Outdoor operation, AI speaker linkage Smart Remote Controller

- Outdoor operation using MQTT
- Voice operation by AI speaker cooperation

Contents <<Outdoor operation, AI speaker cooperation>>

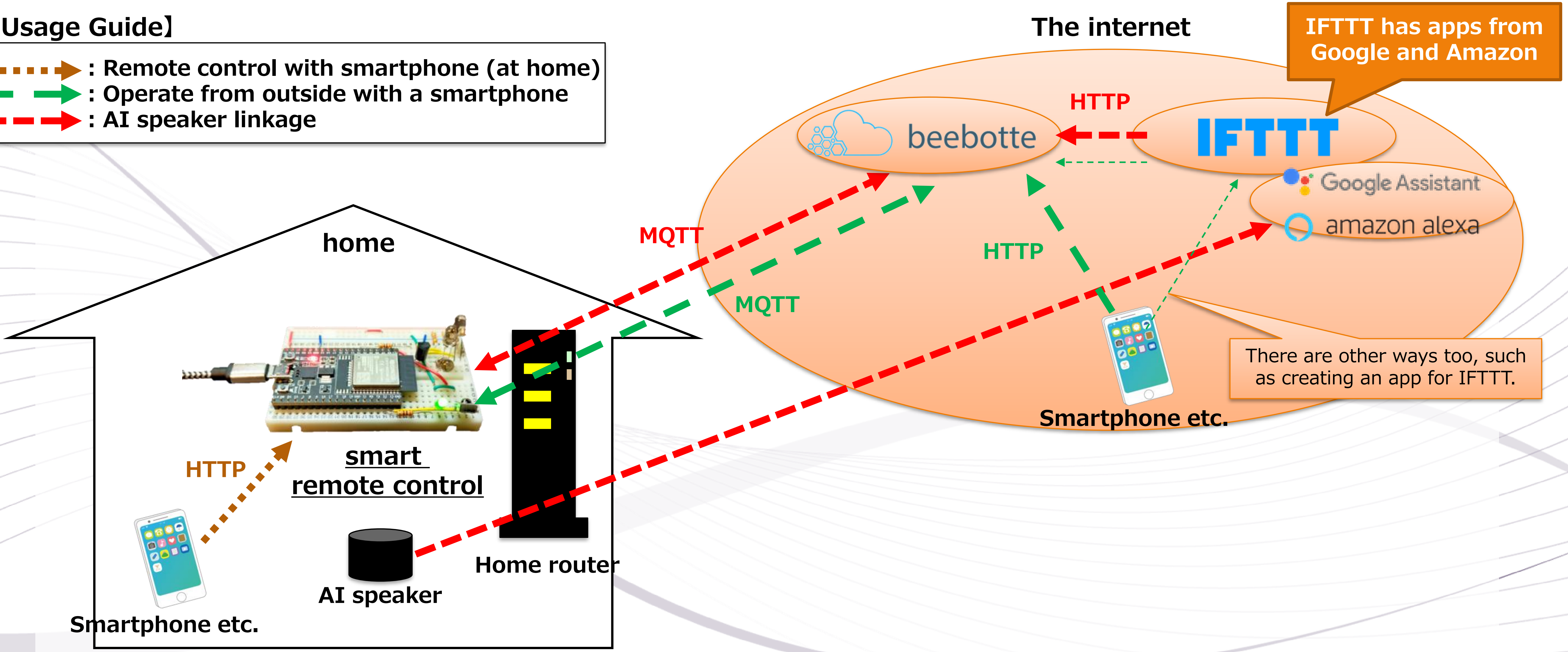
1-1. Overall flow of Smart Remote Controller production

No	Item	Content	Hard	Soft	Note
1	Overview	Overall flow, system configuration, items used, reasons for selection, development environment, etc.	-	-	Delivered in another video
2	LED	Learn the basics for beginners. We will make "L blinking" that lights up and blinks the LED.	○	○	
3	Infrared receiving sensor	Description of infrared receiving sensor Schematic to Wiring, Software	○	○	
4	Infrared transmission LED	Infrared transmission LED description Schematic to Wiring, Software	○	○	
5	LED operation with smartphone(at home)	We will create software to operate the LED with smartphone. (Web server function, SPIFFS operation)	-	○	
6	Remote control with smartphone(at home)	We will create software that to operate the remote control with smartphone indoors. (Button name, signal save/read)	-	○	
7	Operate from outside And AI speaker cooperation	We will create software to operate the remote control with smartphone from the outdoors, and AI speaker cooperation.	-	○	this time this video

2. System configuration

【Usage Guide】

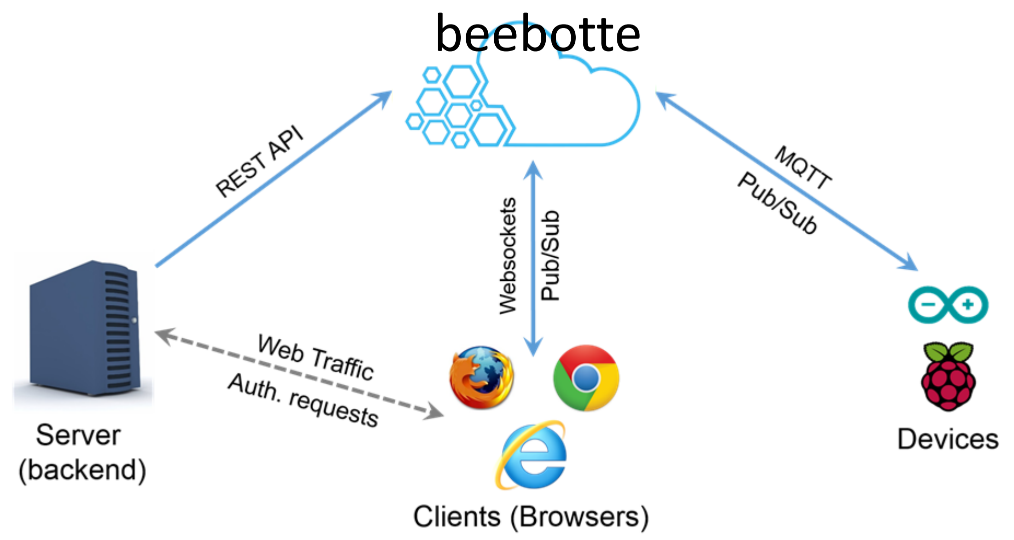
▪▪▪▪▶	: Remote control with smartphone (at home)
‒ ‒▶	: Operate from outside with a smartphone
‒ ‒ ‒▶	: AI speaker linkage



2. What is beebotte?

What is beebotte?

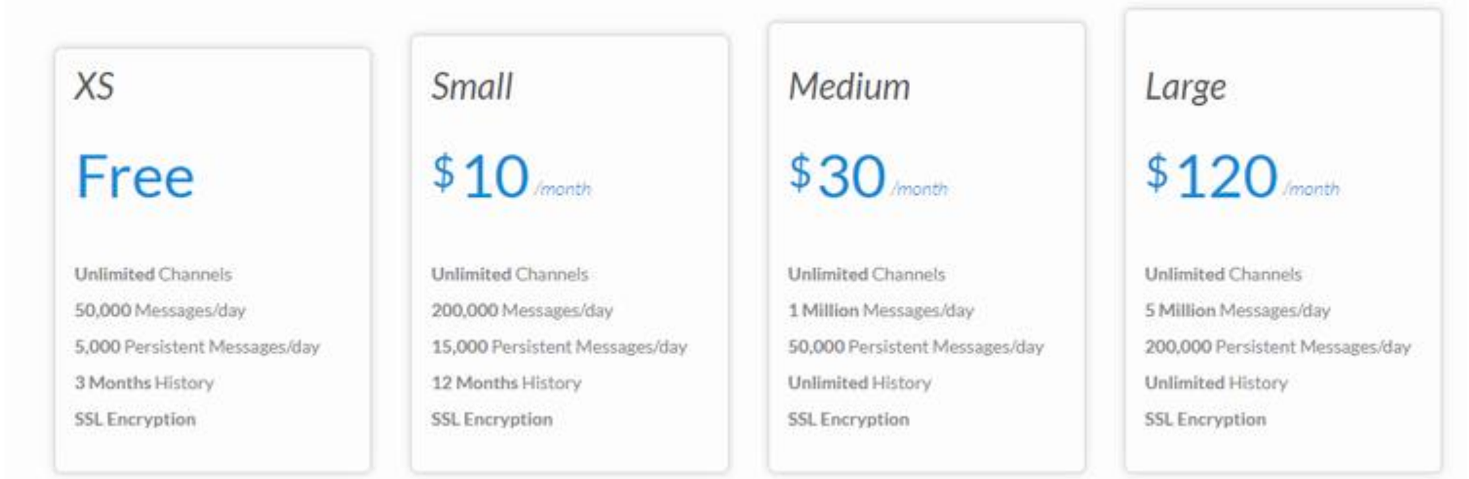
https://beebotte.com/overview



A cloud service that provides websocket connections for MQTT and REST-API [HTTP]
Data communication is possible because it transfers data between different protocols such as MQTT and HTTP.

Price

https://beebotte.com/plans



XS	Small	Medium	Large
Free	$10 /month	$30 /month	$120 /month
Unlimited Channels	Unlimited Channels	Unlimited Channels	Unlimited Channels
50,000 Messages/day	200,000 Messages/day	1 Million Messages/day	5 Million Messages/day
5,000 Persistent Messages/day	15,000 Persistent Messages/day	50,000 Persistent Messages/day	200,000 Persistent Messages/day
3 Months History	12 Months History	Unlimited History	Unlimited History
SSL Encryption	SSL Encryption	SSL Encryption	SSL Encryption

Even if it is free, you can use 50,000 messages a day

● Message volume required for only connection
Requires one KeepAlive every 15 seconds.
4 (time/min) * 60 (min) * 24 (h) * 2 (send/recv) = 11520

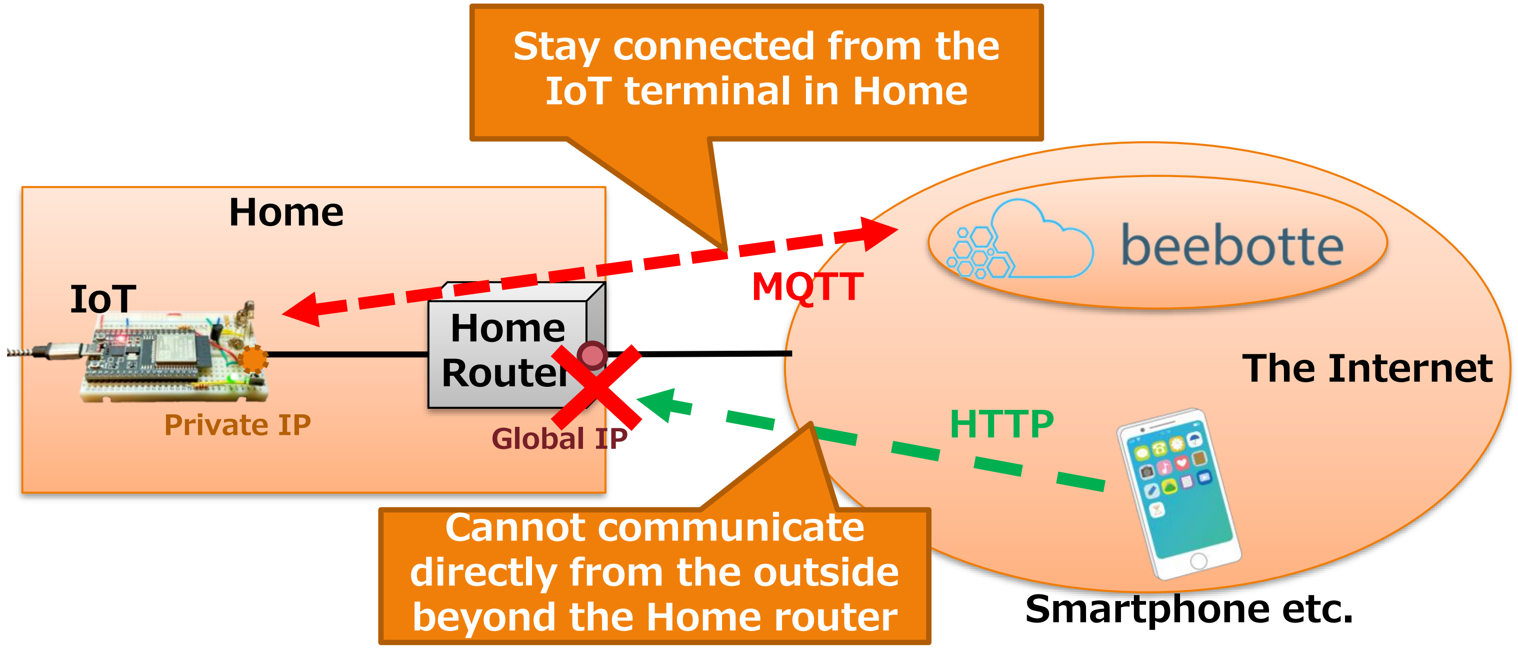
About 12,000 messages

3. What is MQTT?

What is MQTT?

MQTT is an abbreviation of "Message Queuing Telemetry Transport" and is a simple and lightweight protocol developed for devices such as IoT to communicate with each other and the cloud.
Since it has few functions and operates lightly, the consumption of CPU and memory can be kept small, so it can be said that it is suitable for IoT terminals and when you want to reduce resource consumption.
However, it is not suitable for communicating large data such as images, so it is used for sending small information.

Why we need MQTT


Stay connected from the IoT terminal in Home
Home
IoT
Private IP
Home Router
Global IP
MQTT
beebotte
The Internet
HTTP
Smartphone etc.
Cannot communicate directly from the outside beyond the Home router

Terminals in Home cannot be directly accessed from outdoors because they are private IP addresses that can only be used in Home.

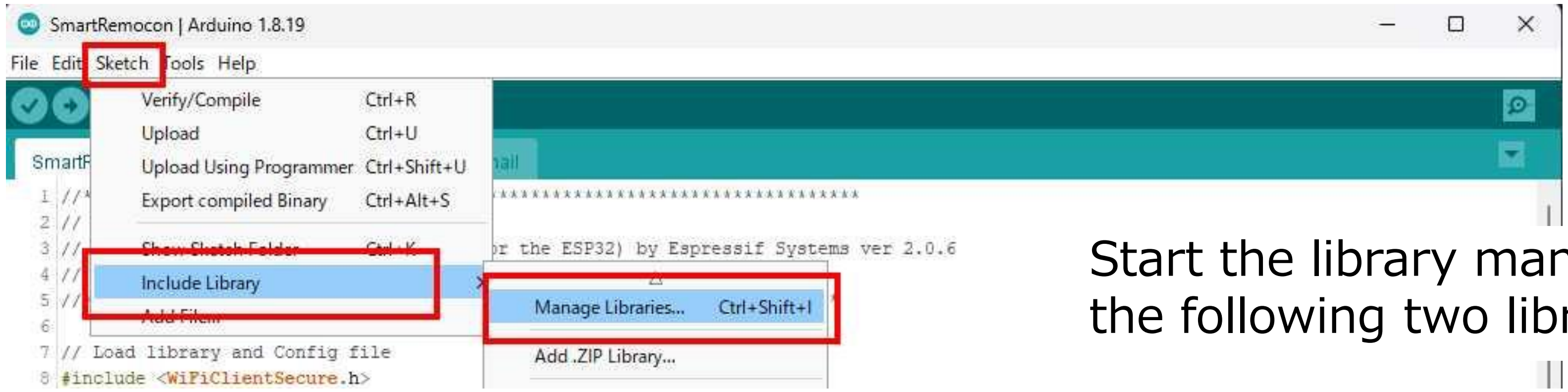
Communication from a terminal in Home is converted to a global IP address by the Home router, so communication is possible.

For this reason, by always performing MQTT communication from the IoT terminal in Home to Beebotte and connecting it, it is possible to access from the outside using that communication.

Hobby-IT channel
《IP address and communication mechanism》
Detailed explanation in the video

4. Add libraries

Launch Library Manager

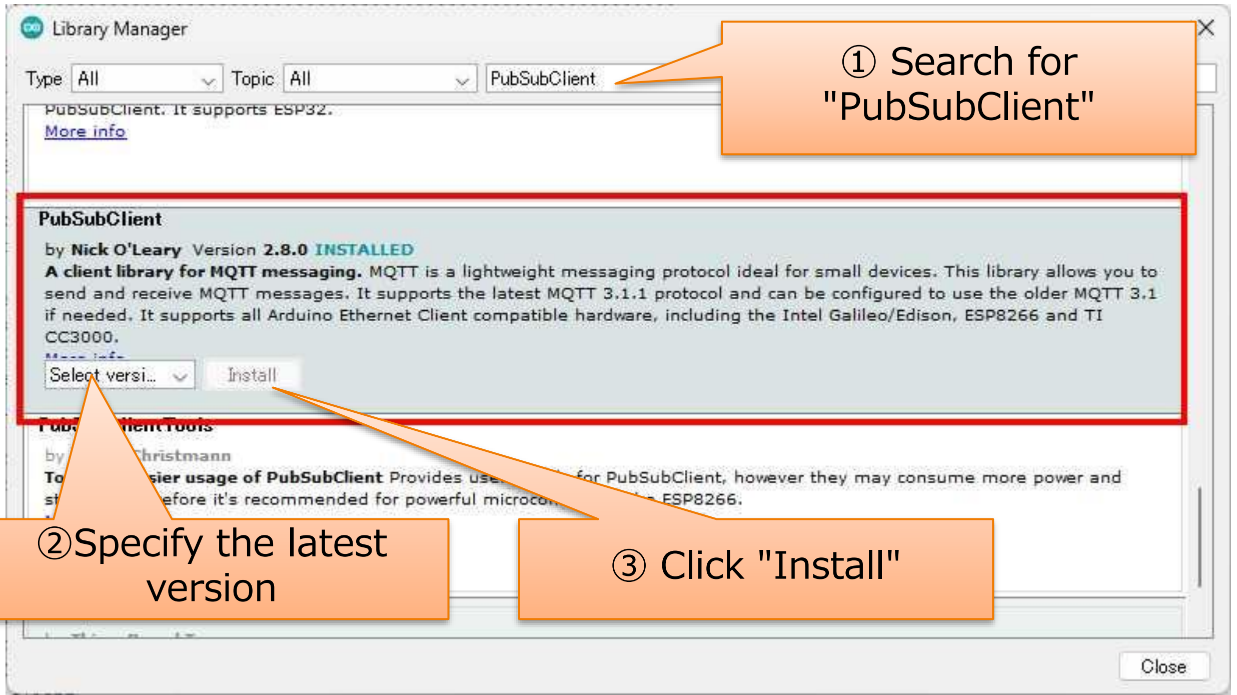

SmartRemocon | Arduino 1.8.19
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...

```
//********************************
//
//  ...or the ESP32) by Espressif Systems ver 2.0.6
//
//

// Load library and Config file
#include <WiFiClientSecure.h>
```

Start the library manager and add the following two libraries

1) Install "PubSubClient"

Add MQTT functionality.



PubSubClient. It supports ESP32.
More info

PubSubClient
by **Nick O'Leary** Version **2.8.0** INSTALLED
A client library for MQTT messaging. MQTT is a lightweight messaging protocol ideal for small devices. This library allows you to send and receive MQTT messages. It supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT 3.1 if needed. It supports all Arduino Ethernet Client compatible hardware, including the Intel Galileo/Edison, ESP8266 and TI CC3000.

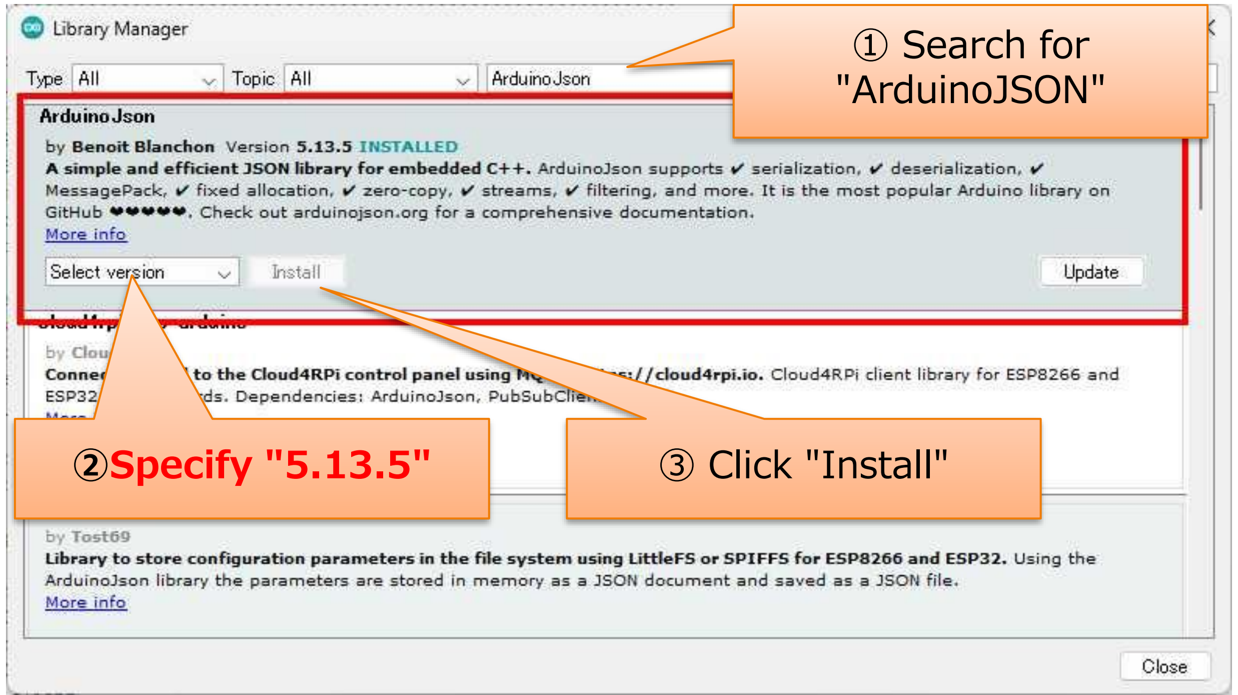
① Search for "PubSubClient"

②Specify the latest version

③ Click "Install"

2) Install "Arduino JSON"

Makes it easy to handle JSON format data received by MQTT.



ArduinoJson
by **Benoit Blanchon** Version **5.13.5** INSTALLED
A simple and efficient JSON library for embedded C++. ArduinoJson supports ✔ serialization, ✔ deserialization, ✔ MessagePack, ✔ fixed allocation, ✔ zero-copy, ✔ streams, ✔ filtering, and more. It is the most popular Arduino library on GitHub ❤❤❤❤❤. Check out arduinojson.org for a comprehensive documentation.
More info

by Tost69
Library to store configuration parameters in the file system using LittleFS or SPIFFS for ESP8266 and ESP32. Using the ArduinoJson library the parameters are stored in memory as a JSON document and saved as a JSON file.
More info

① Search for "ArduinoJSON"

②**Specify "5.13.5"**

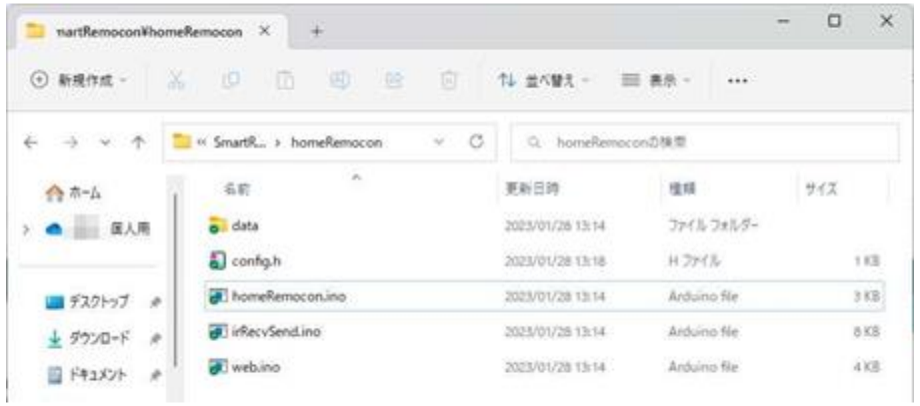
③ Click "Install"

5. File structure of the program

●File structure

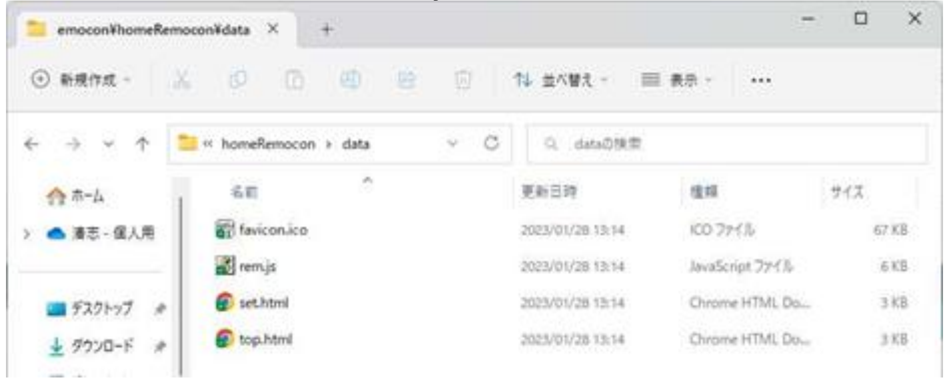
			Program type
outdoor.html		: Control screen operated from an outdoor smartphone (used as a local file on the smartphone)	HTML Javascript
homeRemocon	homeRemocon.ino	: Programs such as setup functions and loop functions	Arduino
	config.h	: Preferences such as Wi-Fi SSID, password, IP address	Arduino
	irRecvSend.ino	: Program related to remote control operation	Arduino
	web.ino	: A program related to the web server	Arduino
	data — top.html	: Top screen (button control screen)	HTML
	data — set.html	: Settings screen	HTML
	data — rem.js	: Perform button settings etc.	Javascript
	data — favicon.ino	: It doesn't have to be. browser icon mark	

This sketch folder

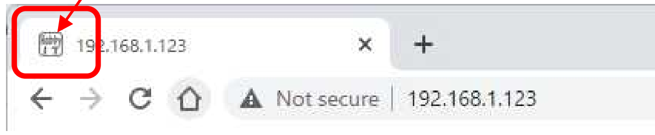


```
nartRemocon¥homeRemocon
新規作成  並べ替え  表示
« SmartR... › homeRemocon        homeRemoconの検索
ホーム
[redacted] 個人用
デスクトップ
ダウンロード
ドキュメント

名前               更新日時            種類              サイズ
data               2023/01/28 13:14   ファイル フォルダー
config.h           2023/01/28 13:18   H ファイル          1 KB
homeRemocon.ino    2023/01/28 13:14   Arduino file       3 KB
irRecvSend.ino     2023/01/28 13:14   Arduino file       8 KB
web.ino            2023/01/28 13:14   Arduino file       4 KB
```

data folder used by SPIFFS

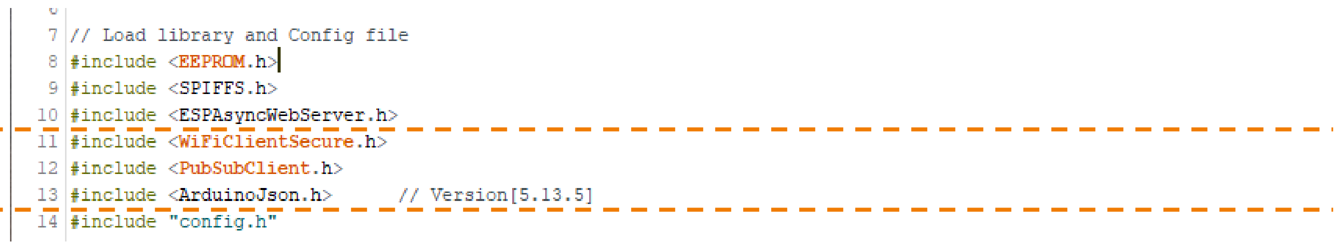


```
emocon¥homeRemocon¥data
新規作成  並べ替え  表示
« homeRemocon › data        dataの検索
ホーム
清志 - 個人用
デスクトップ
ダウンロード

名前           更新日時            種類                 サイズ
favicon.ico    2023/01/28 13:14   ICO ファイル          67 KB
rem.js         2023/01/28 13:14   JavaScript ファイル    6 KB
set.html       2023/01/28 13:14   Chrome HTML Do...     3 KB
top.html       2023/01/28 13:14   Chrome HTML Do...     3 KB
```



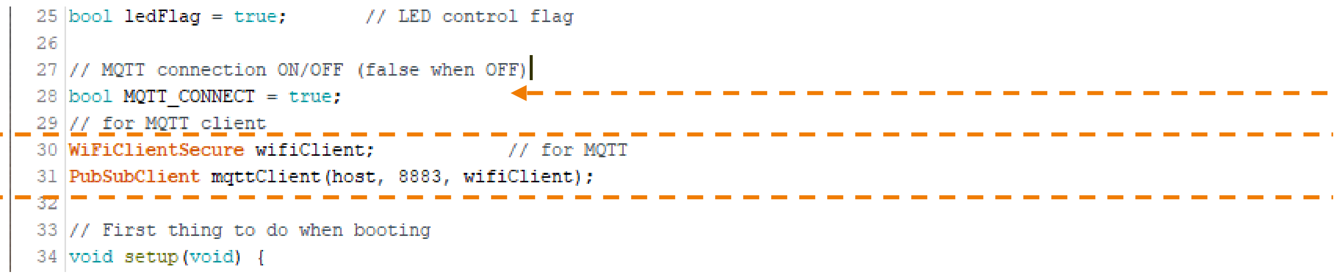
6. Arduino program

●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"
```

Load libraries (TLS function, MQTT, JSON format data)

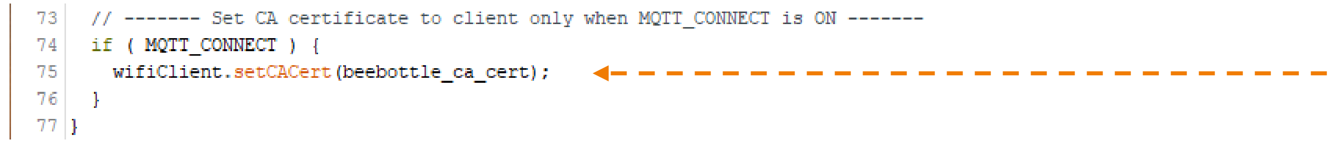


```
bool ledFlag = true;       // LED control flag

// 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) {
```

Implemented MQTT ON/OFF switch (true: ON, false: OFF)

Defining the use of MQTT

●SmartRemocon.ino [setup Function]

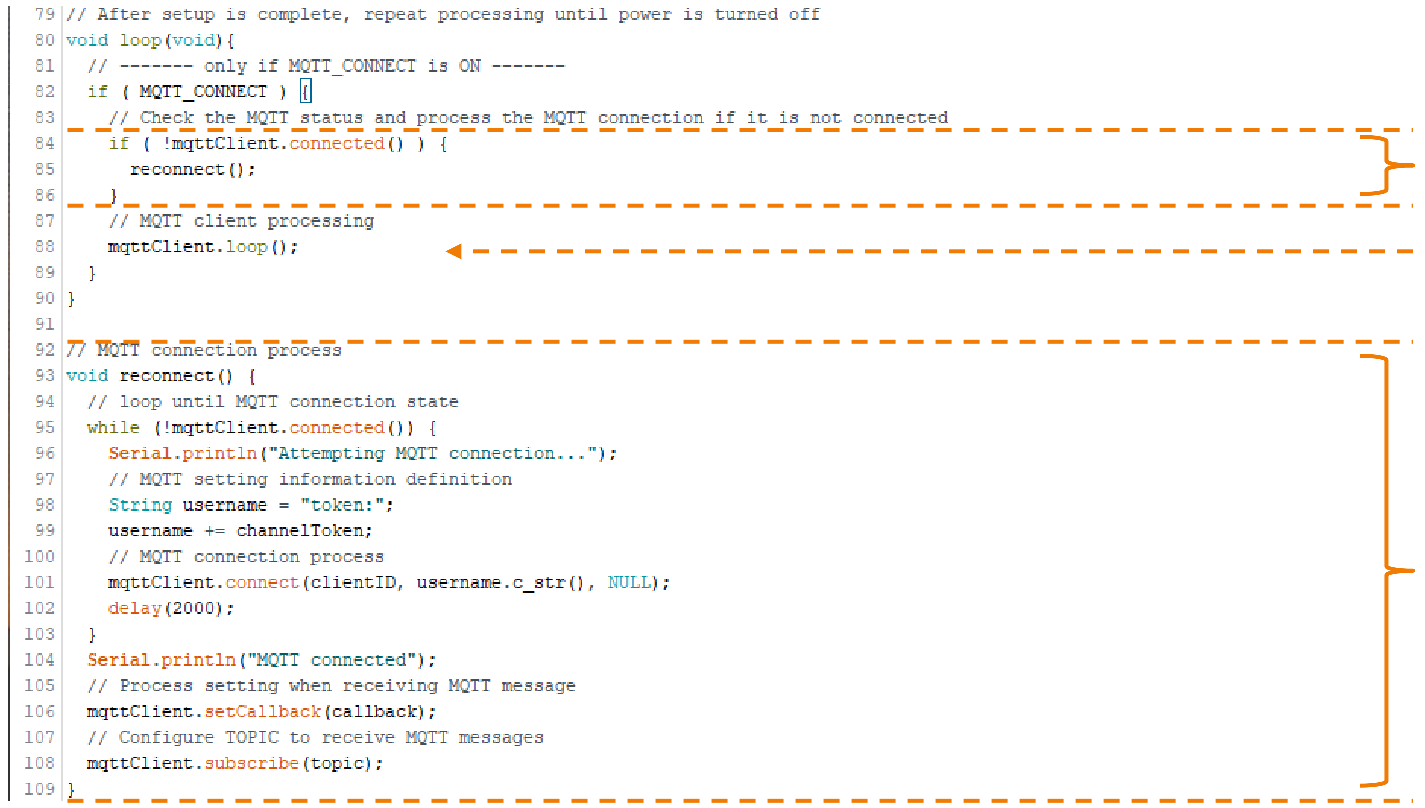


```
  // ------- Set CA certificate to client only when MQTT_CONNECT is ON -------
  if ( MQTT_CONNECT ) {
    wifiClient.setCACert(beebottle_ca_cert);
  }
}
```

Configure a certificate to authenticate the MQTT server

6. Arduino program

●SmartRemocon.ino [loop/reconnect Function]



```
// After setup is complete, repeat processing until power is turned off
void loop(void){
  // ------- only if MQTT_CONNECT is ON -------
  if ( MQTT_CONNECT ) {
    // Check the MQTT status and process the MQTT connection if it is not connected
    if ( !mqttClient.connected() ) {
      reconnect();
    }
    // MQTT client processing
    mqttClient.loop();
  }
}

// MQTT connection process
void reconnect() {
  // loop until MQTT connection state
  while (!mqttClient.connected()) {
    Serial.println("Attempting MQTT connection...");
    // MQTT setting information definition
    String username = "token:";
    username += channelToken;
    // MQTT connection process
    mqttClient.connect(clientID, username.c_str(), NULL);
    delay(2000);
  }
  Serial.println("MQTT connected");
  // Process setting when receiving MQTT message
  mqttClient.setCallback(callback);
  // Configure TOPIC to receive MQTT messages
  mqttClient.subscribe(topic);
}
```

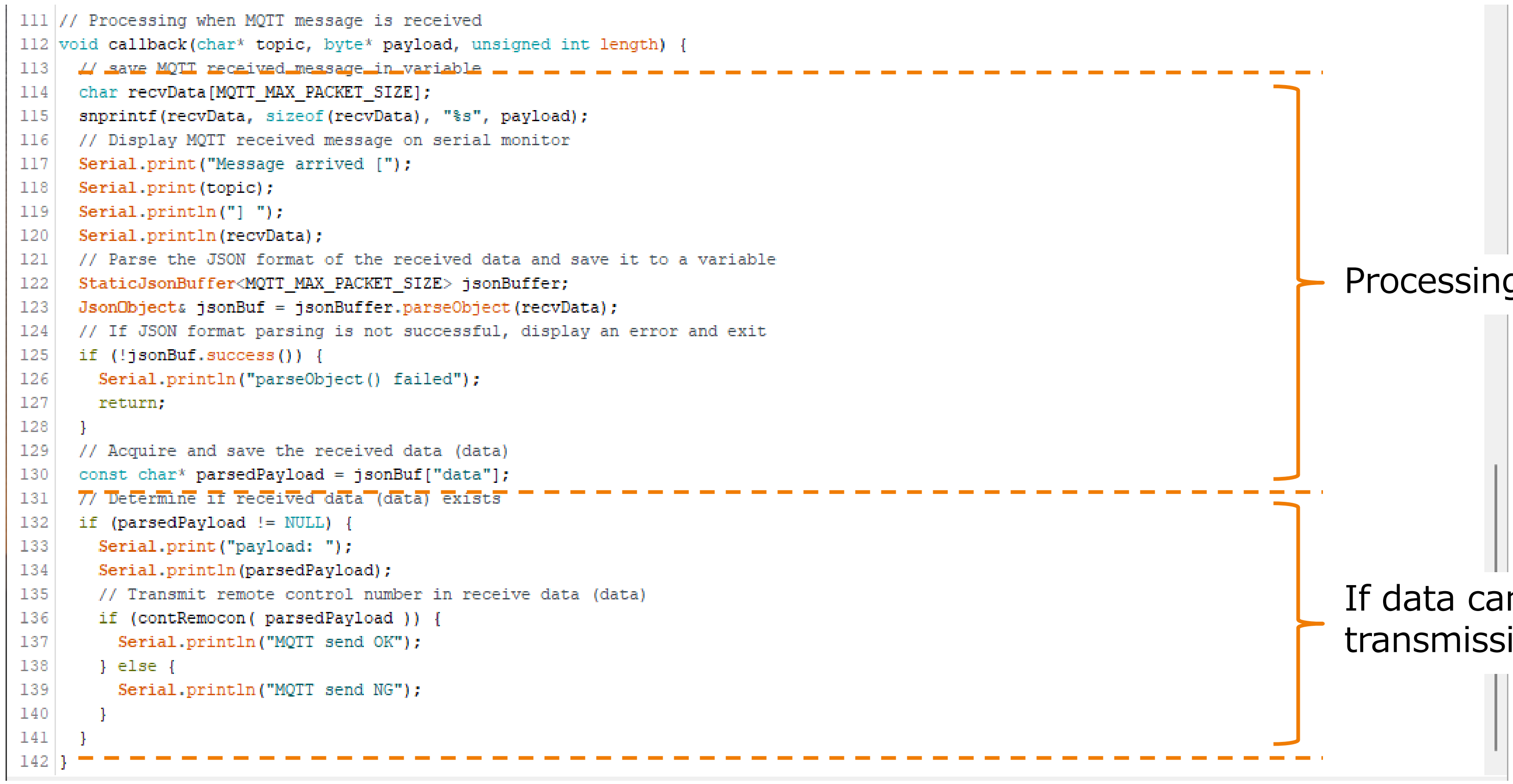
Check the connection status of MQTT and call the connection function when not connected

Confirm receipt of MQTT, etc.

Make MQTT connection to beebotte Cloud

6. Arduino program

●SmartRemocon.ino [callback Function]



```
// Processing when MQTT message is received
void callback(char* topic, byte* payload, unsigned int length) {
  // save MQTT received message in variable
  char recvData[MQTT_MAX_PACKET_SIZE];
  snprintf(recvData, sizeof(recvData), "%s", payload);
  // Display MQTT received message on serial monitor
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.println("] ");
  Serial.println(recvData);
  // Parse the JSON format of the received data and save it to a variable
  StaticJsonBuffer<MQTT_MAX_PACKET_SIZE> jsonBuffer;
  JsonObject& jsonBuf = jsonBuffer.parseObject(recvData);
  // If JSON format parsing is not successful, display an error and exit
  if (!jsonBuf.success()) {
    Serial.println("parseObject() failed");
    return;
  }
  // Acquire and save the received data (data)
  const char* parsedPayload = jsonBuf["data"];
  // Determine if received data (data) exists
  if (parsedPayload != NULL) {
    Serial.print("payload: ");
    Serial.println(parsedPayload);
    // Transmit remote control number in receive data (data)
    if (contRemocon( parsedPayload )) {
      Serial.println("MQTT send OK");
    } else {
      Serial.println("MQTT send NG");
    }
  }
}
```

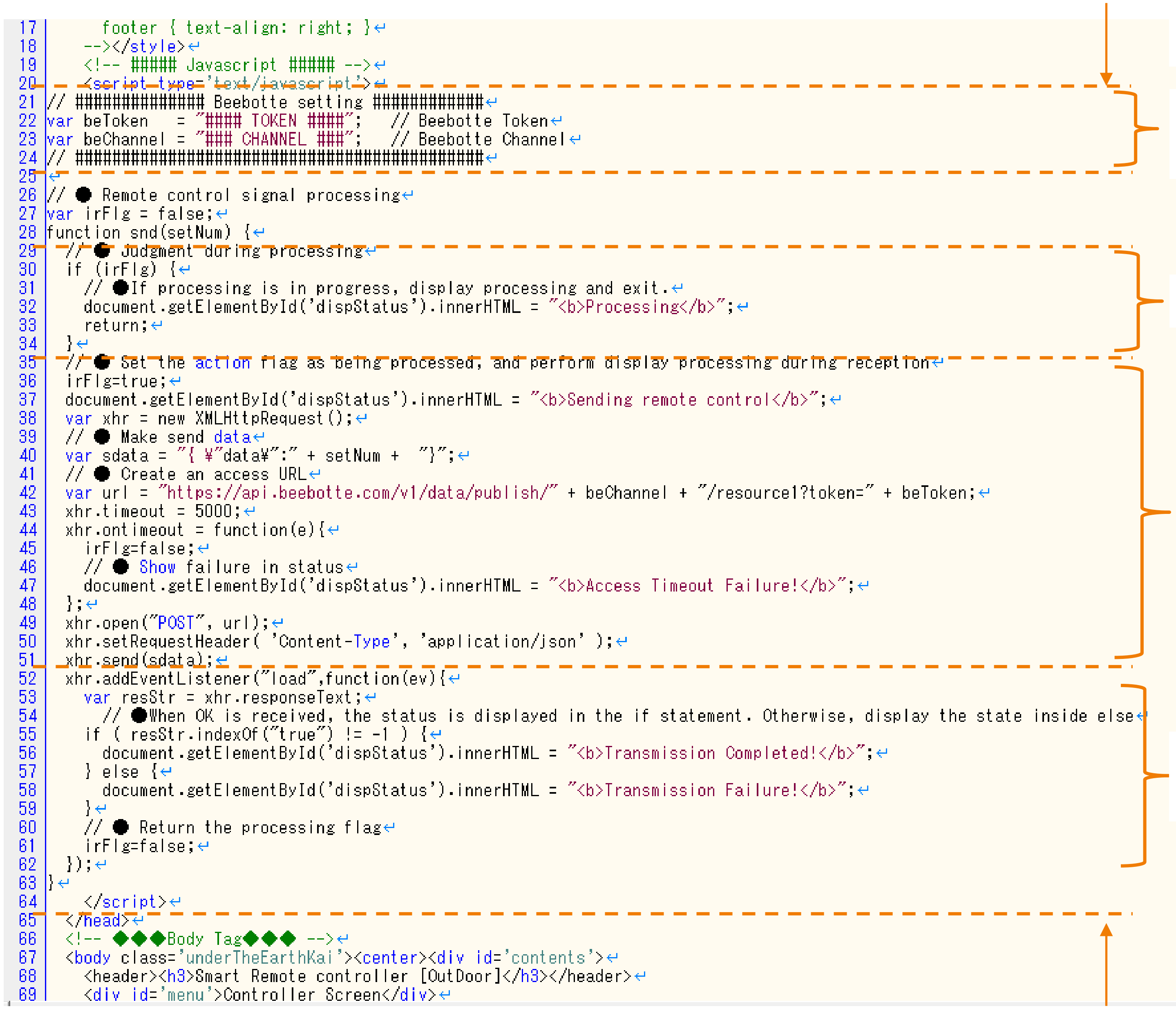
Processing to retrieve received data

If data can be received, remote control transmission processing

Process
when MQTT is received

7. Javascript program

●outdoor.html　[Japascript]



```
      footer { text-align: right; }
    --></style>
    <!-- ##### Javascript ##### -->
    <script type='text/javascript'>
// ############## Beebotte setting ############
var beToken   = "#### TOKEN ####";    // Beebotte Token
var beChannel = "### CHANNEL ###";    // Beebotte Channel
// ############################################

// ● Remote control signal processing
var irFlg = false;
function snd(setNum) {
  // ● Judgment during processing
  if (irFlg) {
    // ●If processing is in progress, display processing and exit.
    document.getElementById('dispStatus').innerHTML = "<b>Processing</b>";
    return;
  }
  // ● Set the action flag as being processed, and perform display processing during reception
  irFlg=true;
  document.getElementById('dispStatus').innerHTML = "<b>Sending remote control</b>";
  var xhr = new XMLHttpRequest();
  // ● Make send data
  var sdata = "{ ¥"data¥":" + setNum +  "}";
  // ● Create an access URL
  var url = "https://api.beebotte.com/v1/data/publish/" + beChannel + "/resource1?token=" + beToken;
  xhr.timeout = 5000;
  xhr.ontimeout = function(e){
    irFlg=false;
    // ● Show failure in status
    document.getElementById('dispStatus').innerHTML = "<b>Access Timeout Failure!</b>";
  };
  xhr.open("POST", url);
  xhr.setRequestHeader( 'Content-Type', 'application/json' );
  xhr.send(sdata);
  xhr.addEventListener("load",function(ev){
    var resStr = xhr.responseText;
      // ●When OK is received, the status is displayed in the if statement. Otherwise, display the state inside else
    if ( resStr.indexOf("true") != -1 ) {
      document.getElementById('dispStatus').innerHTML = "<b>Transmission Completed!</b>";
    } else {
      document.getElementById('dispStatus').innerHTML = "<b>Transmission Failure!</b>";
    }
    // ● Return the processing flag
    irFlg=false;
  });
}
    </script>
  </head>
  <!-- ◆◆◆Body Tag◆◆◆ -->
  <body class='underTheEarthKai'><center><div id='contents'>
    <header><h3>Smart Remote controller [OutDoor]</h3></header>
    <div id='menu'>Controller Screen</div>
```

StyleSheet : Set contents about design

Change to the value obtained by beebotte　**Must change**

Judging whether processing is in progress

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

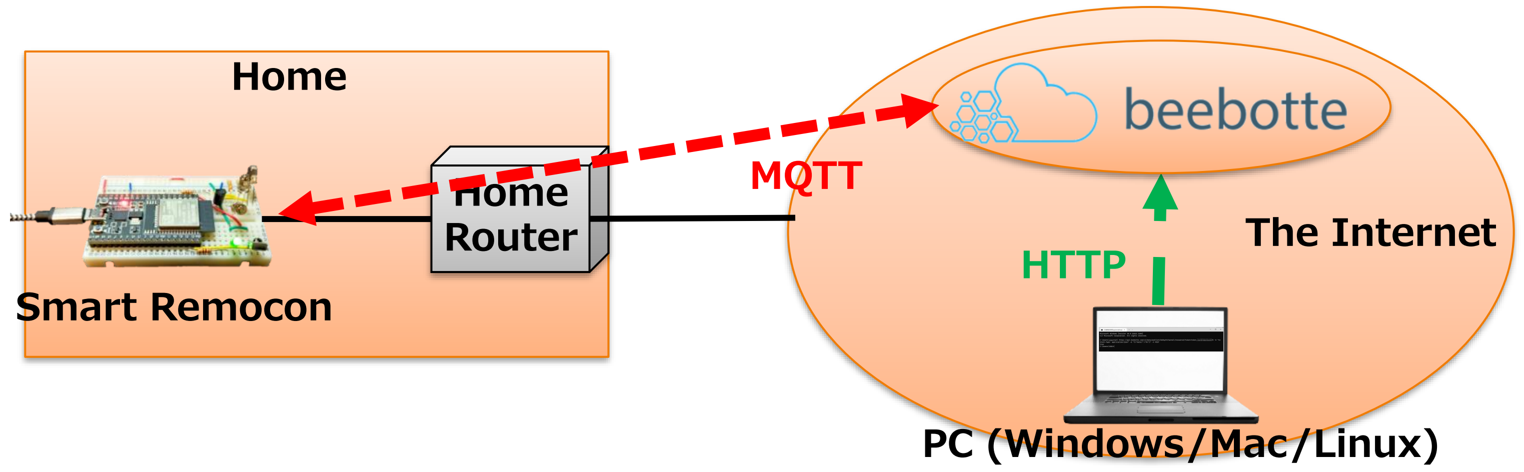
Display completed or failed in the status column depending on the response

Javascript

HTML

8. beebotte operation check

● Operation check image



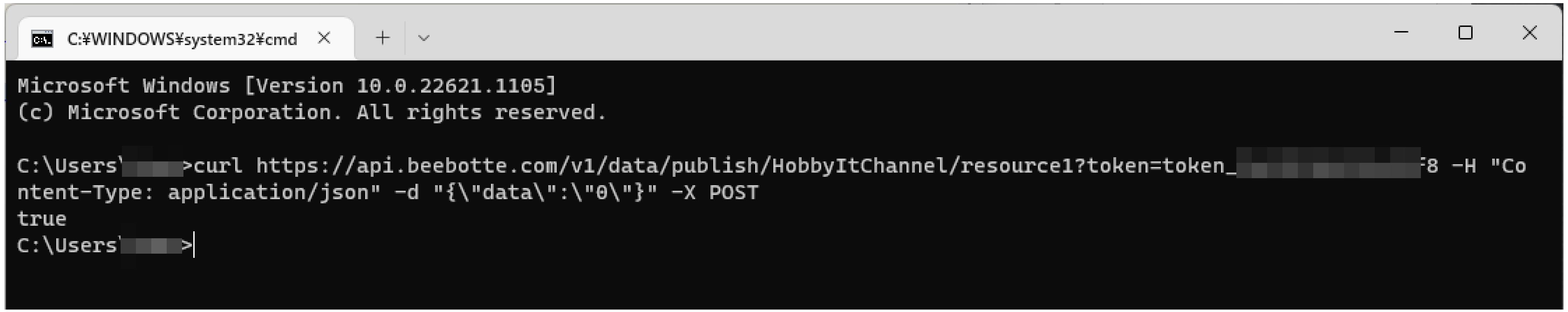
● Operation check command

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

beebotte setting value　　beebotte setting value

Button number (0-9)
HTML screen is 1 to 10, so a value of -1

● Windows command prompt (example)

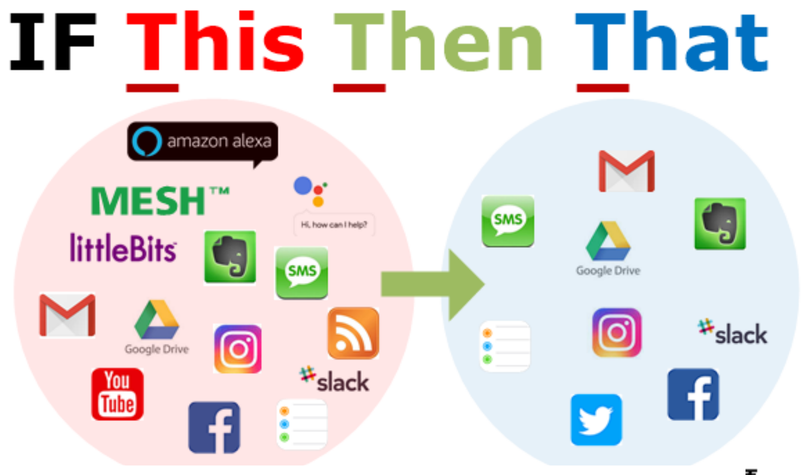


```
C:¥WINDOWS¥system32¥cmd

Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>curl https://api.beebotte.com/v1/data/publish/HobbyItChannel/resource1?token=token_[redacted]8 -H "Co
ntent-Type: application/json" -d "{\"data\":\"0\"}" -X POST
true
C:\Users\[redacted]>
```

9. What is IFTTT?

What is IFTTT?

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



More than 450 types such as

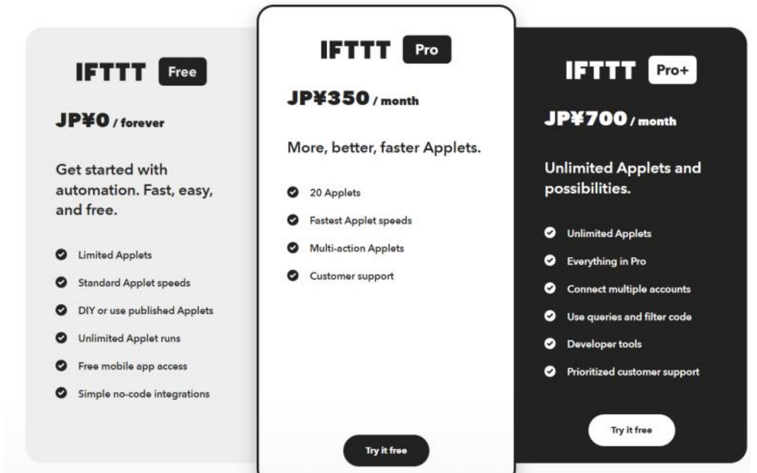
400 types such as

IFTTT（IFTTT : IF This Then That)
In other words, "If you do this, then do that."

Provide integration with cloud services on the Internet

Price

https://ifttt.com/plans



IFTTT Free

JP¥0 / forever

Get started with automation. Fast, easy, and free.

- Limited Applets
- Standard Applet speeds
- DIY or use published Applets
- Unlimited Applet runs
- Free mobile app access
- Simple no-code integrations

IFTTT Pro

JP¥350 / month

More, better, faster Applets.

- 20 Applets
- Fastest Applet speeds
- Multi-action Applets
- Customer support

Try it free

IFTTT Pro+

JP¥700 / month

Unlimited Applets and possibilities.

- Unlimited Applets
- Everything in Pro
- Connect multiple accounts
- Use queries and filter code
- Developer tools
- Prioritized customer support

Try it free

A free plan is also available.

There were up to three applets, but the service contents change from time to time, so please check the official website.