- [7] きむ茶工房ガレージハウス (2015) 「湿度センサー (HDC1000/AM2302) を動作させて見ます」, <<http://www.geocities.jp/zattouka/GarageHouse/micon/Arduino/Humidity/Humidity.htm>> (参照 2015-11-7).
- [8] Bootstrap (2015) 「Bootstrap Top」, <<http://getbootstrap.com>> (参照 2015-11-7).
- [9] eclipse.org (2015) 「paho」, <<https://eclipse.org/paho/>> (参照 2015-11-7).
- [10] 電子工作と介護と生活と・・・(2015) 「スイッチサイエンスさんの WROOM 開発ボード使ってみました」, <<http://qiita.com/akito1986/items/e9ca48cfcd56fdbf4c9d>> (参照 2015-11-18).
- [11] P I Cで遊ぶ電子工作 (2006) 「PWM機能を使ってスピーカを鳴らす」, <http://homepage3.nifty.com/mitt/pic/pic683_7.html> (参照 2016-1-27).
- [12] Disce gaudere. 楽しむ事を学べ (2012) 「簡易 HTTP な CoAP(Constrained Application Protocol) プロトコル」, <<http://d.hatena.ne.jp/keroring/20120417/1334639673>> (参照 2016-1-28).
- [13] Happy & Quick (2015) 「さくら VPS の標準 OS(CentOS 6.7) に Python3.5.0 をインストールする」, <<http://hqac.hatenadiary.com/entry/2015/11/29/125634>> (参照 2016-2-11).
- [14] akito1986 (2014) 「Pyenv による Python3.x 環境構築 (CentOS, Ubuntu)」, <<http://qiita.com/akito1986/items/be5dcd1a502aaaf22010b>> (参照 2016-2-11).
- [15] akito1986 (2014) 「sudo 時に PATH を引き継ぐ方法」, <<http://qiita.com/akito1986/items/e9ca48cfcd56fdbf4c9d>> (参照 2016-2-11).
- [16] 株式会社N T Tデータ・他 (2015) 『絵で見てわかる IoT/センサの仕組みと活用』翔泳社.