屋外操作、AIスピーカ連携
スマートリモコン

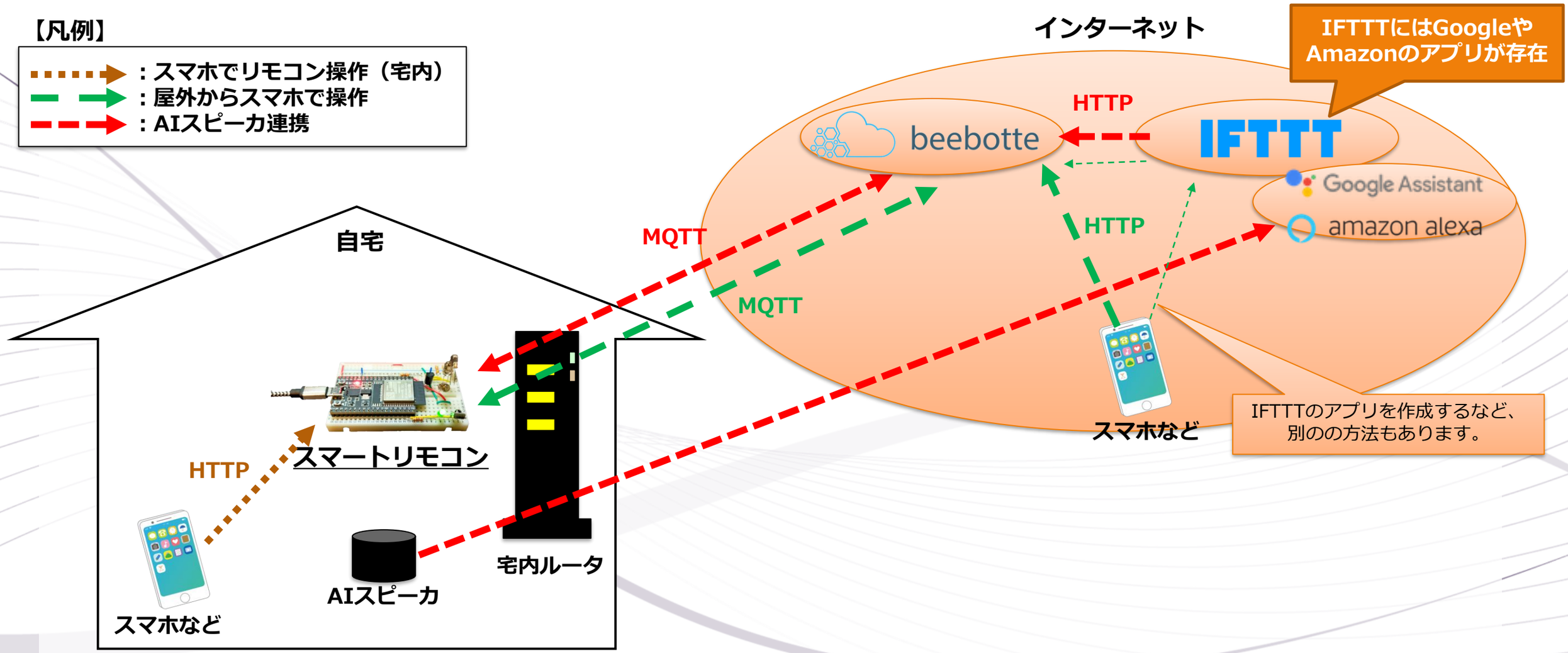
- MQTT利用による屋外操作
- AIスピーカ連携による音声操作

目　次　《屋外操作、AIスピーカ連携編》

1－1．スマートリモコン製作全体の流れ

No	項目	内容	ハード	ソフト	記事
1	概要	全体の流れ、システム構成、利用物品、 選定理由、開発環境など	-	-	別動画で配信
2	LED	初めて電子工作される方向けの基本を行います。 LEDの点灯、点滅を行う「Lチカ」を製作します。	○	○	
3	赤外線受信センサ	赤外線受信センサーの説明 回路図から配線、ソフトウェア	○	○	
4	赤外線送信LED	赤外線送信LEDの説明 回路図から配線、ソフトウェア	○	○	
5	スマホでLED操作 （宅内）	工作したリモコンのLEDを屋内のスマホから操作する ソフトウェアを製作します。（Webサーバ機能、SPIFFS操作）	-	○	
6	スマホでリモコン操作 （宅内）	工作したリモコンを屋内のスマホから操作する ソフトウェアを製作します。（ボタン名、信号保存・読出）	-	○	
7	屋外からスマホで操作 及び、AIスピーカ連携	工作したリモコンを屋外からスマホで操作したり AIスピーカ連携を実現するソフトウェアを製作します。	-	○	今回はこの動画

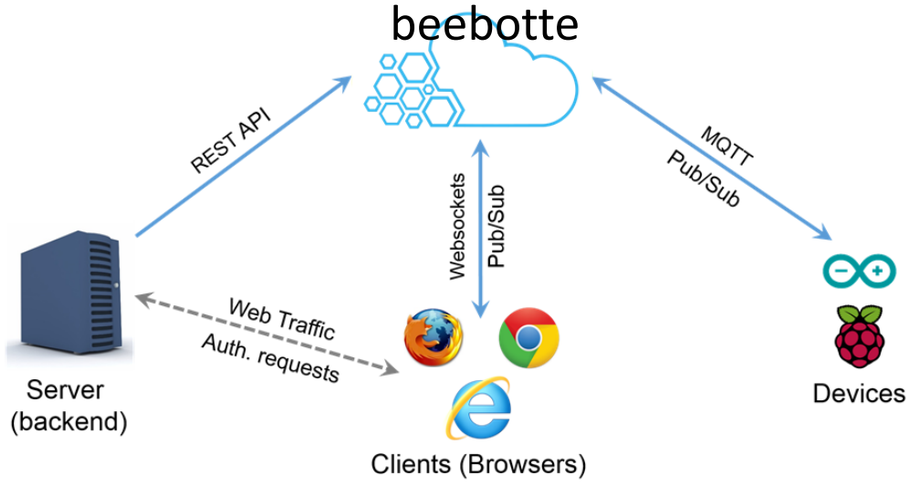
1－2. システム構成


【凡例】
：スマホでリモコン操作（宅内）
：屋外からスマホで操作
：AIスピーカ連携
インターネット
IFTTTにはGoogleやAmazonのアプリが存在
beebotte
IFTTT
HTTP
Google Assistant
amazon alexa
HTTP
MQTT
MQTT
自宅
スマートリモコン
HTTP
スマホなど
AIスピーカ
宅内ルータ
スマホなど
IFTTTのアプリを作成するなど、別のの方法もあります。

2. beebotteとは？

beebotteとは？

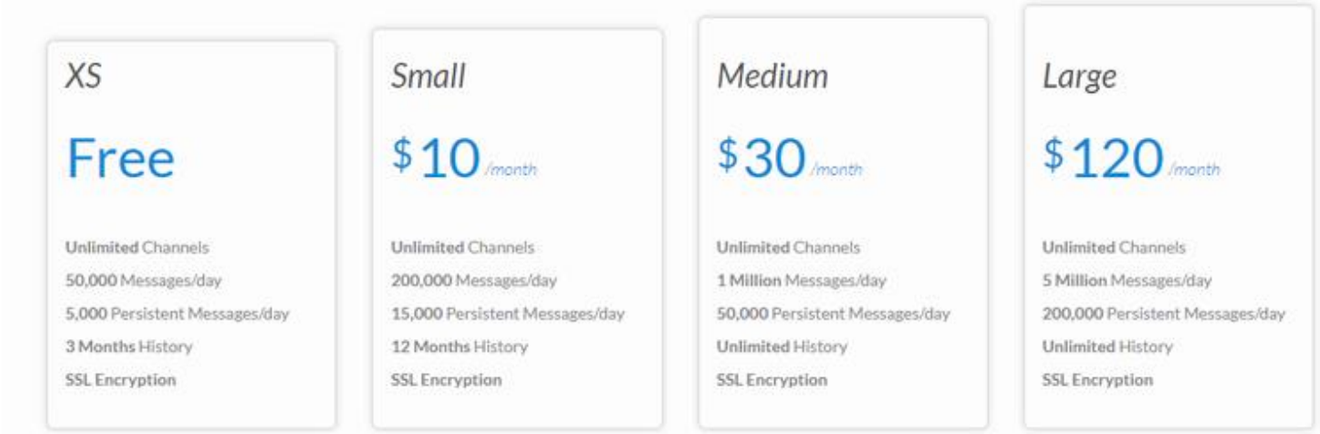
https://beebotte.com/overview



MQTTやREST-API[HTTP]の、Webソケット接続を提供するクラウドサービス
MQTTとHTTPなど異なるプロトコル間のデータを乗せ換えてくれるので、データ通信が可能となります。

料金

https://beebotte.com/plans



無料でも、１日５万メッセージの利用が可能

●接続だけで必要なメッセージ量
１５秒に１回のKeepAliveが必要。
4(回/分)*60(分)*24(時間)*2(送受) = 11520

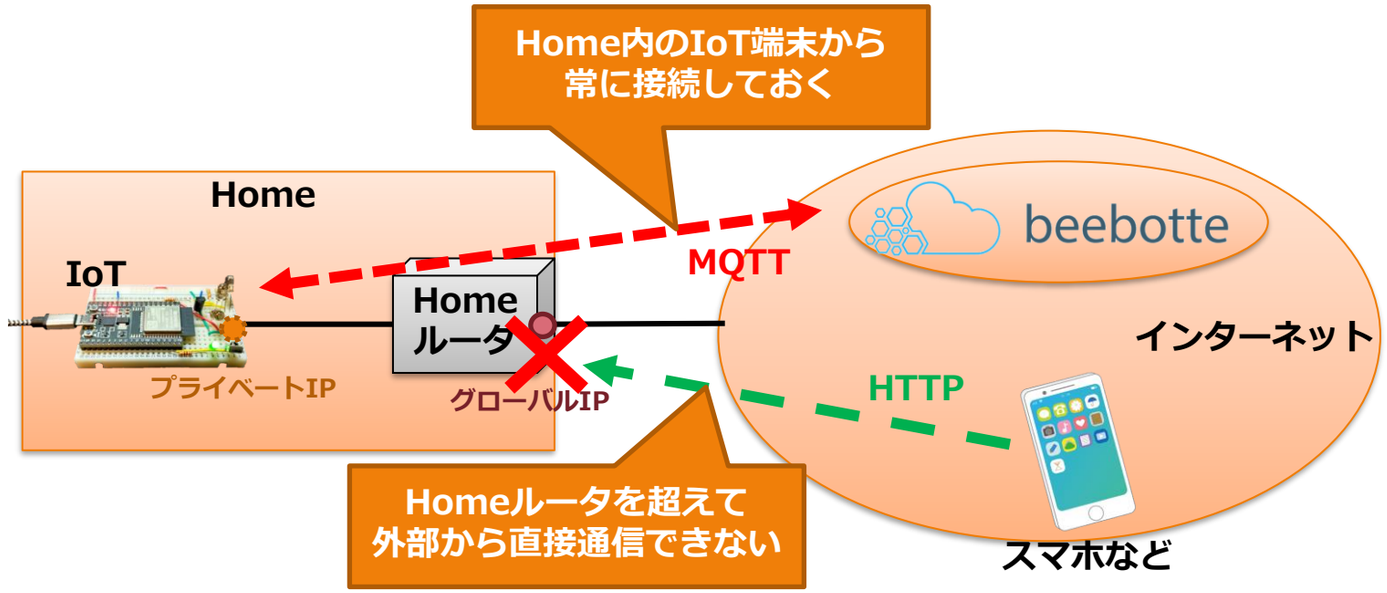
約１万２千メッセージ

3．MQTTとは？

MQTTとは

MQTTは「Message Queueing Telemetry Transport」の略で、IoTなどの機器が、機器同士やクラウドと通信するために開発されたシンプルで軽量なプロトコルです。
機能も少なく軽く動作するので、CPUやメモリなどの消費が小さく抑えられるので、IoT端末やリソース消費を抑えたい場合に適していると言えます。
ただ、画像などの大きなデータを通信することは適していないので、小さな情報を送るような場合に利用されます。

MQTTが必要な理由



Home内の端末はHome内でのみ利用可能なプライベートIPアドレスのため、屋外から直接アクセスできません。

Home内の端末からの通信はHomeルータによりグローバルIPアドレスに変換されるため、通信が可能となります。

このため、Home内のIoT端末からBeebotteへ常にMQTT通信を実施し接続しておくことで、その通信を利用し外部からのアクセスを可能としています。

Hobby-ITチャンネル
《IPアドレスと通信の仕組み》動画にて詳細解説

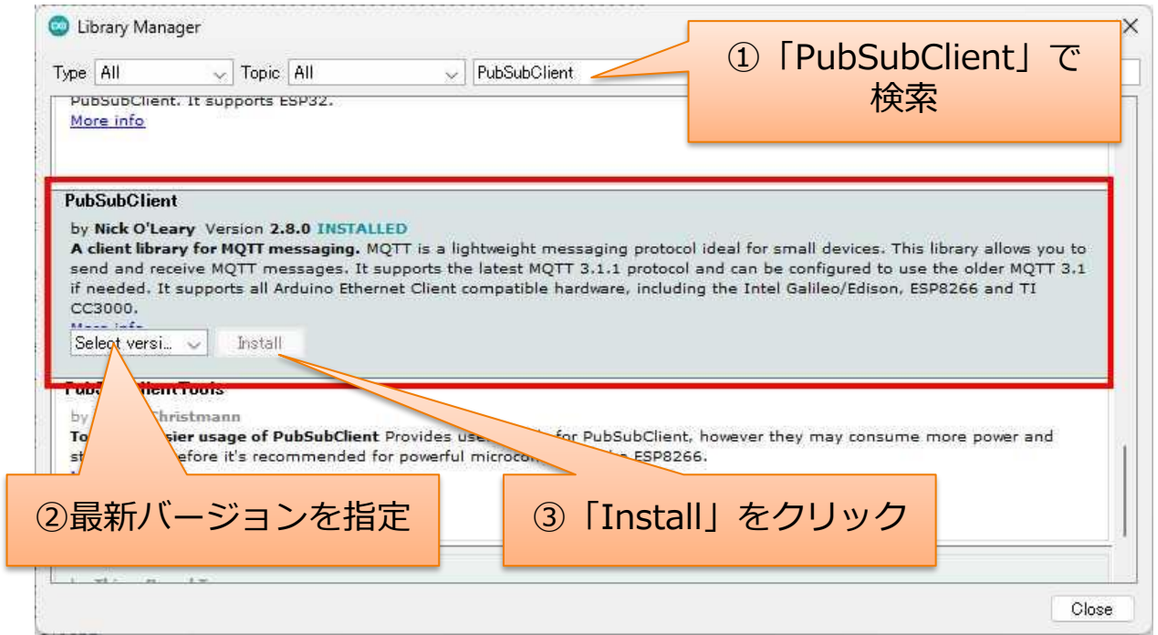
4. ライブラリの追加

ライブラリマネージャーの起動

ライブラリマネージャーを起動し
以下の２つのライブラリを追加

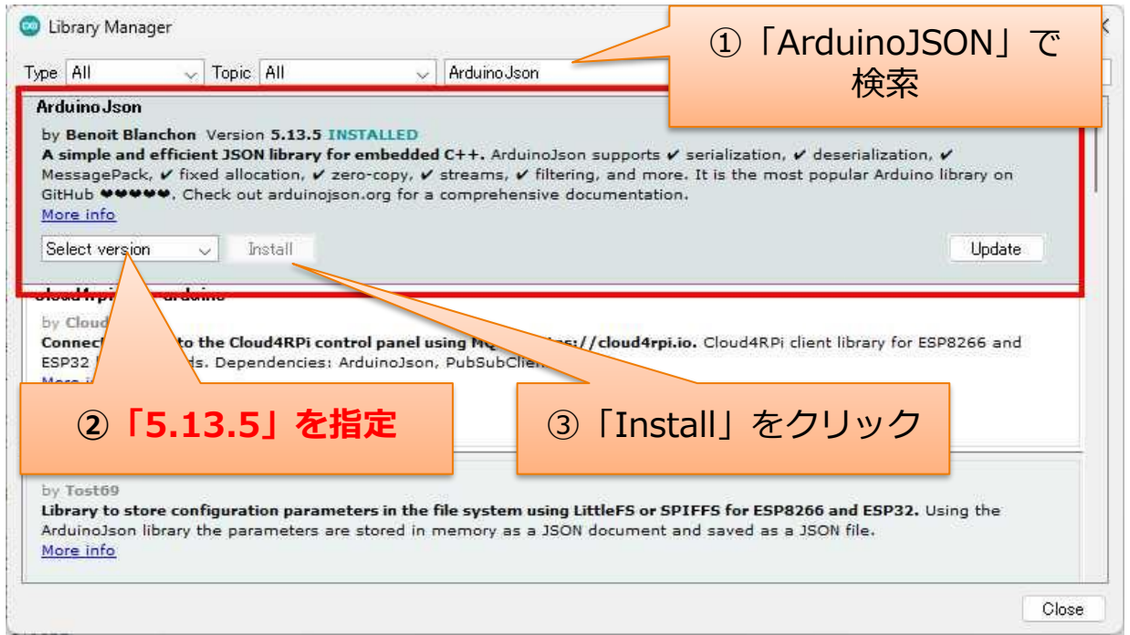
1)「PubSubClient」のインストール

MQTT機能を追加します。



2)「ArduinoJSON」のインストール

MQTTで受信するJSON形式データを容易に扱えるようにします。



5. プログラムのファイル構成

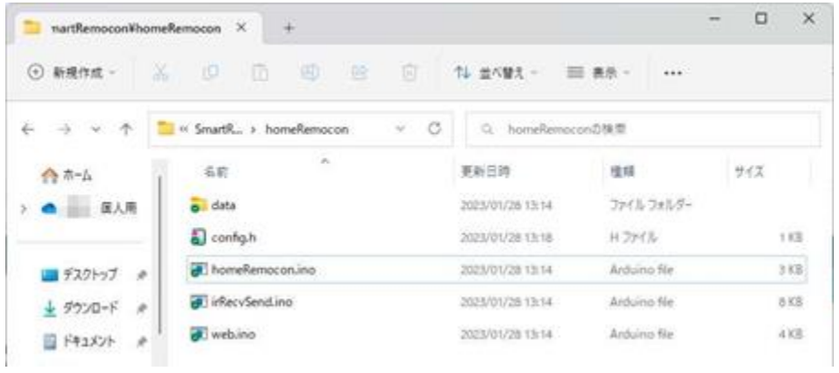
●ファイル構成

プログラム種別

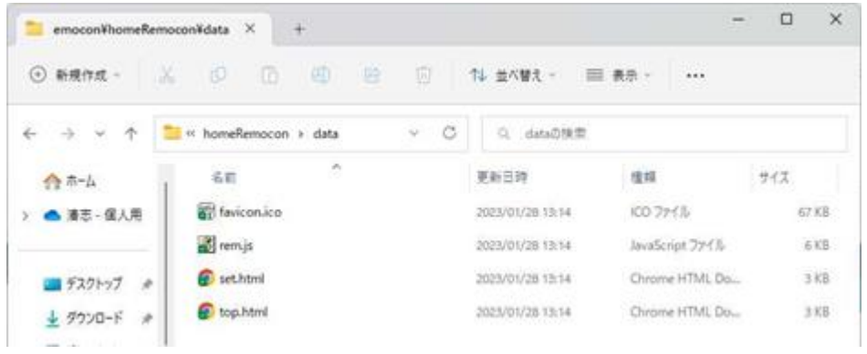
outdoor.html　：　屋外のスマホから操作する制御画面
（スマホのローカルファイルとして利用）　HTML Javascript

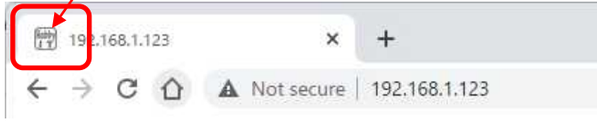
smartRemocon
- smartRemocon.ino　：　setup関数やloop関数、MQTT機能などのプログラム
- config.h　：　Wi-FiのSSID、パスワード、IPアドレスなどの環境設定
- irRecvSend.ino　：　リモコン操作に関するプログラム
- web.ino　：　Webサーバに関するプログラム　Arduino
- data
  - top.html　：　トップ画面（ボタン制御画面）
  - set.html　：　設定画面　HTML
  - rem.js　：　ボタン設定や操作を実施　Javascript
  - favicon.ino　：　無くても良い。ブラウザのアイコンマーク

今回のスケッチフォルダ



SPIFFSで利用するdataフォルダ





6. Arduinoプログラム

●SmartRemocon.ino [Global Area]

```
// Load library and Config file
#include <EEPROM.h>
#include <SPIFFS.h>
#include <ESPAsyncWebServer.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>      // Version[5.13.5]
#include "config.h"
```

ライブラリ追加（TLS機能, MQTT, JSON形式データ）

```
bool ledFlag = true;       // LED control flag

// MQTT connection ON/OFF (false when OFF)
bool MQTT_CONNECT = true;
// for MQTT client
WiFiClientSecure wifiClient;            // for MQTT
PubSubClient mqttClient(host, 8883, wifiClient);

// First thing to do when booting
void setup(void) {
```

MQTTのON/OFFスイッチを実装（true:ON, false:OFF）

MQTTの利用を定義

●SmartRemocon.ino [setup Function]

```
  // ------- Set CA certificate to client only when MQTT_CONNECT is ON -------
  if ( MQTT_CONNECT ) {
    wifiClient.setCACert(beebottle_ca_cert);
  }
}
```

MQTTサーバを認証するための証明書を設定

6. Arduinoプログラム

●SmartRemocon.ino [loop/reconnect Function]

```
79 // After setup is complete, repeat processing until power is turned off
80 void loop(void){
81   // ------- only if MQTT_CONNECT is ON -------
82   if ( MQTT_CONNECT ) {
83     // Check the MQTT status and process the MQTT connection if it is not connected
84     if ( !mqttClient.connected() ) {
85       reconnect();
86     }
87     // MQTT client processing
88     mqttClient.loop();
89   }
90 }
91
92 // MQTT connection process
93 void reconnect() {
94   // loop until MQTT connection state
95   while (!mqttClient.connected()) {
96     Serial.println("Attempting MQTT connection...");
97     // MQTT setting information definition
98     String username = "token:";
99     username += channelToken;
100    // MQTT connection process
101    mqttClient.connect(clientID, username.c_str(), NULL);
102    delay(2000);
103  }
104  Serial.println("MQTT connected");
105  // Process setting when receiving MQTT message
106  mqttClient.setCallback(callback);
107  // Configure TOPIC to receive MQTT messages
108  mqttClient.subscribe(topic);
109 }
```

MQTTの接続状態を確認し、接続されていない時には接続関数を呼び出す

MQTTの受信確認などを行う

BeebotteクラウドにMQTTの接続を行う

6. Arduinoプログラム

●SmartRemocon.ino [callback Function]

```
111 // Processing when MQTT message is received
112 void callback(char* topic, byte* payload, unsigned int length) {
113   // save MQTT received message in variable
114   char recvData[MQTT_MAX_PACKET_SIZE];
115   snprintf(recvData, sizeof(recvData), "%s", payload);
116   // Display MQTT received message on serial monitor
117   Serial.print("Message arrived [");
118   Serial.print(topic);
119   Serial.println("] ");
120   Serial.println(recvData);
121   // Parse the JSON format of the received data and save it to a variable
122   StaticJsonBuffer<MQTT_MAX_PACKET_SIZE> jsonBuffer;
123   JsonObject& jsonBuf = jsonBuffer.parseObject(recvData);
124   // If JSON format parsing is not successful, display an error and exit
125   if (!jsonBuf.success()) {
126     Serial.println("parseObject() failed");
127     return;
128   }
129   // Acquire and save the received data (data)
130   const char* parsedPayload = jsonBuf["data"];
131   // Determine if received data (data) exists
132   if (parsedPayload != NULL) {
133     Serial.print("payload: ");
134     Serial.println(parsedPayload);
135     // Transmit remote control number in receive data (data)
136     if (contRemocon( parsedPayload )) {
137       Serial.println("MQTT send OK");
138     } else {
139       Serial.println("MQTT send NG");
140     }
141   }
142 }
```

受信データを取り出す処理

データが受信できれば、リモコン送信処理

MQTTの受信が
あった場合に処理

7. Javascriptプログラム

●outdoor.html [Japascript]

```
17      footer { text-align: right; }
18    --></style>
19    <!-- ##### Javascript ##### -->
20    <script type='text/javascript'>
21  // ################ Beebotte setting ##############
22  var beToken   = "#### TOKEN ####";   // Beebotte Token
23  var beChannel = "### CHANNEL ###";   // Beebotte Channel
24  // ################################################
25
26  // ● Remote control signal processing
27  var irFlg = false;
28  function snd(setNum) {
29    // ● Judgment during processing
30    if (irFlg) {
31      // ●If processing is in progress, display processing and exit.
32      document.getElementById('dispStatus').innerHTML = "<b>Processing</b>";
33      return;
34    }
35    // ● Set the action flag as being processed, and perform display processing during reception
36    irFlg=true;
37    document.getElementById('dispStatus').innerHTML = "<b>Sending remote control</b>";
38    var xhr = new XMLHttpRequest();
39    // ● Make send data
40    var sdata = "{ ¥"data¥":" + setNum +  "}";
41    // ● Create an access URL
42    var url = "https://api.beebotte.com/v1/data/publish/" + beChannel + "/resource1?token=" + beToken;
43    xhr.timeout = 5000;
44    xhr.ontimeout = function(e){
45      irFlg=false;
46      // ● Show failure in status
47      document.getElementById('dispStatus').innerHTML = "<b>Access Timeout Failure!</b>";
48    };
49    xhr.open("POST", url);
50    xhr.setRequestHeader( 'Content-Type', 'application/json' );
51    xhr.send(sdata);
52    xhr.addEventListener("load",function(ev){
53      var resStr = xhr.responseText;
54        // ●When OK is received, the status is displayed in the if statement. Otherwise, display the state inside else
55      if ( resStr.indexOf("true") != -1 ) {
56        document.getElementById('dispStatus').innerHTML = "<b>Transmission Completed!</b>";
57      } else {
58        document.getElementById('dispStatus').innerHTML = "<b>Transmission Failure!</b>";
59      }
60      // ● Return the processing flag
61      irFlg=false;
62    });
63  }
64    </script>
65  </head>
66  <!-- ◆◆◆Body Tag◆◆◆ -->
67  <body class='underTheEarthKai'><center><div id='contents'>
68    <header><h3>Smart Remote controller [OutDoor]</h3></header>
69    <div id='menu'>Controller Screen</div>
```

StyleSheet：デザインに関する内容を設定

Beebotteで取得した値に変更　**必ず変更が必要**

処理中か判断

HTTP Post Request
[https://api.beebotte.com/v1/data
/publish/###CHANNEL###/resource1?token=###TOKEN###]

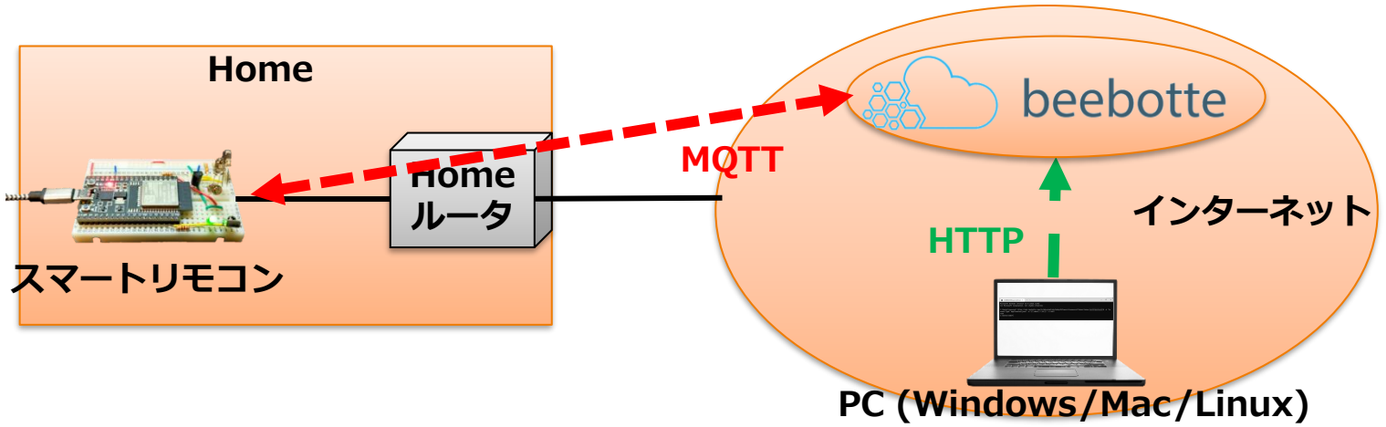
応答によりステータス欄に完了か失敗を表示

Javascript

HTML

8. Beebotte動作確認

●動作確認イメージ



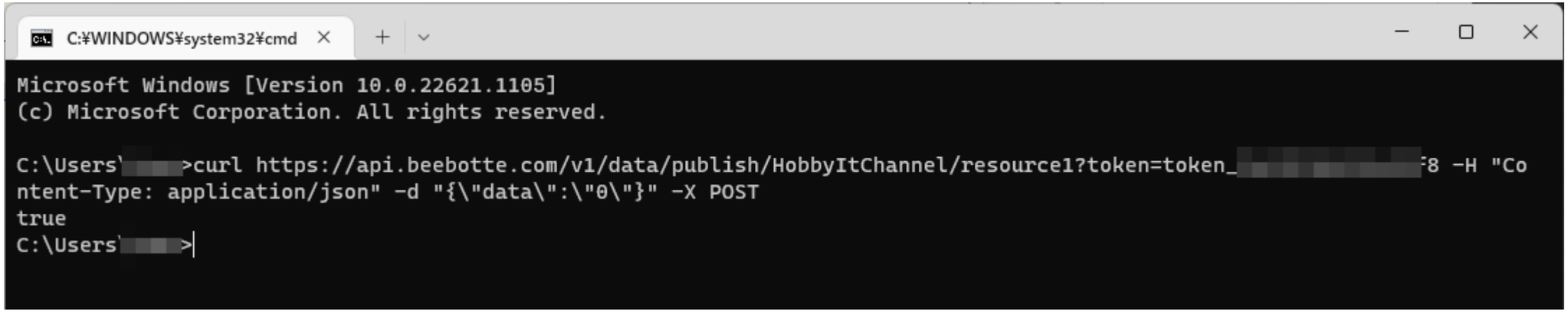
●動作確認コマンド

curl https://api.beebotte.com/v1/data/publish/###CHANNEL###/resource1?token=###TOKEN### -H "Content-Type: application/json" -d "{¥"data¥":¥"0¥"}" -X POST

Beebotte設定値　　Beebotte設定値

ボタン番号（0～9）
HTML画面は1～10なので、-1の値

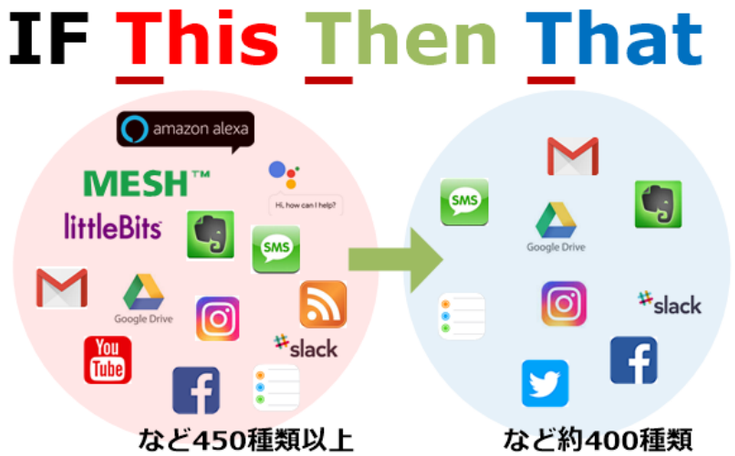
●Windowsコマンドプロンプト（例）



9. IFTTTとは？

IFTTTとは？

https://atmarkit.itmedia.co.jp/ait/articles/1711/22/news031.html

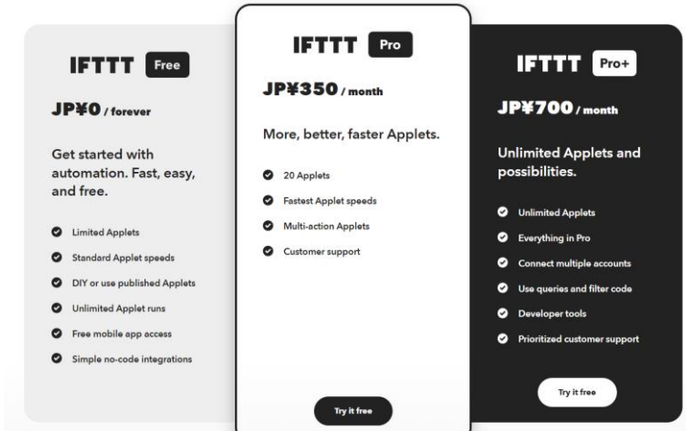


IFTTT（イフト：IF This Then That）
つまり「これ(this)をしたらあれ(that)をする」

インターネット上のクラウドサービスに対して
連携を提供する

料金

https://ifttt.com/plans



無料プランも用意されている。

アプリは3つまででしたが、サービス内容が随時変更されるので、公式サイトを確認下さい。