The Bulletin of Seisa Dohto University
No.5, 93－104, 2024

研究ノート

IoT デバイスの遠隔コントロールとプログラミング：クラウドプラットフォームを利用した開発手法について

由　水　　伸

要約

スマート家電製品によるスマートホームの実現が進んでおり，これにより冷暖房，照明，家のセキュリティなどの家庭内制御が自動化されている。また，IoT 機能を持つ小型製品によって，従来の家電もネットワークから制御することが可能となり，応用は広がっている。ただし，異なるメーカーの製品を混在して使用する場合，それぞれのメーカーごとに管理アプリが異なるため操作が複雑になり，機器間の連携にも問題が生じることがある。いま，スマートホームの概念を学校の教室に応用し，例えばスマート教室を想定した場合，このようなメーカーごとに異なるアプリの操作性の問題は利用上の障害となる可能性がある。複数の教員が利用する場合，使いやすいユーザーインターフェースを提供するアプリが必須であり，新たにアプリの作成が必要となる。本稿では，その下準備として，公開されている Web API に基づき IoT デバイスを制御するためのプログラミングの基本的な手法について調査したものをまとめている。

1．はじめに

インターネットに接続してスマートホームを実現するためのスマート家電製品が各社から販売されている。これらを導入することで，例えば冷暖房，照明，家のセキュリティなど，家の中の制御を自動化し便利にするスマートホームが実現できるとされている。

その一方で，IoT 機能を持つ電源コンセント，赤外線リモコン，スイッチ機器，温度計など，図 1-1 のような機能別に特化した小型の製品が販売されている。これらの機材を組み合わせると，従来のスマート化されていない家電製品をネットワークを介してある程度制御することができる。また，必要に応じて機能を追加できるため，目的に合わせた構成が可能となる。

これらの利用（制御）はスマートフォンやタブレット PC などのモバイルデバイスのスマートホーム制御のための専用アプリから操作するのが一般的である。しかし，その専用アプリはメーカーまたは機種ごとに存在する。複数のメーカーの製品を混在してシステムを構成したときは，複数の専用アプリをインストールして切り替えながら使うことになる。その際は，実用面で操作が煩雑になり機器相互の連携に不自由さも生じる。

もし，教室空間を IoT 機器で制御するスマート教室の構想を考えるとすると，この煩雑さが操作の障害となってしまう。複数の教員が利用する場合，なるべくわかりやすい操作性が必要となり，そのためには，特定のモバイルデバイスや制御のための専用アプリに縛られずに，一貫したユーザーインターフェースのもとに各機材を横断的に制御する独自のアプリ作成が求められることになり，Web API を利用したプログラミング手法についての基本的な理解が必要となる。

本稿はそのための基本的な制御手法をまとめた

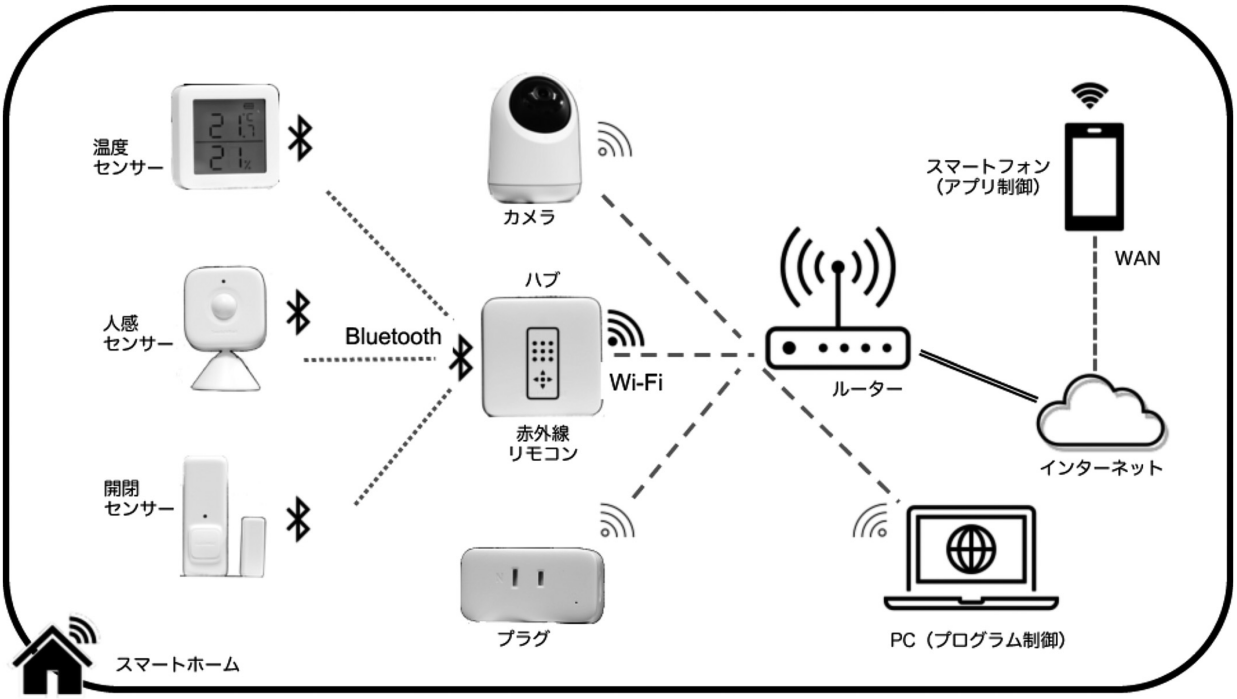

図 1-1　IoT デバイスを使ったスマートホームのネットワーク構成図

ものとなる。

2．IoT デバイスの機種選定について

本研究では，Switchbot API として制御情報を公開している Switchbot 社の製品を選定した。

⑴ Switchbot 社について

Switchbot 社は 2015 年に設立された IoT スマートホームデバイスの企画，製造，及び販売を専門としている，業界の大手である。本社を中国深圳市に置き，日本においては SWITCHBOT 株式会社として現地法人が存在する。その製品は世界 100 以上の国や地域で販売され，販売累計台数は 500 万台を超える。また，日本国内でも 100 万世帯以上のユーザーと 300 万台を超える販売実績がある。

⑵ Switchbot 社の製品展開

Switchbot 社の製品は年々増え，2023 年 12 月現在，30 種類以上の製品群が展開されている。（表 2-1）

表 2-1　主な製品群

ホームオートメーション	8 製品	スマートカメラ	6 製品
スマート家電	5 製品	スマート照明	3 製品
スマートロック	5 製品	スマートプラグ	2 製品
スマートセンサー	2 製品	ロボット掃除機	3 製品

3．IoT 機器を遠隔操作の手順とプラグラミング

⑴ SwitchBot と管理アプリ

SwitchBot デバイスの制御は，通常はスマートフォンのアプリを介して行う。図 3-1 および図 3-2 は iPhone のアプリのホーム画面と「Plug 8EDD」の設定画面の例である。

⑵ 制御に必要な情報の取得

プログラムなどを介して SwitchBot デバイスを制御する場合に必要な情報は「token」と「Secret Key」と「deviceId」である。

・token と Secret Key

このトークン（token）とクライアントシークレット（Secret Key）はアプリの開発者向けオプションに記載されている。本稿ではプログラミングの表記に合わせて，それぞれ「token」,「Secret Key」と表記する。

iPhone の SwitchBot アプリでは「開発者向けオプション」から調べることができる

token はユーザーの認証に使われる。16 進数 96 桁で表され，2 進数 384bit 長である。

Sekret Key はセキュリティ認証に使われる。16 進数 32 桁で表され，2 進数 128bit 長である。

セキュリティ上の問題や何らかの不都合が発生した場合は，既存値をリセットして，新しい値に設定することが可能である。

・Device ID

Device ID は 16 進数 12 桁で表され，2 進数では 48bit 長である。本稿ではプログラミングの表記に合わせて「deviceId」と表記する。

SwitchbotAPI を利用してデバイスリストを取得し，該当の「deviceId」を探す。

SwitchBotAPI v1.0 利用する場合は token があれば取得できる。

SwitchBotAPI v1.1 を利用する場合は，token の他に，Secret Key とアクセス時刻を使って生成された計算値が必要となる。

図 3-1　ホーム画面

図 3-2　設定画面

図 3-3　開発者オプション

⑶ API ドキュメントの参照先

SwitchBot を外部から操作するために必要な API（Application Programming Interface）の情報は以下のサイトに公開されている。

GitHub - OpenWonderLabs/SwitchBotAPI: SwitchBot Open API Documents

https: //github. com/OpenWonderLabs/SwitchBotAPI#send-device-control-commands

API にはデバイスの各パラメーターと，プログラムから利用するために必要な情報が掲載されている。例えば Plug mini（JP）については表 3-1 のように記載されている。

表 3-1　Plug mini (JP)のパラーメータ

Key	Value Type	Description
deviceId	String	device ID
deviceType	String	device type. Plug Mini (JP)
hubDeviceId	String	device's parent Hub ID. 000000000000 when the device itself is a Hub or it is connected through Wi-Fi.
voltage	Float	the voltage of the device, measured in Volt
version	String	the current BLE and Wi-Fi firmware version, e.g. V3.1-6.3
weight	Float	the power consumed in a day, measured in Watts
electricityOfDay	Integer	the duration that the device has been used during a day, measured in minutes
electricCurrent	Float	the current of the device at the moment, measured in Amp

表 3-2　Plug Mini (JP)を制御するコマンド

deviceType	commandType	Command	command parameter	Description
Plug Mini (JP)	command	turnOn	default	set to ON state
Plug Mini (JP)	command	turnOff	default	set to OFF state
Plug Mini (JP)	command	toggle	default	toggle state

⑷ IoT デバイスのコマンド制御

自分の所有する Switchbot デバイスの情報を Unix（Mac・Linux 等）または Windows10/11 のターミナルから，curl コマンドを用いて取得する操作の例を示す。操作結果は JSON 形式で取得できるので，jq で展開表示するものとする。

なお，各 curl コマンドは見やすさのために引数ごとに改行して表記しているが，実際には改行せず，1 行で入力する必要があるので注意のこと。

⑷ Switchbot の全デバイス一覧の取得例

SwitchBotAPI v1.0 を利用したコマンドの例

```
curl -X GET https://api.switch-bot.com/v1.0/devices
        -H 'Authorization:実際の token#16 進数 96 桁' | jq
```

SwitchBotAPI v1.1 を利用したコマンドの例（一部＊で隠蔽）

```
curl -X GET https://api.switch-bot.com/v1.1/devices
        -H 'Authorization:実際の token#16 進数 96 桁'
        -H 'sign:IyQ**************************************4Us ='
        -H 't:1**********3'
        -H 'nonce:c1b6b9f2-****-****-****-1cc4dc5ab974' | jq
```

実行結果

SwitchBotAPI の v1.0，v1.1 のいずれも同様に実行結果 3-1 のように結果が得られる。

実行結果 3-1　全デバイス一覧の取得結果


```
{
  "statusCode": 100,
  "body": {
    "deviceList": [
      {
        "deviceId": "000000000000",
        "deviceName": "プラグミニ",
        "deviceType": "Plug Mini（JP）",
        "enableCloudService": true,
        "hubDeviceId": "000000000000"
      },
      {
        "deviceId": "000000000000",
        "deviceName": "人感センサー",
        "deviceType": "Motion Sensor",
        "enableCloudService": true,
        "hubDeviceId": "000000000000"
      },
      {
        "deviceId": "000000000000",
        "deviceName": "Hub Mini Sapporo",
        "deviceType": "Hub Mini",
        "hubDeviceId": "000000000000"
      },
      {
        "deviceId": "000000000000",
        "deviceName": "リモートボタン Test",
        "deviceType": "Remote",
        "enableCloudService": false,
        "hubDeviceId": "000000000000"
      },
      {
        "deviceId": "000000000000",
        "deviceName": "スイッチボット",
        "deviceType": "Bot",
        "enableCloudService": true,
        "hubDeviceId": "000000000000"
      },
```

```
        {
          "deviceId": "000000000000",
          "deviceName": "玄関外の温度",
          "deviceType": "Meter",
          "enableCloudService": true,
          "hubDeviceId": "000000000000"
        },
      ],
      "infraredRemoteList": [
        {
          "deviceId": "02-000000000000-28634717",
          "deviceName": "テレビ",
          "remoteType": "DIY TV",
          "hubDeviceId": "000000000000"
        },
        {
          "deviceId": "02-000000000000-41071302",
          "deviceName": "ライト",
          "remoteType": "DIY Light",
          "hubDeviceId": "000000000000"
        },
      ]
    },
    "message": "success"
}
```


"deviceId"は隠蔽のため実在の値ではなく"000000000000"と表記していることに注意

実行結果からは，各デバイスの"deviceId"以外の情報も取得できていることがわかる。

(b) デバイスにコマンドを送信する例

Plug mini (JP)をcurlコマンドで制御する例を示す。成功した場合の結果を実行結果　3-2に示す。

SwitchBotAPI v1.0を利用して「Plug mini（JP)」をONにするコマンドの例

```
$ curl -X POST https://api.switch-bot.com/v1.0/devices/{deviceId}/commands
                    -H 'Authorization:実際の token#16 進数 96 桁'
                    -H 'Content-Type: application/json'
                    -d '{"command": "turnOn","parameter": "default","commandType": "command"}'
```

実行結果　3-2 「Plug mini（JP）」を ON にすることに成功した場合

```
{"statusCode":100,"body":{},"message":"success"}
```

SwitchbotAPI v1.0 にしたがって「Plug mini (JP)」を ON にするためは，Switchbot Api1.0 の URL に続けて　/devices/{deviceId}/　として deviceId でデバイスを指定する。デバイスに与えるコマンドは，行末の引数
-d '{"command": "turnOn","parameter": "default", "commandType": "command"}'
の部分で{"command": "turnOn"として指示を書く。

OFF にする場合は"command": "turnOn"の部分を"command": "turnOff"とする。

デバイスに用意されている「command」は，SwitchBotAPI のドキュメントに先述の表 3-2 のように示されている。ここから「Plug Mini (JP)」の場合は Command 列の"turnOn", "turnOff", "toggle"が利用できることがわかる。

なお，旧型の「Plug」の場合は，"turnOn", "turnOff"しかない。デバイスの外見が類似していても型式が異なる場合があるので正確に確認する必要がある。

(c) python 3 を使ったプログラミング

GitHub に公開された SwitchbotAPI のドキュメントには，デバイスのパラメータやコマンド一覧以外に，Python 2，Python 3，JavaScript，C#，Java 11 +，PHP などのプログラムサンプルが掲載されている。

SwitchBotAPI v1.0 を利用した例

SwitchBotAPI v1.0 を利用して「Plug Mini (JP)」を ON にする例を Python 3 で記述した例を示す。

SwitchBotAPI v1.0 を利用してスマートプラグを ON にする Python 3 のコード

```
import requests

token = '実際の token' #16 進数 96 桁
deviceId = 'Plug Mini (JP)の deviceId' #16 進数 12 桁

url = f'https://api.switch-bot.com/v1.0/devices/{deviceId}/commands'
headers = {'Authorization': token, 'Content-Type': 'application/json'}
payload = '{"command": "turnOn"}'
response = requests.post(url, headers=headers, data=payload)
print(response.status_code)
print(response.status_code)
```

SwitchBotAPI v1.1 を利用した例

SwitchBotAPI v1.1 を利用して「Plug Mini (JP)」を ON にする例を Python 3 で記述した例を示す。SwitchBotAPI v1.0 利用の場合に比べプログラムは複雑になる。

SwitchBotAPI v1.1 を利用してスマートプラグを OFF にする Python 3 のコード

```
import json
import time
import hashlib
import hmac
import base64
import uuid
import requests

deviceId = 'Plug Mini (JP)の deviceId' #16 進数 12 桁
apiHeader = {}
token = '実際の token' #16 進数 96 桁
secret = '実際の Secret Key' #16 進数 32 桁
nonce = uuid.uuid4()
t = int(round(time.time() * 1000))
string_to_sign = '{}{}{}'.format(token, t, nonce)

string_to_sign = bytes(string_to_sign, 'utf-8')
secret = bytes(secret, 'utf-8')
sign = base64.b64encode(hmac.new(secret, msg=string_to_sign, digestmod=hashlib.sha256).digest())

apiHeader['Authorization']=token
apiHeader['Content-Type']='application/json'
apiHeader['charset']='utf8'
apiHeader['t']=str(t)
apiHeader['sign']=str(sign, 'utf-8')
apiHeader['nonce']=str(nonce)

url = f'https://api.switch-bot.com/v1.1/devices/{deviceId}/commands'
payload = '{"command":"turnOn"}'
response = requests.post(url, headers=apiHeader, data=payload)
print(response.status_code)
```

セキュリティが高められた SwitchbotAPI v1.1

SwitchbotAPI v1.1 では SwitchbotAPI v1.0 よりセキュリティが高められたことで，認証が複雑になった。SwitchbotAPI v1.0 では"Authorization:" に token を記述するだけで利用できるのに対し，SwitchBotAPI v1.1 では，"Authorization:"の token に加えて，"sign:", "t:", "nonce:"と 4 つのパラメータの引き渡しが必要となる。"sign:", "t:", "nonce:" の値は所定の計算式で生成する必要があり，その値の有効時間は 5 分に設定されている。そのため，SwitchBotAPI v1.1 を使ってデバイスを制御するためにはプログラムを用いる必要がある。

Excel の VBA での制御

ここでは，SwitchBotAPI v1.0 用いて，Plug または Pulg Mini（JP/US）を制御する VBA のコードを示す。

SwitchBotAPI v1.1 を利用してスマートプラグを ON/OFF する Excel VBA のコード

```
Private Sub TurnOnButton_Click()
    Call SendSwitchBotPlug("turnOn")
End Sub

Private Sub TurnOffButton_Click()
    Call SendSwitchBotPlug("turnOff")
End Sub

Private Sub SendSwitchBotPlug(command As String)
    Dim Http As Object
    Dim url As String
    Dim token As String
    Dim jsonBody As String
    Dim deviceId As String

    Set Http = CreateObject("MSXML2.XMLHTTP")

    'YOUR_DEVICE_ID は Plug Mini (JP)の deviceId(16 進数 12 桁)に置き換えること
    deviceId = "YOUR_DEVICE_ID"
    url = "https://api.switch-bot.com/v1.0/devices/" & deviceId & "/commands"
    token = " 実際の token を書く"  '16 進数 96 桁
    'jsonBody を作成  # ダブルコーテーション(")を式に含めるための, ""と """"に注意
    jsonBody = "{""command"": """" & command & """", ""parameter"": ""default"", _
                ""commandType"": ""command""}"
    'HTTP リクエストを送信
    With Http
        .Open "POST", url, False
        .SetRequestHeader "Content-Type", "application/json"
        .SetRequestHeader "Authorization", token
        .Send (jsonBody)
        '応答ステータスを表示（不要ならコメントアウトか削除）
        MsgBox .ResponseText
    End With
End Sub
```

⑸ SwichBot 社以外のデバイス制御について

IoT デバイスを扱うメーカーは SwichBot 社以外にもあるが，ここでは TP-Link 社を取り上げる。

TP-Link 社から提供されている API ドキュメントは限られている。不足分する情報をインターネットコミュニティなどから事例や情報を集めることで，制御プログラムを作成することは可能である。しかし，その場合，非公式な情報を元にしている以上，発生した結果については自己責任となる。その点，十分な注意が必要である。

以下はローカルネットワーク上の TP-Link の HS100 スマートプラグを ON にするコマンドの例である。

```
curl -X POST http://192.168.0.100:9999 -d '{"system":{"set_relay_state":{"state": 1 }}}'
```

４．制御手法のまとめ

プログラミングでの制御について

Switchbot デバイスの場合，Switchbot API v1.1 でセキュリティが向上している。その分，プログラムが複雑化するが認証部分に必要な処理は一度作成すれば，以後，使い回しができるのでさほど問題はないと言える。

API が公開されていない IoT デバイスは動作を自分で解析しなければならず，また，予期しない動作やそれによってトラブルが起きる可能性もあるので，慎重な判断が必要である。

セキュリティへの配慮について

Switchbot API v1.0 の場合はユーザーの token，Switchbot API v1.1 の場合はユーザーの token と Secret Key がわかれば，第３者からもデバイスの制御が可能である。token と Secret Key については，ID，パスワード同様，厳重な秘匿が必要である。自身が所有する PC やネットワーク機器のセキュリティ確保は確実に行う必要がある。

なお，token は 384bit，Secret Key は 128bit の長さがあるため，ランダムに生成して偶然一致する確率は少ないと考えられる。また，Secret Key はそのままの形では通信に流さず，時刻等と組み合わせて暗号化し，さらに，生成された値の有効時間を短く制限していることから安全性は確保されていると考えられる。

しかしながら，基本ではあるが，利用する PC のマルウェア対策やネットワークセキュリティの確保は重要である。合わせて，メーカーからアナウンスされる機器のファームウェアの更新やセキュリティ情報には注意を向けておくべきである。

また，物理的な面での安全への配慮は重要である。IoT 接続されたデバイスの操作に伴い，家具や暖房器具が倒れたことにより，破損だけでなく火災の発生につながる場合がある。電源プラグに規定以上の電流が流れ加熱することも考えられる。メーカーの説明書には，そのような事態も想定し，暖房器具には使えないことなど注意事項が記載されているので，必ず読んでおく必要がある。

５．おわりに

本研究における今後の展望として，スマート化された教室やスマートスクールのあり方について検討していく予定である。

Alexa，Siri，Google などのスマートスピーカや AI に基づく音声コントロールインターフェースへの関心がある他，メーカーごとに異なる規格やプラットフォームが混在しても，統一的に扱える手法も検討する必要がある。

この点に関しては，スマート機器の共通規格である Matter の動向に注目している。Matter は，異なるメーカーのスマートホーム製品間の互換性を高めることを目的としたオープンスタンダードであり，様々なデバイスやサービス間の統合を容

易にすると言われている。今後，規格が安定し各社の製品が対応された段階で，検証するつもりである。

参考文献

① GitHub.SwitchBot API v1.0.
https://github.com/OpenWonderLabs/SwitchBotAPI/blob/main/README-v1.0.md,
（アクセス日：2023/12/28）

② GitHub.SwitchBot API v1.1.
https://github.com/OpenWonderLabs/SwitchBotAPI/blob/main/README.md,
（アクセス日：2023/12/28）

③ SWITCHBOT 株式会社．SWITCHBOT　公式 Web サイト．
hhttps://www.switchbot.jp/,
（アクセス日：2023/12/28）

④ GitHub.tp-link-hs110-api.
https://github.com/larmic-iot/tp-link-hs110-api,
（アクセス日：2023/12/28）

⑤ IT NEED SPACE.How to control your TP-Link HS100 smartplug from Internet.
https://itnerd.space/2017/01/22/how-to-control-your-tp-link-hs100-smartplug-from-internet/,
（アクセス日：2023/12/28）

Remote control and programming of IoT devices: Development methods using cloud platforms

YOSHIMIZU Shin

Abstract

The implementation of smart homes using smart home appliances is progressing, leading to the automation of domestic controls such as heating, lighting, and home security. Additionally, small IoT-enabled devices now allow traditional appliances to be controlled via network, expanding their applications. However, when integrating products from different manufacturers, the variation in management apps for each manufacturer complicates operations and can lead to issues with device interoperability. In the context of schools, applying the concept of smart homes to classrooms, such as in the creation of smart classrooms, poses potential challenges in usability due to these different apps. When multiple teachers use the system, an app with an easy-to-use user interface becomes essential, necessitating the development of new applications. This paper summarizes research on basic programming methods for controlling IoT devices, based on publicly available Web APIs, as a preliminary step in this process.